Раздел III

Защита информационных процессов в компьютерных системах

Афанасьев А.С., Присталов П.М.
Научный руководитель: к.т.н., доцент Мали В.А.
г. Пенза ,ПГУ

Лозунг: Анализ угроз

РАЗРАБОТКА МОДЕЛИ УГРОЗ СИСТЕМ ТЕСТИРОВАНИЯ

В работе произведен анализ угроз для систем автоматизированного тестирования с использованием средств вычислительной техники. На основе проведенного анализа разработана модель угроз для двух архитектур систем тестирования: локальной и «клиент-сервер». Выявлены актуальные угрозы для каждой архитектуры и описаны соответствующие модели нарушителя.

Система тестирования, угроза, уязвимость, модель нарушителя.

В последнее время в нашей стране уделяется особое внимание качеству образовательных услуг, о чем свидетельствует принятие и проведение приоритетной национальной программы «Образование», а также множество проводимых в данной сфере реформ. Перестройка системы образования приводит к появлению новых принципов и подходов к проведению учебного процесса, содержанию программ, а также контролю результатов.

Контроль результатов как одна из важнейших задач образовательного учреждения осуществляется различными способами: путем сдачи зачетов, экзаменов, курсовых и дипломных работ, проведением контрольных и самостоятельных работ и т.д. Оценка качества перечисленных выше работ зависит от мнения принимающего их лица, а значит, является субъективной.

К формам контроля знаний учащихся, которые в большей степени объективны, следует отнести тестирование, а также проверку знаний независимыми экспертами. И если проведение экспертной оценки знаний является достаточно редким явлением и используется в случаях, когда крайне важно получить объективную оценку знаний ученика или студента (например, при проведении ЕГЭ), то тестирование используется в учебных заведениях весьма часто.

К причинам широкого распространения тестирования можно отнести следующие:

- простота процесса тестирования, а также составления тестов и их проверки;
- относительно высокая объективность результатов тестирования;
- возможность проведения удаленного контроля знаний посредством рассылок ранее составленных тестов;
- возможность автоматизации тестирования.

Автоматизированные системы тестирования в настоящее время чрезвычайно распространены. В учебных заведениях используется множество компьютерных приложений для тестирования знаний обучаемых, различающихся в основном программной архитектурой.

Использование компьютерных систем тестирования, как и любых других компьютерных программ, связано с созданием, накоплением, обработкой, передачей, хранением и уничтожением информации. В рассматриваемом случае некоторая информация, циркулирующая в системе, может представлять интерес для тестируемого. В случае если тестируемый имеет возможность читать, создавать, модифицировать или удалять данные, он, теоретически, сможет повлиять на объективность результатов.

Ясно, что при проверке знаний обучаемых при помощи автоматизированных средств компьютерного тестирования, для таких средств необходимо обеспечить соответствующий уровень защищенности путем разработки, внедрения, применения, и усовершенствования политики безопасности. Для этого необходимо для каждой конкретной архитектуры приложения выявить уязвимости, определить актуальные угрозы, модель нарушителя, а также выработать механизмы защиты.

Целью данной работы является построение модели угроз для различных реализаций систем тестирования, определение актуальных угроз и возможных уязвимостей, а также моделей нарушителя и рекомендаций по защите таких систем от преднамеренных атак.

Предполагается, что результаты исследований, проводимых в рамках данной работы, будут использоваться при проектировании программных систем тестирования студентов, разрабатываемых в Пензенском государственном университете.

Термины и определения

Система тестирования **–** программа или программный комплекс, используемый для проверки знания тестируемого посредством проведения тестирования.

Тестируемый **–** лицо, проходящее тестирование при помощи системы тестирования.

Тестирование **–** процесс проверки знаний тестируемого, осуществляемый в форме тестов.

Разработка информационно-логической модели систем тестирования

Как сказано выше, данная работа проводится с целью определения методов защиты различных реализаций систем тестирования, проектируемых и разрабатываемых в Пензенском государственном университете.

На первом этапе разработки информационно-логической модели систем тестирования необходимо определить цель организации.

В Уставе Пензенского государственного университета сказано, что «основной целью образовательной деятельности университета… является повышение образовательного и культурного уровня населения Российской Федерации, подготовка и воспитание специалистов с высшим образованием и специалистов высшей квалификации по характерному для университетов широкому спектру уровней, направлений, специальностей и научных направлений обучения и в соответствии с действующими образовательными программами».

Достижение цели производится путем выполнения множества функций, среди которых можно выделить следующие основные:

- проведение научных изысканий по различным направлениям;
- подготовка, переподготовка, повышение квалификации работников с высшим образованием и научно-педагогических работников высшей квалификации.

Каждая из описанных функций состоит из ряда более мелких подфункций, так, например, подготовка специалистов предполагает контроль их знаний, проводимый в соответствии с действующими образовательными стандартами.

Одним из методов такого контроля является проведение тестирования студентов по основным образовательным дисциплинам. Зачастую тестирование проводится в компьютерных классах при помощи автоматизированных систем.

Как сказано выше, для таких систем необходимо разрабатывать политику безопасности. Рассматривая подфункцию проведения тестирований как часть функции подготовки, переподготовки и повышения квалификации специалистов, можно представить ее в виде процесса, для которого можно выделить следующие ее характеристики:

- входной информацией является вопросы к тесту, варианты ответов на вопросы теста, ответы тестируемого на вопросы теста;
- процесс обработки информации представляет собой сравнение ответов тестируемого на вопросы теста с правильными ответами;
- выходной информацией является результат прохождения теста в виде оценки тестируемого, выставляемой по определенной шкале.

Описание процесса позволяет идентифицировать информационные ресурсы, которые участвуют в процессе. Ими будут:

- перечень вопросов теста;
- варианты ответов на вопросы теста;
- правильные ответы на вопросы теста;
- ответы тестируемого на вопросы теста;
- результат проверки ответов тестируемого.

По степени критичности относительно доступности могут быть следующие виды информации:

- критическая: информация, без которой работа системы останавливается;
- очень важная: информация, без которой система может работать, но очень короткое время;
- важная: информация, без которой система может работать некоторое время, но рано или поздно она понадобится;
- полезная: информация, без которой система может работать, но ее использование экономит ресурсы;
- несущественная: устаревшая или неиспользуемая информация, не влияющая на работу системы.

По степени критичности относительно целостности могут быть следующие виды информации:

- критическая: информация, несанкционированное изменение которой приведет к неправильной работе системы; последствия модификации необратимы;
- очень важная: информация, несанкционированное изменение которой приведет к неправильной работе системы через некоторое время, если не будут предприняты некоторые действия; последствия модификации необратимы;
- важная: информация, несанкционированное изменение которой приведет к неправильной работе системы через некоторое время, если не будут предприняты некоторые действия; последствия модификации обратимы;
- значимая: информация, несанкционированное изменение которой скажется через некоторое время, но не приведет к сбою в работе системы; последствия модификации обратимы;
- незначимая: информация, несанкционированное изменение которой не скажется на работе системы.

По степени критичности относительно конфиденциальности могут быть выделены следующие виды информации:

- критическая: информация, разглашение которой приведет к невозможности реализации целей системы;
- очень важная: информация, разглашение которой приведет к значительному ущербу, если не будут предприняты некоторые действия;
- важная: информация, разглашение которой приведет к незначительному ущербу, если не будут предприняты некоторые действия;
- значимая: информация, разглашение которой приведет только к моральному ущербу;
- незначимая: информация, разглашение которой не влияет на работу системы.

В табл. 1 представлены оценки степени критичности информационных ресурсов относительно свойств информационной безопасности.

Таблица 1

Степени критичности ИР

	По степени критичности относительно доступности	По степени критичности относительно целостности	По степени критичности относительно конфиденциальности
Перечень вопросов теста	*критическая*	*критическая*	*незначимая*
Варианты ответов на вопросы теста	*критическая*	*критическая*	*очень важная*
Правильные ответы на вопросы теста	*критическая*	*критическая*	*очень важная*
Ответы тестируемого на вопросы теста	*критическая*	*критическая*	*значимая*
Результат проверки ответов тестируемого	*критическая*	*критическая*	*незначимая*

Исходя из данных табл. 1, можно выделить основные угрозы информационным ресурсам процесса тестирования в виде угроз нарушения целостности, доступности и конфиденциальности информации:

- несанкционированное чтение перечня вопросов теста;
- несанкционированная модификация перечня вопросов теста;
- несанкционированное удаление перечня вопросов теста;
- несанкционированное чтение вариантов ответов на вопросы теста;
- несанкционированная модификация вариантов ответов на вопросы теста;
- несанкционированное удаление вариантов ответов на вопросы теста;
- несанкционированное чтение правильных ответов на вопросы теста;
- несанкционированная модификация правильных ответов на вопросы теста;
- несанкционированное удаление правильных ответов на вопросы теста;

- несанкционированное чтение ответов тестируемого на вопросы теста;
- несанкционированная модификация ответов тестируемого на вопросы теста;
- несанкционированное удаление ответов тестируемого на вопросы теста;
- несанкционированное чтение результатов проверки ответов тестируемого;
- несанкционированная модификация результатов проверки ответов тестируемого;
- несанкционированное удаление результатов проверки ответов тестируемого.

Описание угроз для системы тестирования

Построение системы тестирования возможно по нескольким архитектурам, в зависимости от которых изменяется распределение информационных ресурсов и функциональных компонентов. Всего можно выделить две основные архитектуры построения систем тестирования: монолитную и распределенную или архитектуру «клиент – сервер».

При использовании монолитной архитектуры все функции системы реализованы в одном программном модуле, расположенном на рабочем месте тестируемого. Там же расположены все информационные ресурсы системы тестирования.

При использовании архитектуры «клиент – сервер» функции и компоненты системы распределены между двумя взаимодействующими модулями, которые обмениваются данными при помощи линий связи. Информационные ресурсы могут иметь различное распределение между модулями.

В табл. 2 приведен пример описание угроз для системы тестирования для архитектуры «клиент-сервер».

Таблица 2

Пример описания угроз

Угроза несанкционированного чтения правильных ответов для архитектуры «клиент-сервер»	
Уязвимость	Хранение правильных ответов в открытом виде. Возможность доступа к правильным ответам. Возможность сетевого взлома компьютера, на котором хранятся правильные ответы
Метод нападения	Открытие файла с правильными ответами. Чтение оперативной памяти. Проведение сетевой атаки, позволяющей получить доступ к правильным ответам
Объект нападения	Информация, хранящаяся в оперативной памяти и на накопителе
Тип потери	Конфиденциальность
Масштаб ущерба	Высокий
Источник угрозы	Администратор, контролирующий, тестируемый
Опыт	Средний уровень
Знания	Знание функционирования сети и протоколов межсетевого взаимодействия
Доступные ресурсы	Специализированное программное обеспечение
Возможная мотивация действий	Помощь тестируемому, улучшение общих результатов тестирования, улучшение результата самого тестируемого

Приведенный в табл. 2 формат описания может быть применен для всех возможных угроз для системы тестирования. Это позволит сформировать перечень актуальных угроз для различных архитектур. На основании перечня можно будет в дальнейшем определить требования к подсистеме защиты системы тестирования, которые необходимо будет реализовать при ее разработке для получения достаточного уровня защищенности.

Бейфус А.В.
Научный руководитель: Горшенин В.В.
г. Челябинск, ЧелГУ

ОБНАРУЖЕНИЕ ХАКЕРСКОЙ АКТИВНОСТИ ПРИ ПОМОЩИ ОТСЛЕЖИВАНИЯ ОБРАЩЕНИЙ К ФАЙЛАМ В WINDOWS NT (2000, XP, 2003)

Целью данной работы является реализация обнаружения несанкционированного доступа и защиты файлов в системах Windows NT (2000, XP, 2003). Результатом работы является программа – драйвер ядра Windows, реализующий возможности контроля доступа и защиты от несанкционированного доступа системных и пользовательских файлов в ОС Windows NT (2000, XP, 2003).

Мониторинг обращения к файлам, руткиты, Native API функций, запрет доступа к файлу.

После взлома операционной системы хакер может долгое время оставаться незамеченным. Для этого он может использовать руткиты – программы позволяющие скрывать своё присутствие в системе. При этом программы, загруженные злоумышленником на машину жертвы, могут копировать и передавать файлы, содержащие важные данные, например пароли пользователей системы.

Одним из способов обнаружения хакера, скрывающегося таким образом, может быть мониторинг обращения к некоторым важным файлам в системе. Идея состоит в создании некоторого подобия honeypot – ловушки для хакера. При помощи логирования обращений к важным файлам в системе, доступ к которым необходим злоумышленнику после взлома, мы можем обнаружить активность хакера, даже если он использует руткиты для сокрытия своих действий.

Работа с файлами в Windows

Рассмотрим реализацию данной идеи в системе Windows NT (2000, XP, 2003). Для работы с файлами в системе Windows приложения используют API-функции. Но Windows NT была спроектирована таким образом, чтобы дать возможность исполняться не только Win32 приложениям, но так же Win16, MsDos, OS/2v 1.x и POSIX. Для этого в Windows NT были созданы встроенные подсистемы, реализующие функции соответствующих систем. Функции всех встроенных подсистем сводятся к вызову Native API-функций. В регистр EAX помещается некоторый условный номер (индекс) функции. В регистр EDX помещается указатель на стек переданных функции аргументов за вычетом 4 байт, адреса точки возврата. Затем происходит переключение режима процессора командой вызова прерывания INT 0x2E, зарезервированного как точка входа в обработчики системных сервисов.

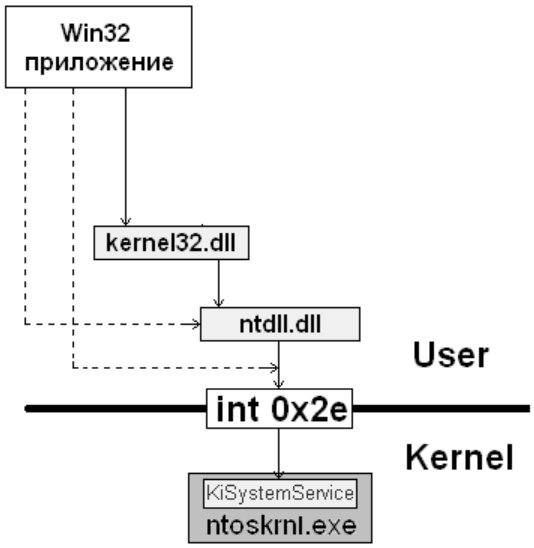


Рис. 1. Взаимодействие режима пользователя и режима ядра с Windows

В ядре ОС существует структура данных, экспортируемая модулем ntoskrnl.exe, называемая таблицей дескрипторов системных сервисов:

```
typedef struct _SERVICE_DESCRIPTOR_TABLE {
SYSTEM_SERVICE_TABLE NtoskrnlTable;// ntoskrnl.exe (native api)
SYSTEM_SERVICE_TABLE Table2;//определяется пользователем, либо для
win32k.sys
SYSTEM_SERVICE_TABLE Table3;// определяется пользователем, либо для IIS
SYSTEM_SERVICE_TABLE Table4;// определяется пользователем
}
SERVICE_DESCRIPTOR_TABLE,
* PSERVICE_DESCRIPTOR_TABLE,
**PPSERVICE_DESCRIPTOR_TABLE;
```

её первая подструктура NtoskrnlTable является таблицей системных сервисов:

```
typedef struct _SYSTEM_SERVICE_TABLE{
 PNTPROC ServiceTable; // указатель на массив точек входа в обработчики
 PDWORD  CounterTable; // счётчик вызовов сервисов (не используется)
 DWORD   ServiceLimit; // количество поддерживаемых сервисов
 PBYTE   ArgumentTable; // указатель на массив аргументов сервисов
}
SYSTEM_SERVICE_TABLE,
* PSYSTEM_SERVICE_TABLE,
**PPSYSTEM_SERVICE_TABLE;
```

ServiceTable – является указателем на массив адресов обработчиков Native API-функций. Индекс функции, помещаемый в регистр EAX, перед вызовом прерывания INT 0x2E, есть номер элемента в этом массиве. Обработчик прерывания INT

0x2E ─ KiSystemService извлекает адрес функции из массива, и после проведения нескольких проверок корректности данных, вызывает обработчик, расположенный по этому адресу.

В Windows для открытия файлового объекта используются Native API-функций NtCreateFile и NtOpenFile, а для выполнения операций чтения и записи используются функции NtReadFile и NtWriteFile. Кроме того, в Windows существует возможность создать объект – проекцию файла в памяти при помощи функции CreateFileMapping и работать с файлом как с массивом, не используя при этом функции NtReadFile и NtWriteFile. Native API-функцией, которую вызывает CreateFileMapping, является функция NtCreateSection.

Реализация

Таким образом, для логирования обращений к файлам нам достаточно заменить адреса обработчиков функций NtCreateFile, NtOpenFile, NtReadFile, NtWriteFile и NtCreateSection на адрес функции логирования, которая в свою очередь будет проверять, принадлежит ли файл к списку важных файлов и в случае необходимости сохранять запись об обращении к файлу.

Функции NtCreateFile и NtOpenFile имеют следующие прототипы:

```
NTSYSAPI
NTSTATUS
NTAPI
ZwCreateFile(
   OUT PHANDLE FileHandle,
   IN ACCESS_MASK DesiredAccess,
   IN POBJECT_ATTRIBUTES ObjectAttributes,
   OUT PIO_STATUS_BLOCK IoStatusBlock,
   IN PLARGE_INTEGER AllocationSize OPTIONAL,
   IN ULONG FileAttributes,
   IN ULONG ShareAccess,
   IN ULONG CreateDisposition,
   IN ULONG CreateOptions,
   IN PVOID EaBuffer OPTIONAL,
   IN ULONG EaLength
);

NTSYSAPI
NTSTATUS
NTAPI
ZwOpenFile(
   OUT PHANDLE FileHandle,
   IN ACCESS_MASK DesiredAccess,
   IN POBJECT_ATTRIBUTES ObjectAttributes,
   OUT PIO_STATUS_BLOCK IoStatusBlock,
   IN ULONG ShareAccess,
   IN ULONG OpenOptions
);

Параметр ObjectAttributes является указателем на структуру
typedef struct _OBJECT_ATTRIBUTES{
```

```
  ULONG Length;
  HANDLE RootDirectory;
  PUNICODE_STRING ObjectName;
  ULONG Attributes;
  PSECURITY_DESCRIPTOR SecurityDescriptor;
  PSECURITY_DESCRIPTOR SecurityQualityOfService;
}
OBJECT_ATTRIBUTES,
*POBJECT_ATTRIBUTES,
**PPOBJECT_ATTRIBUTES;
```

в которой ObjectName – указатель на юникод строку, содержащую имя открываемого файла.

Функции NtReadFile, NtWriteFile и NtCreateSection имеют прототипы:

```
NTSYSAPI
NTSTATUS
NTAPI
NtReadFile(
   IN HANDLE FileHandle,
   IN HANDLE Event OPTIONAL,
   IN PIO_APC_ROUTINE ApcRoutine OPTIONAL,
   IN PVOID ApcContext OPTIONAL,
   OUT PIO_STATUS_BLOCK IoStatusBlock,
   OUT PVOID Buffer,
   IN ULONG Length,
   IN PLARGE_INTEGER ByteOffset OPTIONAL,
   IN PULONG Key OPTIONAL
);

NTSYSAPI
NTSTATUS
NTAPI
NtWriteFile(
   IN HANDLE FileHandle,
   IN HANDLE Event OPTIONAL,
   IN PIO_APC_ROUTINE ApcRoutine OPTIONAL,
   IN PVOID ApcContext OPTIONAL,
   OUT PIO_STATUS_BLOCK IoStatusBlock,
   IN PVOID Buffer,
   IN ULONG Length,
   IN PLARGE_INTEGER ByteOffset OPTIONAL,
   IN PULONG Key OPTIONAL
);

NTSYSAPI
NTSTATUS
NTAPI
NtCreateSection (
   OUT PHANDLE SectionHandle,
```

```
    IN ACCESS_MASK DesiredAccess,
    IN POBJECT_ATTRIBUTES ObjectAttributes OPTIONAL,
    IN PLARGE_INTEGER MaximumSize OPTIONAL,
    IN ULONG SectionPageProtection,
    IN ULONG AllocationAttributes,
    IN HANDLE FileHandle OPTIONAL
);
```

Параметр FileHandle является описателем файла, с которым производится операция. При помощи функции ObReferenceObjectByHandle() по описателю файла мы можем получить указатель на системную структуру – объект-файл:

```
typedef struct _FILE_OBJECT{
   SHORT Type;
   SHORT Size;
   PDEVICE_OBJECT DeviceObject
   PVPB Vpb;
   PVOID FsContext;
   PVOID FsContext2;
   PSECTION_OBJECT_POINTERS SectionObjectPointer;
   PVOID PrivateCacheMap;
   ULONG FinalStatus;
   PFILE_OBJECT RelatedFileObject;
   BYTE LockOperation;
   BYTE DeletePending;
   BYTE ReadAccess;
   BYTE WriteAccess;
   BYTE DeleteAccess;
   BYTE SharedRead;
   BYTE SharedWrite;
   BYTE SharedDelete;
   ULONG Flags;
   UNICODE_STRING FileName;
   LARGE_INTEGER CurrentByteOffset;
   ULONG Waiters;
   ULONG Busy;
   PKEVENT LastLock;
   KEVENT Lock;
   KEVENT Event;
   PIO_COMPLETION_CONTEXT CompletionContext;
}
FILE_OBJECT,
*PFILE_OBJECT,
**PPFILE_OBJECT;
```

в которой FileName – юникод-строка, содержащая имя открываемого файла.

По имени файла функция логирования может определять, нужно ли сохранять запись об обращении к данному файлу. Кроме того, функция может подменять значение, возвращаемое оригинальным обработчиком, и таким образом запретить доступ к заданному файлу. При этом запись о попытке обращения к

файлу останется в файле лога. Функция логирования может сохранить идентификатор процесса и потока, попытавшегося обратиться к файлу, а также его командную строку, полученную из его структуры Peb, и время попытки обращения к файлу.

Применение

Данную идею можно использовать для обнаружения попыток доступа к важным, в том числе системным файлам. Например, если установить такую защиту для файла SAM, расположенного в каталоге %systemroot%\system32\config\, который хранит хеши паролей пользователей в системе Windows, и его резервной копии, расположенной в каталоге %systemroot%\repair\, то можно обнаружить попытку несанкционированного доступа к ним. Файл SAM захватывается системой для монопольного использования, но его резервная копия доступна любому пользователю в группе Everyone. Поэтому злоумышленник может скопировать резервную копию файла даже если он имеет минимальные права в системе. Если же установить блокировку для защищаемых файлов, то есть подмену возвращаемого значения функций на сообщение об ошибке доступа, то можно не только обнаружить попытку несанкционированного доступа, но и пресечь её.

Так же пользователь может использовать такую систему защиты не только для файлов, хранящих системную информацию, но и для защиты собственной конфиденциальной информации, например баз данных или информации, используемой при электронных платежах.

Кроме того, данную идею можно использовать для создания honeypot – ловушки для хакера, так как при правильном выборе файлов для мониторинга она позволяет обнаружить хакерскую активность и сохранить данные о процессе злоумышленника в файле лога.

БИБЛИОГРФИЧЕСКИЙ СПИСОК

1. WASM.RU Слежение за вызовом функций Native API [Cardinal]
2. *М. Руссинович, Д. Соломон* Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. /Пер. с англ. – 4-е изд. – М.: Издательско-торговый дом «Русская редакция»; СПб.: Питер; 2005. – 992с.: ил.

Жилкин С.Д.
Научный руководитель: к.т.н., доцент Петров В.А.
г. Москва, МИФИ

Лозунг: Выявлять аномалии, сравнивая с эталоном

МОДЕЛИ ПОВЕДЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ВЫЯВЛЕНИЕ АНОМАЛИЙ ПОВЕДЕНИЯ

Данная статья предлагает методы моделирования поведения ПО с целью нахождения аномалий работы данного ПО в будущем. Выявление аномалий прежде всего подразумевает выявление недекларированных возможностей ПО или подмену исполняемого кода. Предложены методы моделирования для ПО с высокой степенью ветвления исполняемого кода, когда поведение ПО заранее непредсказуемо или зависит от пользователя.

Модель работы ПО, аномалии работы ПО, аудит, нейронные сети, НДВ.

Предпосылкой к ведению работ в данном направлении является сложившаяся ситуация в сфере информационной безопасности. На данный момент существует множество антивирусов, препятствующих выполнению вредоносного кода. Большинство из них основываются на сигнатурном и эвристическом методах анализа исполняемого файла для выявления вредоносного кода. Однако куда меньшее внимание уделено проблеме выявления и локализации недекларированных возможностей ПО. Так как зачастую антивирусы проводят анализ начальных и конечных частей файла в поисках вредоносного кода и, учитывая, что внедрение компьютерной закладки, приводящей, например, к НСД, может быть осуществлено в середине файла, можно говорить о том, что антивирусы не предоставляют полной защиты в данной области. Также учитывая, что код, приводящий к НСД, сам по себе может не являться вредоносным, становится очевидным, что одни антивирусы не обеспечивают должный уровень защиты данных от НСД и НДВ.

Основной задачей, рассматриваемой в данной статье, является разработка методов и пути реализации программного комплекса по моделированию поведения ПО и нахождения отклонений от штатной работы ПО. Использование такого комплекса подразумевает следующие шаги:

1) моделирование работы ПО на одном компьютере в режиме, который мы заведомо считаем доверенным;

2) использование полученной модели при анализе работы этого ПО на множестве компьютеров с целью выявления аномального поведения.

С точки зрения моделирования всё ПО можно разделить на два класса. Первый – специализированное алгоритмическое ПО (к такому относятся, например, системные службы, службы, работающие в фоновом режиме, и т.д.) и второй – ПО с пользовательским интерфейсом (к каковому относятся прикладные программные пакеты, например, Adobe Photoshop, 1С, Microsoft Office). В первом случае поведение ПО заранее предопределено или имеет мало ветвлений. Моделирование поведения такого ПО не представляет особой сложности – для этого в некоторых ОС даже имеются стандартные приложения (ps-watcher, pwatch, pScan и прочие для Linux). Во втором случае время работы и последовательность действий ПО неизвестны заранее. Именно для этой категории ПО предназначены методы моделирования и нахождения аномалий, представленные ниже.

О поведении ПО можно судить по его взаимодействию с операционной системой: обращение к жёсткому диску, сетевым ресурсам, вызовы функций драйверов, работа с реестром и прочее. Так как при моделировании и проведении последующего анализа использован математический аппарат, необходимо описывать поведение ПО некоторым набором числовых значений – вектором координат. Предлагается описывать поведение процесса количественными, логическими и статистическими характеристиками. Количественные характеристики могут включать в себя такие параметры, как число обращений к системным файлам, библиотекам и прочим ресурсам системы. Логические характеристики описывают общее поведение ПО: создавались ли сетевые соединения, порождались ли дополнительные процессы и прочее. Статистические характеристики могут показывать частоту выделения дополнительной памяти, обращений к жёсткому диску и прочие частотные параметры. Чем более подробной будет информация о взаимодействии процесса с операционной системой, тем более развёрнутым будет вектор, описывающий поведение ПО, и тем более точными будут результаты моделирования и анализа. В дальнейшем вектор, описывающий поведение ПО за некоторое время работы, будем называть профилем работы (рис. 1).

Рис.1. Профиль поведения ПО

Для того чтобы получать информацию такого рода о поведении ПО, потребуется дополнительная система аудита. Она может основываться на журнальных файлах операционной системы или, например, на более подробных данных, полученных на уровне драйвера. При реализации методов моделирования поведения ПО для сбора информации использовалась система “Инсайдер”. Основными преимуществами данной системы является высокий уровень подробности информации и применимость к таким системам, как Linux, Unix и Solaris, помимо Microsoft Windows. Структура комплекса аудита, совмещённая с комплексом анализа, представлена на рис. 2.

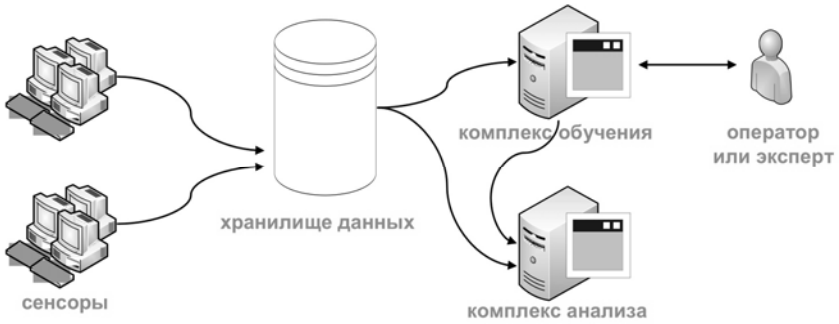


Рис. 2. Структура комплекса моделирования поведения ПО и анализа

Так как уже достаточно долгое время для распознавания образов с успехом используются нейронные сети [1], было принято решение воспользоваться этим математическим аппаратом в данной работе. В предлагаемом методе моделирования и анализа в качестве модели поведения ПО используется некоторая нейронная сеть, на вход которой подаётся профиль поведения (т.е. вектор координат). Выходом нейронной сети является число от 0 до 1, показывающее вероятность соответствия профиля модели [2]. Для того чтобы получить такую сеть, требуется провести процесс обучения, который заключается во множестве калибровок коэффициентов и смещений нейронов методом обратного распространения ошибки с целью получения результата, близкого к единице [3].

Учитывая, что каждый рабочий цикл ПО несколько отличается от предыдущего, для более точного обучения нейронной сети на поведение ПО необходимо произвести множество запусков целевого ПО, пройдя множество рабочих циклов ПО (рис. 3). Во время рабочего цикла желательно выполнить как можно больше различных действий с помощью целевого ПО или выполнить все легитимные действия, разрешённые оператору данного ПО. Дополнительное обучение осуществляется обучением нейронной сети на обратный результат (близкий к 0) на поведении ПО, схожего по функционалу с целевым ПО [4].

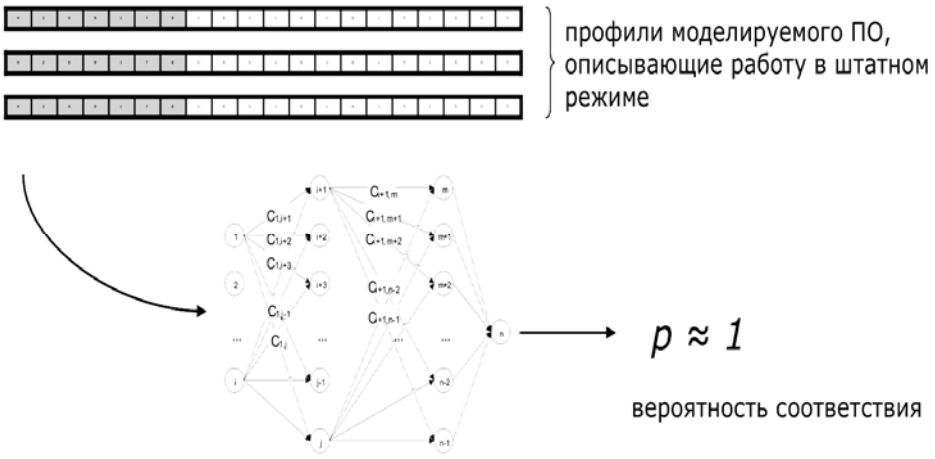


Рис. 3. Обучение нейронной сети на поведение ПО

Учитывая, что поведение целевого ПО заранее неизвестно, необходимо разделять его работу на фазы. В реализации программного комплекса обучения предлагаются алгоритмы, выделяющие в работе ПО фазы трёх типов: запуск, специфические действия и завершение работы. Разделение на фазы работы основывается на анализе активности ПО во времени (условно представлено на рис. 4). Специфические действия ПО включают в себя все действия, совершаемые по запросу пользователя после того, как произошла загрузка ПО. Таких специфических действий может быть несколько. Для каждого действия каждой фазы создаётся и обучается собственная нейронная сеть.

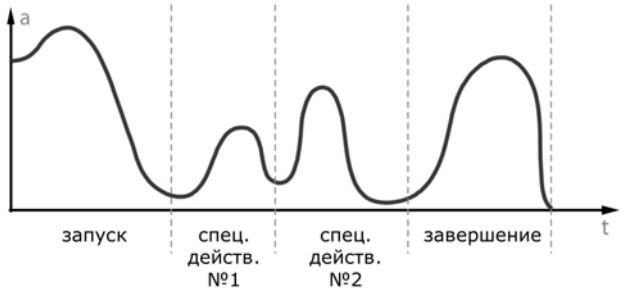


Рис.4. Выделение фаз работы на основе анализа активности ПО

Таким образом, модель поведения ПО будет состоять из нескольких нейронных сетей – как минимум двух (на запуск и завершение работы). Проверка реальной работы ПО относительно модели проверяется следующим образом. Во время работы ПО модуль анализа динамически разделяет данные о работе на фазы, создаёт вектор, описывающий каждую фазу, и подаёт этот вектор на вход соответствующей нейронной сети. Если результат выхода нейронной сети будет ниже некоторого заданного при обучении порогового значения (например, 0,9), можно говорить о том, что в данной фазе поведения ПО возникла аномалия.

В случае необходимости с использованием логики ИЛИ/И можно установить гибкий или жёсткий порядок выполнения специфических действий. В таком слу-

чае любое отклонение от их порядка выполнения будет также считаться аномалией поведения.

На практике предложенная методология позволяет выявлять следующие типы вирусов и вредоносные действия: классические вирусы, трояны, макровирусы, НДВ и подмена ПО. В частности, предложенный комплекс обучения и анализа был опробован на таких приложениях, как MS Word, MS Excel, Acrobat Reader (версия 8.1.2), Internet Explorer, а также ряде специально написанного тестового ПО.

В случае с MS Word тестирование проводилось на нескольких десятках макровирусов, все из которых были обнаружены системой анализа на разных этапах работы. Самые известные из этих вирусов включают в себя: Fries.a, Over.a, Want,Nail.a, Antiavs и прочие, совершающие изменения файловой системы и реестра. В случае с Acrobat Reader было выявлено различное поведение приложения на компьютерах с интернет-соединением и без (связанное с автоматическими обновлениями), а также НДВ, связанные с выполнением специально сформированного PDF-файла, который вызвал переполнение буфера и давал возможность выполнить произвольный код на целевой системе.

Дополнительным преимуществом предложенной системы является то, что, составив модель поведения ПО на одном компьютере, её можно использовать для этого же ПО, установленного на других компьютерах с аналогичной операционной системой. Выявленные аномалии в таком случае будут также свидетельствовать об особенностях поведения на компьютерах с другой конфигурацией, то есть о возможных НДВ. Кроме того, разработанный программный комплекс не зависит от операционной системы, под управлением которой работает моделируемое ПО, что позволяет с помощью одного комплекса обучения построить модели поведения приложений как Microsoft Windows, так и Linux, и Solaris и прочих.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Ясницкий Л.Н.* Введение в искусственный интеллект. – М.: Издательский центр "Академия", 2005.

2. *Swingler K.* "Applying Neural Networks. A practical Guide", Morgan Kaufmann; Pap/Dsk edition, 1996.

3. *Drucker H., Le Cun Y.* Improving Generalization Performance Using Backpropagation, IEEE Transactions on Neural Networks, Vol.3, N5, 1992.

4. *Malki H.A., Moghaddamjoo A.* Using the Karhunen-Loe`ve Transformation in the Back-Propagation Training Algorithm, IEEE Transactions on Neural Networks, Vol.2, N1, 1991.

Карайчев Г.В.
Научный руководитель: к. ф.-м. н., доцент Нестеренко В.А.
г. Ростов-на-Дону, ЮФУ

Лозунг: Divide et impera

ВЫЯВЛЕНИЕ АНОМАЛЬНОЙ АКТИВНОСТИ МЕТОДОМ АДАПТИВНЫХ СЕТОК

В данной статье рассматривается метод выявления аномальной активности на основе адаптивного построения профиля системы в процессе самообучения высокой производительности. Первичные данные о соединениях преобразуются методом главных компонент и полученные векторы кластеризуется методом адаптивных сеток. Результаты представленной работы (на основе данных

из 1999 KDD CUP) показывают, что эффективность предлагаемого подхода достигает 91% обнаружения вторжений уже при 5% ложных тревог при сравнительно высокой «зашумленности» аномалиями обучающего трафика.

Сетевая безопасность, аномальный анализ, метод главных компонент, кластеризация, адаптивная сетка.

Существующие технологии выявления аномальной активности подразделяются на два типа: на основе управляемого построения профиля (*supervised*) и на основе допустимого построения профиля (*unsupervised*). При первом подходе эталонное состояние системы строится на основе свободного от атак трафика. Его недостаток выражается в том, что требуется дополнительная работа по фильтрации и очистке исходного набора данных от аномалий. Второй подход не требует очистки эталонных данных, используемых для обучения системы. Выявление аномалий на основе допустимого построения профиля требует выполнения следующих условий: базовый профиль основывается на наборе данных, в котором количество нормального трафика значительно превышает количество аномального; набор данных, содержащий атаку, статистически отличается от нормального набора.

В предлагаемой работе для выявления аномальной активности предлагается использование подхода на основе допустимого построения базового профиля с использованием метода кластеризации данных. В настоящее время существует множество методов кластеризации, в данной работе используются решеточные методы (*grid-based*), разделяющих пространство объектов на конечное число клеток, формируя решеточную структуру. Их преимуществом над другими методами является высокая производительность.

Пусть $X_1,...,X_N$ – некоторая последовательность случайных событий с соответствующими числовыми характеристиками. В общем случае, пространство характеристик является многомерным, его размерность зависит от особенностей исследуемого процесса и определяется спецификой решаемой задачи. Применив метод кластеризации к множеству событий в системе, разобьем его на подмножества так, чтобы каждый кластер состоял из похожих объектов, а элементы различных кластеров имели существенные различия (внутренняя однородность и внешняя изолированность). В результате нормальные и аномальные события станут, различимы, т.е. будут попадать в разные кластеры.

Мы будем использовать двумерное пространство характеристик. Для случая многомерного пространства справедливы аналогичные рассуждения. Введем в выбранном пространстве сетку размера $n_1 \times n_2$: $A = \{a_{ij}\}$, a_{ij} – количество событий в клетке (*i, j*). Пусть K_1 и K_2 координаты левого нижнего и правого верхнего углов сетки, соответственно: $K_1 = (k_{11}, k_{12})$, $K_2 = (k_{21}, k_{22})$, а векторы $B_1 = (b_{1,1},...,b_{1,(n_1+1)})$, $B_2 = (b_{2,1},...,b_{2,(n_2+1)})$, содержат отношение ширины данной границы к общей ширине сетки (рис. 1). Таким образом, тройка (*A*, *K*, *B*) полностью определяет нашу сетку.

При построении адаптивной сетки начнем с построения сетки с равномерным шагом $b_{ij} = \dfrac{j-1}{n_i}, j = \overline{1, n_i + 1},\ i = 1,2$.

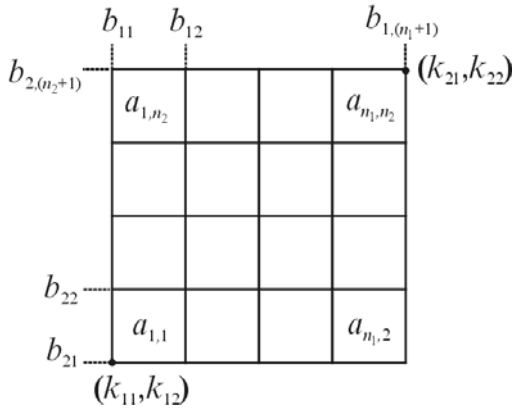


Рис. 1. Сетка (A, K, B)

Пусть $X = (x_1, x_2)$ – очередное событие. Рассмотрим действие функции, отображающей множество координат точек из пространства событий во множество номеров клеток сетки. Номер ячейки (*s*, *t*) вычисляется по формулам:

$$\begin{aligned} (k_{2,1} - k_{1,1}) \cdot b_{1,s} + k_{1,1} < x_1 \le (k_{2,1} - k_{1,1}) \cdot b_{1,s+1} + k_{1,1},\ s = \overline{1,n_1}, \\ (k_{2,2} - k_{1,2}) \cdot b_{2,t} + k_{1,2} < x_2 \le (k_{2,2} - k_{1,2}) \cdot b_{2,t+1} + k_{1,2},\ t = \overline{1,n_2}, \end{aligned} \quad (1)$$

при этом увеличивается число точек в соответствующей ячейке на единицу.

В том случае, если событие *X* попадает за пределы области сетки, вычисляются новые координаты сетки.

Недостаток построенной модели в том, что все клетки имеют один и тот же размер, не обеспечивая эффективной кластеризации. Изменим размеры ячеек сетки таким образом, чтобы количество событий в разных клетках было примерно одинаковым. Перестраивая сетку, мы предполагаем, что события внутри каждой ячейки распределены с равномерной плотностью. Подобное предположение позволяет существенно ускорить процесс адаптации, повышая производительность всего подхода в целом.

Для реализации процесса адаптации сетки введем два множества $R_1 = (r_{11},...,r_{1n_1}), R_2 = (r_{21},...,r_{2n_2})$, где r_{ij} – сумма событий в *j*-м ряду клеток *i*-й размерности:

$$r_{1j} = \sum_{k=1}^{n_1} a_{kj},\ r_{2j} = \sum_{k=1}^{n_2} a_{jk}.$$

Тогда среднее число событий в ряду клеток одной размерности:

$$\bar{r}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} r_{ij},\ i = 1,2.$$

Введем функцию плотности распределения событий в ряду по одной размерности:

$$\rho_{ij} = \frac{r_{ij}}{b_{i,j+1} - b_{i,j}},\ j = \overline{1, n_i},\ i = 1,2\ . \qquad (2)$$

Проведя адаптацию по всем размерностям, получим новую сетку, в которой количество событий в каждой клетке примерно одинаково.

На рис. 2 представлен результат построения адаптивной сетки, совмещенной с пространством случайных событий. Темным цветом отмечены клетки с высокой плотностью событий, светлым – с низкой. Кружками отмечены отдельные события.

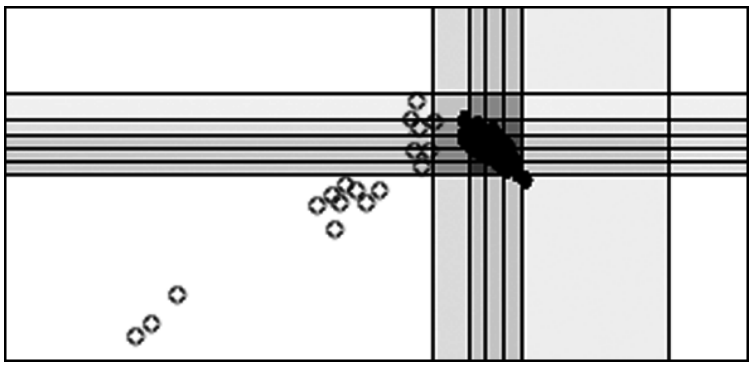


Рис. 2. Результат построения адаптивной сетки: черные кружки – нормальные события, белые кружки – атаки

Размерность пространства характеристик может быть много больше двух [8] и предлагаемый метод легко может быть обобщен на многомерный случай.

Использование метода главных компонент позволяет существенно уменьшить размерность пространства характеристик и выбрать такие компоненты, которые играют решающую роль при выявлении аномальных событий. Следуя методу главных компонент, рассмотрим корреляционную матрицу M размера h, вычисленную по N случайным событиям $X = (X_1,...,X_N)$, где $X_k = (x_{k1},...,x_{kh})$:

$$M = \left\{ \sum_{k=1}^{N} v_{ki} \cdot v_{kj} \right\},\ v_{ki} = \frac{x_{ki} - \overline{x_i}}{\sigma_i},$$

где, $\overline{x_i}$ – математическое ожидание i-й характеристики, σ_i – ее дисперсия.

Для пар собственных значений и собственных векторов матрицы M $(\lambda_1, e_1), (\lambda_2, e_2), ..., (\lambda_h, e_h)$ выполняется свойство $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq ... \geq \lambda_h \geq 0$. А i-й выборочный главный компонент для k-го события будет иметь следующий вид:

$$w_i = e_{i1} v_{k1} + e_{i2} v_{k2} + ... + e_{ih} v_{kh},$$

где e_{ij} – собственный вектор.

Заметим, что w_i имеет выборочную дисперсию λ_i и выполняется свойство $\lambda_1 + \lambda_2 + \lambda_3 + ... + \lambda_h = h$. Это означает, что все случайные величины исходной выборки учтены в главных компонентах.

Главные компоненты, соответствующие большим собственным значениям, в основном, связаны с регулярными зависимостями характеристик друг от друга; компоненты, соответствующие малым собственным значениям, связаны со случайными несущественными отклонениями. Поэтому наиболее эффективно использование компонент, соответствующих середине интервала собственных значений [4].

Из приведенного рис. 2 видно, что атаки попадают в области с низкой плотностью, а большинство нормальных событий – в области с высокой плотностью. Поэтому для разделения нормальных и аномальных событий введем пороговое значение ρ_0 плотности клетки в пространстве характеристик. Тогда ячейки, величина плотности (2) которых выше ρ_0 ($\rho_{st} \geq \rho_0$), считаются нормальными, остальные ($\rho_{st} < \rho_0$) признаются аномальными. И если очередное событие по (1) помещается в такую ячейку, оно классифицируется как потенциальное вторжение.

Для определения эффективности выдвигаемого метода обнаружения вторжений был использован набор данных 1999 KDD CUP (V Международная конференция «Knowledge Discovery and Data Mining») от «Lincoln Labs». Так, в рамках проекта «KDD'99», была создана инфраструктура, условно имитирующая ЛВС ВВС США, и в течение 9 недель записывалась информация о сетевых соединениях в базу данных (более 5 миллионов записей о соединениях). Каждое соединение отмечалось либо как нормальное, либо как содержащее атаку конкретного типа. А для каждого соединения вычислялось порядка 30 различных характеристик. В итоговом наборе данных содержатся атаки 4-х категорий: DOS (отказ в обслуживании); R2l (несанкционированный доступ с удаленного компьютера); U2R (несанкционированный доступ к правам администратора); прослушивающие (зондирование и другие способы сбора информации).

В связи с тем, что в 1999 KDD CUP не используется такая характеристика, как время соединения, в предлагаемой работе «период» обучения выбирается по числу событий, а векторы метода главных компонент выбираются по принципу наилучшей ортогональной проекции исходных данных.

Для оценки эффективности предлагаемого подхода обнаружения аномалий, используем метод ROC-характеристик (*Receiver operating characteristic*) [9]. Основой этого метода оценок является анализ графика ROC-кривой, при построении которого используется относительная частота правильного (TPR) и ложного (FPR) срабатываний системы. Формулы для вычисления кривой ROC приведены ниже:

$$TPR = \frac{TP}{TP + FN},\ FPR = \frac{FP}{FP + TN}, \tag{3}$$

где TP – число выявленных аномалий, FN – число пропущенных аномалий, FP – число ложных срабатываний, TN – число верно идентифицированных событий как нормальные.

Графики поведения ROC-кривых для различных типов атак и методов обнаружения вторжения приведены на рис. 3. Чем площадь фигуры под ROC-кривой больше, тем алгоритм эффективней.

При построении ROC-кривой, в работе в качестве варьируемого параметра используется коэффициент, задающий величину плотности клетки:

$$\rho_0 = k \cdot \max\left(a_{ij}\right),$$

где $\max(a_{ij})$ – показатель плотности ячейки с максимальной величиной, k – коэффициент плотности распределения событий клетки.

Исследование предлагаемого в работе метода проводится для заранее подготовленного набора данных, содержащего аномалии произвольных типов, затрагивающие протокол передачи данных *http*. Для построения профиля системы по этому набору данных было использовано 10 характеристик соединений, включая характеристики соединений, рассчитанные по информации о доменах и вычисленные с использованием двухсекундного окна. Результаты вычисления по формуле (3), в зависимости от коэффициента плотности распределения событий в клетке, представлены в табл. 1.

Таблица 1

Результаты вычисления (3) для набора данных

k	0,0002	0,001	0,003	0,01
TPR	63,1840%	77,6119%	94,5273%	97,5124%
FPR	0,8899%	2,0975%	7,5117%	16,8097%

Теперь сравним эффективность выявления аномальной активности методом адаптивных сеток с другими методами обнаружения вторжений. Результаты сравнения представлены в табл. 2. На рис. 3 изображены кривые, иллюстрирующие эффективность обнаружения атак предлагаемым методом в сравнении с методами, разработанными победителями KDD CUP'99 [5], [6], [7].

Таблица 2

Эффективность выявления аномалий различных методов выявления вторжений

TPR FPR	Адаптивная сетка	Canberra	NN	KDD	LOF
1%	64,51%	4,12%	58,25%	0,6%	0,03%
4%	83,56%	6,13%	81,3%	73,74%	98,7%
8%	94,67%	26,2%	92,78%	87,12%	99,04%
10%	95,32%	28,11%	93,96%	88,99%	99,13%

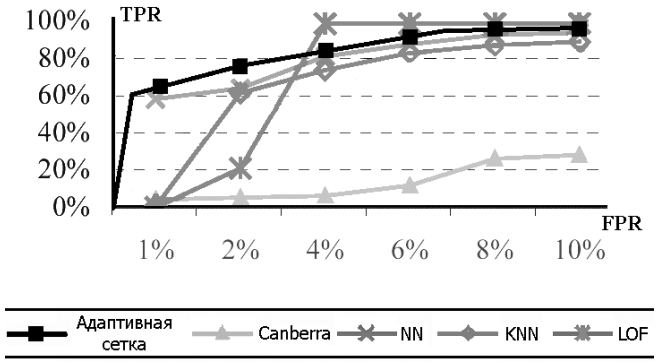


Рис. 3. График ROC-кривых для различных методов выявления вторжений

В заключение данной статьи следует отметить, что предлагаемый метод адаптивных сеток устойчиво работает даже при достаточно зашумленном аномалиями

исходном обучающем трафике, давая высокий показатель обнаружения атак уже при небольшой величине ложных срабатываний. При этом метод имеет высокую производительность и возможность настройки многочисленных параметров при построении сетки.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Карайчев Г.В., Нестеренко В.А.* Применение весовых функций для определения локальных статистических характеристик потока пакетов в сети / Известия высших учебных заведений. Северо-Кавказский регион. Естественные науки. – Ростов н/Д, 2008. – № 1. – С. 10–14.

2. *M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang,* A novel anomaly detection scheme based on principal component classifier. // Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, Melbourne, FL, USA, 2003, pp. 172-179.

3. *Wei Wang, Roberto Battiti.* Identifying Intrusions in Computer Networks with Principal Component Analysis. // Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06), p. 270-279, April 20-22, 2006.

4. *Wei-keng Liao, Ying Liu, Alok Choudhary*. A Grid-based Clustering Algorithm using Adaptive Mesh Refinement. // 7th Workshop on Mining Scientific and Engineering Datasets in conjunction with SIAM International Conference on Data Mining (SDM), pp. 61-69, April 2004, Lake Buena Vista, Florida, USA.

5. *I. Levin.* KDD-99 Classifier Learning Contest: LLSoft's Results Overview. ACM SIGKDD Explorations2000, pp. 67-75, January 2000.

6. *B. Pfahringer.* Winning the KDD99 Classification Cup: Bagged Boosting. ACM SIGKDD Explorations 2000, pp. 65-66, January 2000.

7. *M. Vladimir, V. Alexei, and S. Ivan.* The MP13 Approach to the KDD'99 Classifier Learning Contes». ACM SIGKDD Explorations 2000, pp. 76-77, January 2000.

8. *Lincoln labs*. KDDCup'99. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2003.

9. *Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, Boris Skoric.* Measuring Intrusion Detection Capability: An Information-Theoretic Approach. ASIACCS'06, March 21-24, 2006 Taipei, Taiwan.

Минаков С.В.
Научный руководитель: д.т.н., доцент Финько О.А.
г. Краснодар, КВВУ

Лозунг: Достоверность текстов

ЗАЩИТА ТЕКСТОВЫХ ДАННЫХ ОТ ОШИБОК ПУТЕМ ИЗБИРАТЕЛЬНОГО ИЗБЫТОЧНОГО КОДИРОВАНИЯ

Рассматриваются пути повышения достоверности и информационной живучести текстовых данных на основе использования гибридной двухступенчатой семантико-кодовой избыточности применительно к семантическим единицам текста наиболее уязвимых к смысловым ошибкам.

Достоверность текста, избыточное кодирование, расстояние Левенштейна, семантическая единица.

Текстовый тип данных в настоящее время и в ближайшем будущем будет составлять основу документированной информации. Для обеспечения безопасности электронного документооборота в настоящее время успешно применяется электронная цифровая подпись (ЭЦП). Однако существенным недостатком ЭЦП является то, что ЭЦП гарантирует целостность и достоверность файла, а не документа,

игнорируя, тем самым, возможные видоизменения текстового документа, не влияющие на смысловое содержание документа. Так, лишний знак пробела, исправляемая орфографическая ошибка в тексте документа приведут к полному запрету на его использование. В тоже время определенный интерес могла иметь возможность восстановления семантического содержания документа при появлении в тексте неисправляемых ошибок, например ошибки в цифровых последовательностях, трансформации разрешенных слов естественного языка в другие разрешенные слова. В сочетании с ЭЦП такая возможность позволила бы восстановить истинное содержание документа без повторных запросов исходного документа. Такие ситуации возникают в случае невосполнимой утраты подлинника или его части, неприемлемо больших (убыточных) временных затрат, связанных с повторными запросами и пр.

Неравномерность избыточности текста

Текст неравномерен по смысловой нагрузке, следовательно, требования к повышению достоверности семантических единиц (СЕ) текста, влияющих на смысл, должны быть разными. Целесообразно повысить достоверность СЕ вероятность искажения, которых максимально влияет на смысл текста, и применять к ним методы повышения достоверности.

Одним из известных методов повышения достоверности текстовых данных является внесение семантической и лингвистической избыточностей (повторы текстовых единиц, применение синонимов, перефразировки, повторы цифровых данных словами и т.д.). Другим путем повышения достоверности текстовых данных является помехоустойчивое кодирование [1], достоинством которого является возможность обнаруживать или исправлять ошибки на выходе конечных устройств. Недостаток – кодирование не учитывает смысл (ценность) передаваемой информации.

В основе предлагаемого решения повышения достоверности СЕ лежит автоматическое определение СЕ, минимальное искажение которых может привести к трансформации в другое существующее (семантически близкое) слово текста на каком-либо естественном языке. Причем предлагается СЕ сравнивать не с множеством СЕ, составляющих текст, а с множествами образованными онтологическими рядами относительно анализируемой СЕ. Для выявления семантически близких СЕ применим метод динамического программирования [2].

Пусть имеется множество A, соответствующее вводимому тексту, при этом элементами $a_i \in A$ являются СЕ текста. Необходимо для слова $a_i \in A$ найти множество всех слов $b_j \in B$, до которых расстояние Левенштейна равно максимально допустимому числу d. При расчете расстояния Левенштейна вводится расширение алфавита специальным пустым символом ε. Введение пустого символа позволяет свести операции удаления и добавления к операции замены. $d_{i,j} = d\left(a\left(1,i\right), b\left(1,j\right)\right)$ – расстояние между префиксами строк a и b, длины которых равны соответственно i и j. Элементами b_j являются слова, содержащиеся в базе данных множества B. Эти слова объединяются путем разбиения единой базы данных по тематическим признакам. Для расстояния Левенштейна набор элементарных операций состоит из операций замены, вставки, удаления одной буквы. При выполнении нижеследующих условий d является расстоянием Левенштейна:

$$w(a,\varepsilon)=1,$$
$$w(\varepsilon,b)=1,$$
$$w(a,b)=1, \text{ если } a\neq b,$$
$$w(a,b)=0, \text{ если } a=b,$$

где $w(a_i,b_j)$ - цена преобразования символа a_i в символ b_j. В процессе вычислений значения $d_{i,j}$ записываются в массив $(m+1)(n+1)$, вычисляются с помощью следующего рекуррентного соотношения:

$$d_{i,j}=\min\left\{d_{i-1,j}+w(a_i,\varepsilon), d_{i,j-1}+w(\varepsilon,b_j), d_{i-1,j-1}+w(a_i,b_j)\right\}.$$

Избирательное избыточное кодирование семантических единиц текста

Для выбора СЕ текста (рис. 1) возьмем $d=1$. Данному условию удовлетворяют СЕ, выделенные рамкой (все цифры; название месяцев «июль/июнь»; фамилии «Рыбалко/ Рыбалка/ Рыбалков», «И.Сталин/ В.Сталин/ И.Салин», название населенных пунктов «Моховое/ Меховое/ Мохово»; «р.Ока/ р.Оса/ р.Ака» и т.п.).

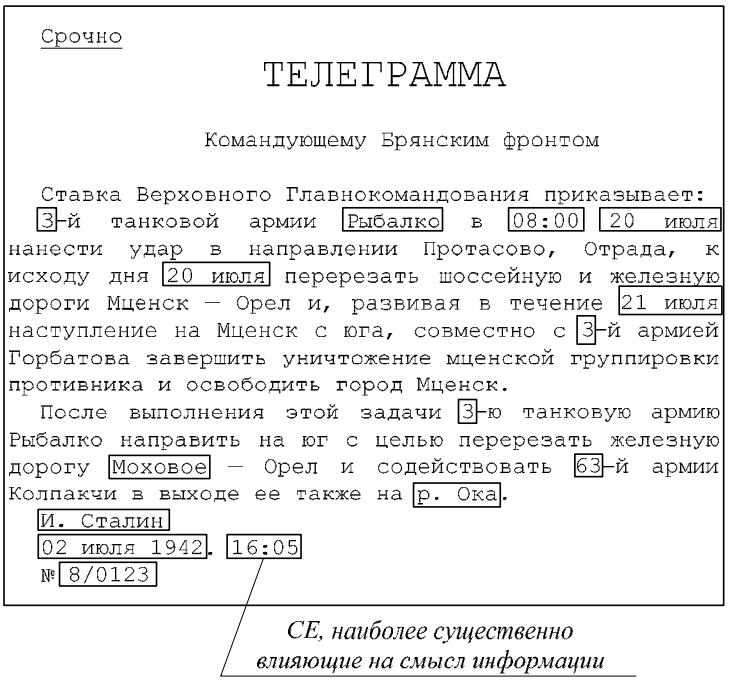


Рис. 1. Пример выделения СЕ в тексте телеграммы

Присвоим выделенным СЕ текста (рис. 1) кодовые обозначения (табл. 1).

Таблица 1

Пример кодирования алфавита

А - 0	М - 12	Ш - 24	! - 33	0 - 48	пробел - 32
Б - 1	Н - 13	Щ - 25	? - 34	1 - 49	
В - 2	О - 14	Ъ - 26	№ - 36	2 - 50	
Г - 3	П - 15	Ы - 27	¶ - 37	3 - 51	
Д - 4	Р - 16	Ь - 28	(- 40	4 - 52	
Е - 5	С - 17	Э - 29	) - 41	5 - 53	
Ж - 6	Т - 18	Ю - 30	, - 44	6 - 54	
З - 7	У - 19	Я - 31	– - 45	7 - 55	
И - 8	Ф - 20		. - 46	8 - 56	
Й - 9	Х - 21		/ - 47	9 - 57	
К - 10	Ц - 22		: - 58		
Л - 11	Ч - 23		; - 59		

Вычислим для каждой полученной таким образом числовой последовательности проверочное число. Суммируем каждую последовательность по модулю 64:

$$M = \left(\sum_{i=1}^{n} a_i \right) \bmod 64,$$

где n – количество символов в СЕ.

Полученные проверочные числа добавим к исходным выделенным СЕ. Для отличия их от текста введем разделитель – ## (редко встречающийся набор символов) и двухзначные проверочные цифры, соответствующие числу M избыточного кода, скрытые от пользователя (рис. 2).

Принимающая сторона вычисляет для соответствующих СЕ проверочные символы и сравнивает их значение с прикрепленными. При несовпадении результатов адресат делает вывод об искажении данной СЕ текста.

Элементами b_j единой базы данных B являются семантические единицы передаваемых сообщений. Пример таких СЕ представлен в табл. 2.

Таблица 2

Пример базы онтологических рядов для анализируемого текста

Тематические признаки	Элементы множества B
Годы (актуальный ряд)	…, 1939, 1940, 1941, 1942, 1943, 1944, 1945 …
Месяцы года	январь, февраль, март, апрель, май, июнь, июль, август, сентябрь, октябрь, ноябрь, декабрь
Даты	01, 02, 03, 04, 05, … , 29, 30, 31
Время	00:00, 00:01, … 01:00, 01:01, 01:02, … , 23:58, 23:59

Продолжение табл. 2

Тематические признаки	Элементы множества *B*
Военачальники ВОВ (маршалы) (актуальный ряд)	Василевский, Говоров, Жуков, Конев, Малиновский, Мерецков, Рокоссовский, Сталин, Тимошенко, Толбухин
Военачальники ВОВ (генералы арий) (актуальный ряд)	Антонов, Апанасенко, Василевский, Еременко, Жуков, Конев, Малиновский, Мерецков, Павлов, Попов, Рокоссовский, Соколовский, Тюленев, …
Орган управления	отделение, взвод, …, дивизия, корпус, армия, …, Генеральный Штаб.
Виды и рода ВС	авиация, ВВС, сухопутные, …, морские, ВМФ, связь
Воинские звания	рядовой, ефрейтор, младший сержант, …, генерал-армии, маршал
Вооружение (танки) (актуальный ряд)	Т-34, Т-34-57, ОТ-34, ТО-34, КВ-1с, КВ-85, ИС-1, ИС-3, ИСУ-152, ИСУ-122, ИСУ-122С, …
…	…
Географические наименования (реки) (актуальный ряд)	Большая Чернава, Быстрая Сосна, Кшень, Нерусса, Общерица, Ока, Олым, Орлик, Семенек
Географические наименования (города) (актуальный ряд)	…, Мценск, Мымрино, Мыцкое, Навесное, Навля, …, Шатилово, Шахово, Шашкино, Щербово, Юшково, Яковлево, …
…	…

Срочно

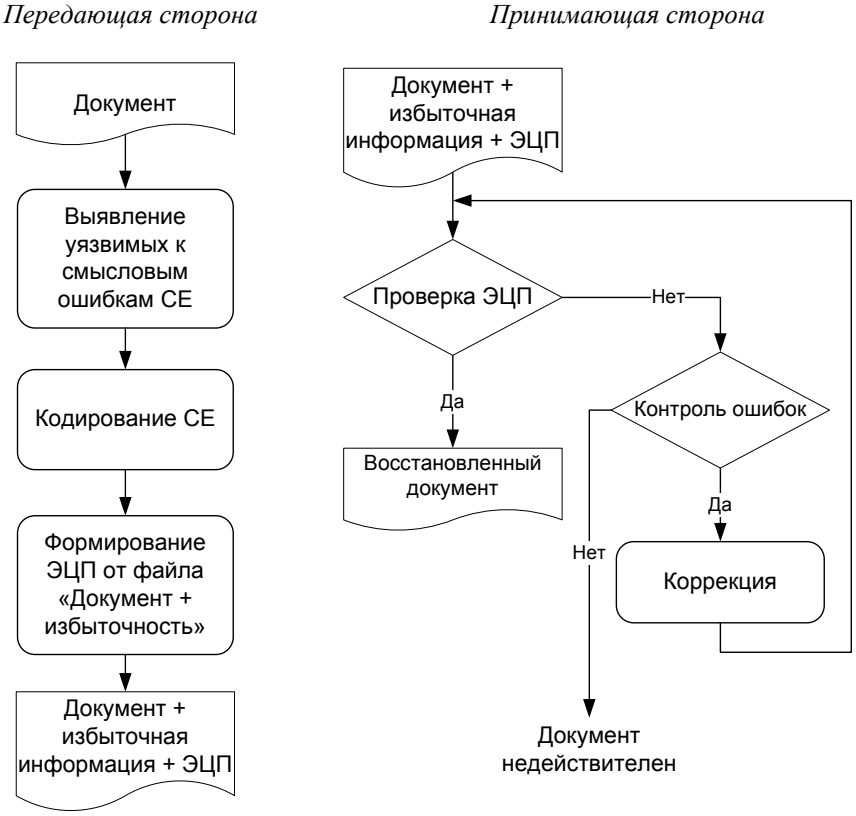
ТЕЛЕГРАММА

Командующему Брянским фронтом

Ставка Верховного Главнокомандования приказывает:
3##51-й танковой армии Рыбалко##15 в 08:00##18 20 июля##18 нанести удар в направлении Протасово##46, Отрада##52, к исходу дня 20 июля##18 перерезать шоссейную и железную дороги Мценск##15 – Орел##46 и, развивая в течение 21 июля##19 наступление на Мценск##15 с юга, совместно с 3##51-й армией Горбатова##04 завершить уничтожение мценской##38 группировки противника и освободить город Мценск##15.
После выполнения этой задачи 3##51-ю танковую армию Рыбалко##15 направить на юг с целью перерезать железную дорогу Моховое##18 – Орел##46 и содействовать 63##41-й армии Колпакчи##27
в выходе ее также на р. Ока##54.
И. Сталин##25
02 июля 1942##02. 16:05##06
№ 8/0123##45

Рис. 2. Текст документа с введенной кодовой избыточностью

На заключительном этапе ЭЦП формируется от всего документа вместе с введенной избыточной информацией. При отрицательном результате проверки подлинности ЭЦП оператор приемной стороны добивается положительного результата проверки путем коррекции СЕ текста (рис. 3).



а б

Рис. 3. Последовательности операций при формировании сообщения (а) и при получении сообщения (б)

Применение гибридной семантической и кодовой избыточности в сочетании со средствами ЭЦП позволит обеспечить не только заданный уровень достоверности, но и информационную живучесть электронного документооборота, недоступную для существующих методов. В дальнейшем предполагается совершенствование рассмотренного в докладе метода путем введения многоступенчатой кодовой избыточности как функции от показателя значимости и ошибкоустойчивости к смысловому содержанию СЕ текста.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Питерсон У.* Коды, исправляющие ошибки/ *У. Питерсон, Э. Уэлдон.* – М.: Мир, 1976. – С. 596.

2. *Graham, Stephen.* String Search/ *Stephen A. Graham.* – UK. School of Electronic Engineering Science University College of North Wales, 1992. – P. 103.

Дорда А.В., Стародубцев С.П.
Научный руководитель: к.т.н., доцент Чипига А.Ф.
г. Ставрополь, СГТУ

Лозунг: Облачные вычисления: новые возможности и новые угрозы

ИССЛЕДОВАНИЕ ОБРАБОТКИ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ С ЦЕЛЬЮ ОПРЕДЕЛЕНИЯ УГРОЗ БЕЗОПАСНОСТИ ИНФОРМАЦИИ

В работе рассмотрены достоинства облачной обработки данных и сделана попытка выявить угрозы безопасности информации, возникающие при использовании вычислительных облаков.

Облачные вычисления, вычислительное облако, безопасность, угрозы, киберпреступность, достоинства облачных вычислений, облачная обработка данных.

Облачная обработка данных (Cloud Computing) – это парадигма, в рамках которой информация постоянно хранится на серверах в сети Интернет и временно кэшируется на клиентской стороне [1]. Метафорически облаком называют Интернет, который скрывает все технические детали.

Концепция вычислительного облака дает возможность масштабировать физические ресурсы (например, процессор и дисковое пространство) и предоставлять к ним доступ через Интернет; при этом обработка и хранение рассматриваются как сервис. Благодаря развитию технологий виртуализации стало возможно еще более эффективно организовывать доступ к общим ресурсам.

Принцип облачных вычислений предполагает, что пользователь получает доступ к мощным вычислительным и дисковым ресурсам, совершенно не задумываясь о расположении и настройке этих ресурсов. Концепция вычислительного облака соприкасается и соединяет в себе множество технологий: кластерные и grid-вычисления, utility computing, SOA, Web 2.0, SaaS, PaaS, IaaS, dSaaS и т. д.

Вычислительные облака являются перспективным направлением развития информационных технологий. Облачные вычисления, с одной стороны, очень удобны для пользователей и обладают рядом преимуществ в организации вычислительного процесса, а с другой стороны, приводят к появлению специфичных угроз безопасности информации, обрабатываемой посредством вычислительных облаков.

Основным достоинством облачных вычислений является гибкий и недорогой аутсорсинг IT-инфраструктуры (технология IaaS). Организации не нужно вкладывать средства в создание и развитие IT-инфраструктуры, вместо этого можно обратиться к провайдеру вычислительного облака и он предоставит все необходимое. Нагрузка между компьютерами, входящими в вычислительное облако, распределяется автоматически.

Еще одним достоинством вычислительного облака является технология SaaS, предполагающая использование ПО как предоставляемую услугу. Для работы с программами не нужно устанавливать их на рабочую станцию, заботиться об их совместимости и управлять лицензиями. Все ПО представлено в виде веб-сервисов и веб-приложений и все, что нужно для их функционирования, – браузер и доступ к сети Интернет [2].

Облачные вычисления позволяют небольшим организациям обеспечить функционирование бизнес-процессов, не затрачивая на это больших средств. Сле-

довательно, эта концепция уже востребована и популярность ее растет. В больших организациях облачная идеология позволяет оптимизировать виртуализацию и обеспечить экономию (за счет упрощения управления системой, за счет более эффективного использования аппаратной части) на поддержку IT- средств до 20-30 процентов, а это весомый аргумент. Создание частных вычислительных облаков позволит крупным компаниям получить выгоду не только от повышения эффективности IT-инфраструктуры, но и от использования свободных облачных ресурсов другими организациями.

Во-первых, существует угроза нарушения конфиденциальности и целостности пользовательской информации. Так как облачная концепция предполагает хранение данных в самом облаке, т.е. вне организации, то существенно возрастает риск доступа к этой информации третьих лиц с целью ознакомления с ней и (или) изменения ее содержания. Кроме того, чем больше облако, тем больше связей между ним и остальной глобальной сетью Интернет и тем больше имеется у нарушителей способов добиться своих целей.

Во-вторых, существует угроза нарушения доступности пользовательской информации и услуг, предоставляемых потребителю вычислительным облаком. Так как само облако является частью сети Интернет, то довольно сложно блокировать или нарушить его работоспособность, но оставить организацию без IT-инфраструктуры, предоставленной облаком, и, как следствие, нарушить функционирование информационных процессов в фирме, тем самым нанеся ей большой ущерб, вполне вероятно.

Самой опасной угрозой является проблема возможного использования облачных ресурсов в противоправных целях. Причем проблема актуальна как при легальном доступе к облаку, так и при несанкционированном, посредством компрометации пользователя или веб-приложений с последующим захватом контроля над частью или целым вычислительным облаком.

Захваченные киберпреступником вычислительные ресурсы могут использоваться им для решения ресурсоемких задач, таких как подбор паролей или ключей шифрования используемых на каком-либо канале связи, интересном злоумышленнику. Также посредством подконтрольных вычислительных ресурсов злоумышленник может наращивать свою вычислительную мощь, захватывая новые облака.

Кроме того, взломщики могут объединить свои усилия и создать собственное вычислительное облако, благо уже сейчас существует множество решений с открытым исходным кодом. Тогда вполне возможно, что преступники будут предлагать взлом как услугу в рамках концепции облачных вычислений. Уже сейчас нарушители предлагают услуги, называемые «Мошенничество 2.0» [3], такие, как организация фишинг-атак, аренду так называемых «ботнетов» – компьютерных сетей, зараженных вредоносным ПО, позволяющим нарушителям их контролировать. Если спрос на такие услуги есть сейчас, то можно не сомневаться, что он останется и в будущем, и найдется много желающих воспользоваться «облачным взломом», который, согласно концепции, должен быть прозрачен для потребителя этой услуги.

Таким образом, можно сделать вывод, что концепция облачных вычислений слишком нова, чтобы решить все или некоторые существующие проблемы безопасности. Не стоит сразу отвергать вычислительные облака, как заведомо уязвимые – ведь и автономные вычислительные машины, информационно-вычислительные сети и сеть Интернет имеют изъяны в обеспечении безопасности данных, хоть и существуют уже давно, не один год.

Ажиотаж вокруг облачных вычислений создает благодатную почву для внедрения облачных вычислений в современную технологическую среду. Естествен-

но, что в начальных решениях недоработан функционал, что в веб-приложениях на основе новой технологии обнаруживают уязвимости. В настоящее время происходит процесс отладки технологии. Вычислительные облака также станут неотъемлемой технологией в нашей жизни, как только они будут соответствовать требованиям функциональности, эффективности и безопасности, как и множество уже привычных нам технологий.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Carl Hewitt,* «ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing», IEEE Internet Computing, vol. 12, no. 5, pp. 96-99, Sep./Oct. 2008, doi:10.1109/MIC.2008.107

2. *M. Tim Jones* (2008.12.25). Cloud Computing и Linux. Получено с веб-узла IBM: http://www.ibm.com/developerworks/ru/library/l-cloud-computing/

3. Взломщики уходят в облака. (30.10.2008). Получено с веб-узла. Информационная безопасность: http://www.itsec.ru/newstext.php?news_id=51252

Саркисов Т.Г., Мирзаян А.В.
Научный руководитель: д.ф.-м.н. Осипян В.О.
г. Краснодар, КГУ

Лозунг: Плотный ранец - залог эффективной криптосистемы

ОБЗОР МЕТОДОВ ОПРЕДЕЛЕНИЯ ПЛОТНОСТИ РАНЦЕВЫХ ВЕКТОРОВ

В работе приводятся три различных подхода для уточнения понятия плотности ранцевых векторов и изучаются их свойства в зависимости от выбранного подхода.

Криптография, стандартный ранцевый вектор, сверхрастущий вектор, мощность ранца, плотность вектора, синтез ранцевых криптосистем.

Рассмотрим классическую задачу об укладке ранца: имеется набор из n предметов с заданными объёмами a_i, $i = 1..n$, а также ранца объемом V, и необходимо найти такой набор предметов, чтобы полностью заполнить ранец.

Математическая модель этой задачи выглядит следующим образом:

$$A = (a_1, a_2, \ldots, a_n) \quad (1)$$

– ранцевый вектор размерности n, n ≥ 3 из n различных натуральных компонентов a_I, $I = 1. . n$ и (A,V) – вход задачи о ранце, где V также некоторое натуральное число. Один из методов определения плотности ранца заключается в использовании формулы[2,3]:

$$d(A) = \frac{n}{\log_2 \max A}. \quad (2)$$

Как видно из такого определения плотности ранца A, она может принимать сколь угодно большое значение, например, для следующего ранца:

$$A = (1,2,3,...,128). \quad (3)$$

$$d(A)=\frac{128}{7}=18.2857.$$

В частности, для сверхрастущего ранцевого вектора (1) имеет место

Теорема. Для любого сверхрастущего вектора А справедливо неравенство

$$d(A)\le n/(n-1).$$

Прежде чем доказать это неравенство, предварительно докажем лемму:

Лемма. Для любого сверхрастущего вектора А имеет место неравенство

$$a_i \ge 2^{i-1}, \text{ где } i \ge 1.$$

Доказательство. Любой компонент сверхрастущего вектора можно представить в следующем виде:

$$a_1 = a_1;$$
$$a_2 = a_1 + c_0, \text{ где } c_0 \ge 1;$$
$$a_3 = 2a_1 + c_0 + c_1, \text{ где } c_1 \ge 1;$$
$$a_4 = 4a_1 + 2c_0 + c_1 + c_2, \text{ где } c_2 \ge 1.$$
$$\ldots$$

$$a_n = 2^{n-2} a_1 + 2^{n-3} c_0 + 2^{n-4} c_1 + 2^{n-5} c_2 + \ldots + 2^{n-(n-1)} c_{n-4} + c_{n-3} + c_{n-2}, \text{ где } c_{n-2} \ge 1.$$

Понятно, что минимальное значение компонент a_i принимает при $a_1 = c_0 = c_1 = \ldots = c_{i-2} = 1$, но в таком случае мы получим, что

$$a_1 = 2^{i-2} + 2^{i-3} + 2^{i-4} + 2^{i-5} + \ldots + +2^{i-(i-1)} + 2^{i-i} + 1.$$

Фактически данная сумма представляет собой сумму первых *n-1* членов геометрической прогрессии, откуда следует, что $a_i = 2^{i-1} - 1 + 1 = 2^{i-1}$, а это означает, для любого сверхрастущего вектора А: $a_i \ge 2^{i-1}$, что и требовалось доказать.

В качестве следствия из леммы рассмотрим доказательство неравенства $d(A) \le n/(n-1)$.

Доказательство. *A* – сверхрастущий ранцевый вектор, следовательно, по определению, $d(A)=\frac{n}{\log_2 \max A}=\frac{n}{\log_2 a_n}$; из леммы следует, что $\log_2 a_n \ge n-1$, следовательно, $d(A) \le n/(n-1)$, что и требовалось доказать.

Рассмотрим еще один метод определения плотности, основанный на плотности числовой последовательности, предложенный Шнирельманом Л.Г.[4]

Обозначим A(p) – число натуральных чисел ранца *A*, не превосходящих *p*, так что $0 \le A(p) \le p \le n$, следовательно, $0 \le \frac{A(p)}{p} \le 1$. Такая дробь может рассматриваться как средняя плотность ранца на отрезке натурального ряда от 1 до *p* [1].

Минимальное значение данного отношения назовем плотностью ранца и обозначим как $d(A)$, фактически $d(A) = \min_{p \to n} \frac{A(p)}{p}$. При $n \to \infty$ определение плотности ранца фактически сводится к определению плотности последовательности. Так, например, для $A = (1,2,3,...,128)$, $d(A) = 1$, однако в общем случае плотность ранца меньше единицы. Следствием такого определения плотности является:

1. Плотность сверхрастущего ранцевого вектора при $n \to \infty$ стремится к нулю.
2. Плотность ранца равна нулю при отсутствии первого элемента множества, в котором рассматривается ранец.
3. Плотность ранца равна 1, если он полностью совпадает с множеством, из которого он выбран.

Для определения плотности ранца еще одним методом, вернемся к исходной задаче об укладке ранца (1). Количество всех различных допустимых числовых значений v для входа (A,v) ранцевого вектора A назовем мощностью ранца и обозначим её через $\mu(A)$ [5]. Например, для ранца $A = (1,2,4,8)$, $\mu(A) = 16$, а для $A = (1,5,7,14,28,56)$, $\mu(A) = 64$. Понятно, что сверхрастущий ранцевый вектор всегда имеет максимальную мощность, равную 2^n.

Ранцевый вектор A размерности n, n≥3 назовем плотным, если разность между произвольными его двумя соседними компонентами по модулю есть число минимальное, а его мощность $\mu(A)$ имеет возможное наибольшее значение [5]. Понятно, что ранец (3) является плотным. Но также плотным на отрезке [1,128] является и

$$A = (1,2,4,8,...,128). \quad (4)$$

Но (4) является одновременно и инъективным вектором на данном отрезке. Свойство инъективности является существенным. Оно необходимо для синтеза ранцевых систем защиты информации [5].

В заключение приведем две теоремы, фактически описывающие алгоритм построения плотных инъективных обобщенных и универсальных ранцев.

Теорема 1. Обобщенный ранцевый вектор $\tilde{A}_p = (a_1, a_2, ..., a_n)$, размерности n, n ≥ 3 является плотным и инъективным, если $a_1 = c, a_j = p^{j-2}((p-1)c+1)$, j=2...n, где c – некоторая целая положительная константа.

Теорема 2. Универсальный ранцевый вектор $\tilde{A} = (a_1, a_2, ..., a_n)$, размерности n, n ≥ 3 с коэффициентами повторов $Z_t = \{k_1, k_2, ..., k_t\}$ и $Z_m = \{m_1, m_2, ..., m_n\}$, $k_1 + k_2 + ... + k_t = n, t \le n$ является плотным и инъективным, если $a_1 = c, a_j = m^{j-2}((m-1)c+1)$, j=2...n, где c – некоторая целая положительная константа.

Доказательство теорем приведено в работе [5].

Рассмотренные методы определения плотности позволяют использовать эффективные ранцевые алгоритмы, что обеспечивает более высокую надежность системы защиты информации ранцевого типа. Более того, приведенный последним метод позволяет определять плотность для заданного ранца, в том числе обобщенного, универсального и функционального [5,6], что позволяет производить синтез

классической задачи с обобщением на основании свойства плотности и инъективности.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Хинчин А.Я.* Три жемчужины теории чисел.– М.,1979.
2. *Саломаа А.* Криптография с открытым ключом. – М.:Мир. 1995.
3. *Ростовцев А.Г., Маховенко Е.Б.* Теоретическая криптография. –СПб., 2004.
4. *Шнирельман Л.Г.* Простые числа. М.-Л., 1940.
5. *Осипян В.О.* Разработка методов построения систем передачи и защиты информации. – Краснодар, 2004.
6. *Осипян В.О., Мирзаян А.В.* Моделирование плотного рюкзака на основе функционального // Сборник трудов II международной научно-практической конференции "Исследование, разработка и применение высоких технологий в промышленности". Т. 6. – СПб., 7-9 февраля 2006. – С. 172-173.

Мирзаян А.В., Саркисов Т.Г.
Научный руководитель: д.ф.-м.н. Осипян В.О.
г. Краснодар, КГУ

Лозунг: Аутентикация класса OTP – надежна и удобна

О СИСТЕМЕ АУТЕНТИКАЦИИ ОДНОРАЗОВЫМИ ПАРОЛЯМИ

Данная работа посвящена изучению возможности использования криптографической системы, основанной на проблеме универсального рюкзака, для построения подкласса систем аутентикации класса OTP (One Time Password), а также анализу преимуществ построенной системы аутентикации. Информация, представленная в данной работе, предполагает знакомство читателя с принципами работы рюкзачных криптосистем в общем, и систем, основанных на проблеме универсального рюкзака в частности.

Аутентикация, информационная безопасность, OTP, задача о рюкзаке, универсально сверхрастущий вектор.

Актуальность проблемы аутентикации в интернете возрастает из года в год. Интенсивное развитие индустрии компьютерных технологий в целом и в частности сферы услуг, предоставляемых во всемирной паутине, требует надёжной защиты от несанкционированного доступа. При этом экспоненциальное увеличение мощности современных вычислительных устройств требует создания новых систем защиты, более стойких к попыткам взлома «грубой силой». С другой стороны, развитие методов социальной инженерии при относительно низкой граммотности пользователей требуют, чтобы новые системы были устойчивы к таким методам незаконного получения конфиденциальной информации, как «претекстинг», «фишинг», «кви про кво» и др.

На сегодняшний день одними из наиболее надёжных и удобных являются системы аутентикации класса OTP (One Time Password). Обычно одноразовый пароль представляет собой последовательность символов – токен-код, которая меняется в соответствии с некоторым алгоритмом, при этом предыдущая последовательность теряет свою актуальность и аутентикация по ней в дальнейшем невозможна. Реализация основывается на вычислении значения односторонней функции от времени (или значения некоторой синхронизированной с сервером аутентикации последовательности) и некоторого секретного значения. Данные системы

получили широкое распространение благодаря своему удобству, гибкости реализации, стойкости ко многим видам атак.

Тем не менее большинство систем данного класса обладают несколькими существенными недостатками:

1. Требуется синхронизация изменяемых параметров односторонней функции (время или значение некоторой последовательности) с сервером.

2. Аутентикация является односторонней, т.е. сервер аутентифицирует пользователя, однако пользователь не имеет возможности аутентифицировать сервер в рамках данного протокола.

3. Сервер аутентикации хранит у себя секретные параметры пользователя. Следовательно, использование одних и тех же секретных параметров на различных серверах несёт в себе потенциальную угрозу в связи с тем, что взлом одного сервера даст возможность злоумышленнику получить доступ к конфиденциальной информации пользователя на всех остальных серверах.

Рассмотрим реализацию системы аутентикации одноразовыми паролями на основе криптографической системы с использованием проблемы универсального рюкзака, предложенную в работах [2,3,4,5]. Принцип работы системы следующий:

1. Пользователь генерирует универсально сверхрастущий вектор A [4,5], такой, что все его компоненты не повторяются, т.е. $ZK_t=(1,1,\ldots,1)$. Пусть $ZC_p=(m_1, m_2, \ldots, m_n)$ – множество коэффициентов повторения для входа задачи о рюкзаке с вектором A. Вектор A является секретным ключом, а ZC_p будет частью открытого ключа.

2. На основе вектора A генерируется вектор общего вида C в соответствии с некоторым алгоритмом E – алгоритмом «взбалтывания».

3. Вектор C является информацией, хранимой на стороне сервера аутентикации. Он вместе с ZC_p передаётся по открытому каналу либо по защищённому каналу в случае двусторонней аутентикации, когда система позволяет пользователю проверять подлинность сервера.

4. Для аутентикации пользователя сервер генерирует случайным образом последовательность $B = \{b_1, b_2, \ldots b_n\}$, где $b_i<m_i$. Далее генерирует число $D = (C, B)$ – скалярное произведение векторов, которое передаётся пользователю. Таким образом, получаем задачу о рюкзаке с входом (C,D), решением которой является последовательность B.

5. Пользователь, используя алгоритм E^{-1}, преобразует полученное число D в H и раскладывает H по компонентам вектора A за линейное время, т.е. решается задача о рюкзаке с входом (A,H), где A – универсально сверхрастущий вектор.

Детально алгоритм системы аутентикации, а также систем защиты информации на основе рюкзачных криптосистем можно найти в работах [2,3,4,5]. Стоит отметить алгоритм «взбалтывания». В классическом случае этим алгоритмом является сильное модульное умножение, однако в связи с тем, что алгоритм преобразования может стать слабым местом системы, в работах [4] были проведены исследования по созданию алгоритма преобразования сверхрастущего вектора A в вектор общего вида C на основе решения системы диофантовых уравнений [4]. В рамках данной статьи мы опускаем вопрос об использованном алгоритме «взбалтывания», так как этот вопрос требует отдельного рассмотрения в связи тем, что использование нестойкого алгоритма нарушит стойкость всей системы.

Отметим, что в приведённом алгоритме полностью отсутствует необходимость постоянной синхронизации какой-либо информации с сервером – сервер генерирует произвольные последовательности из очень большого диапазона, более p^N значений, где $N=\min(m_i)$.

Если на шаге 3 используется передача вектора C на сервер по защищённому каналу, то аутентикация является двусторонней. Пользователь может проверить подлинность сервера по числу D: если число D после преобразования раскладывается по компонентам вектора A, следовательно, число D было получено с использованием вектора C, которым владеет сервер. Если же такое разложение не может быть получено, то подлинность сервера не подтверждается. Отметим, что здесь возможно наложение дополнительных ограничений на вектор A, в частности, вектор не должен быть плотным.

Однако же самым важным достоинством предложенной системы является то, что сервер не владеет секретной информацией пользователя. При несанкционированном доступе на сервер злоумышленник может получить только вектор C. Разложение числа на компоненты нормального вектора является задачей экспоненциальной сложности. Таким образом, пользователь может безопасно использовать одни и те же секретные данные для аутентикации на различных серверах, не опасаясь того, что взлом одного из них поставит под угрозу конфиденциальную информацию на других.

Таким образом, предложенная система является функциональной, удобной в использовании и стойкой по отношению ко многим видам интернет-атак. Тем не менее всегда стоит помнить, что аутентикация является всего лишь одной из компонент информационной безопасности наряду с системами управления доступом, средствами организации VPN, средствами противостояния пассивным атакам, средствами анализа защищенности сервиса, а также компьютерной грамотностью пользователей системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Шеннон К.* Работы по теории информации и кибернетики. – М.: ИЛ, 1963.
2. *Саломаа А.* Криптография с открытым ключом. – М. 1995.
3. *Осипян В. О.* Об одном обобщении рюкзачных криптосистем //Изв. вузов. Сев. - Кавк. регион. Тех. науки. 2003. Прил. №5. – С. 18-25.
4. *Осипян В.О.* Моделирование СЗИ, содержащих диофантовую трудность. Материалы VII международной научно-практической конференции. – Таганрог, 2005. – С. 202-209.
5. *Осипян В.О Мирзаян А.В., Саркисов Т.Г.* Система аутентикации на основе проблемы универсального рюкзака. Материалы X международной научно-практической конференции. Ч. II. – Таганрог, 2008. – С. 163-166.

Тропников Н.В.
Консультанты: д.м.н., профессор Журавлев Б.В.;
к.т.н. Карабачинский А.Л.; к.т.н. Корольков В.А.
г. Москва, в/ч 43753

Лозунг: Овладевая сегодня инструментом управления информацией, завтра обеспечиваем себе превосходство в информационном мире

ИССЛЕДОВАНИЯ МЕТОДОВ ВЫЯВЛЕНИЯ ДЕЙСТВИЙ СКРЫТОГО ИНФОРМАЦИОННОГО ВОЗДЕЙСТВИЯ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

Важное значение при выявлении действий скрытого информационного воздействия будут иметь раннее предупреждение о возможных угрозах, разведывательная информация, полученная из сети при помощи новых методов выявления скрытого информационного воздействия, позволит нам дать оценку атакующим

средствам противника и промоделировать вероятность уязвимости своих компьютерных систем, что даст нам преимущество в разработке адекватных средств защиты.

Компьютерные системы, скрытое информационное воздействие, инструмент управления информацией, информационно-психологическая безопасность, информационное противоборство, специальные автоматизированные системы, психофизиологические средства выявления.

Важнейшим атрибутом нашего времени является глобальная информационная интеграция, основанная на построении компьютерных сетей масштаба предприятия и их объединении посредством сети Internet.

Сложность логической и физической организации современных сетей приводит к объективным трудностям при решении вопросов управления и защиты сетей. В процессе эксплуатации компьютерных сетей администраторам приходится решать две главные задачи:

- диагностировать работу сети и подключенных к ней серверов, рабочих станций и соответствующего программного обеспечения;
- защищать информационные ресурсы сети от несанкционированной деятельности хакеров, воздействий вирусов, сетевых червей и т.п., т.е. обеспечивать их конфиденциальность, целостность и доступность.

При решении задач, связанных с диагностикой и защитой сетевых ресурсов, центральным вопросом является оперативное обнаружение состояний сети, приводящих к потере полной или частичной ее работоспособности, уничтожению, искажению или утечке информации, являющихся следствием отказов, сбоев случайного характера или результатом получения злоумышленником несанкционированного доступа к сетевым ресурсам, проникновения сетевых червей, вирусов и других угроз информационной безопасности. Раннее обнаружение таких состояний позволит своевременно устранить их причину, а также предотвратит возможные катастрофические последствия.

Для их обнаружения используется большой спектр специализированных систем. Так, при решении проблем диагностики сетей применяются средства систем управления, анализаторы сетевых протоколов, системы нагрузочного тестирования, системы сетевого мониторинга. Проблемы защиты информационных ресурсов сетей решаются с помощью межсетевых экранов (firewall), антивирусов, систем обнаружения атак (IDS), систем контроля целостности, криптографических средств защиты.

Характерными особенностями использования этих систем является либо их периодическое и кратковременное применение для решения определенной задачи, либо постоянное использование, но с довольно статическими настройками.

Так, методы анализа, используемые в современных системах, направлены на обнаружение известных и точно описанных типов воздействий, но зачастую оказываются не в состоянии обнаружить их модификации или новые типы, что делает их использование малоэффективным.

Таким образом, на сегодняшний день очень актуальной задачей является поиск более эффективных методов выявления недопустимых событий (аномалий) в работе сети, являющихся следствием технических сбоев или несанкционированных воздействий. Основным требованием к этим методам является возможность обнаружения произвольных типов аномалий, в том числе новых, а также воздействий, распределенных во времени.

Это направление научных исследований является очень молодым. Первые работы, посвященные данной проблеме, были опубликованы в 90-х годах прошлого столетия.

В настоящий момент исследования в этой области ведутся как крупными зарубежными коммерческими компаниями (Cisco, Computer Associates, ISS, Symantec и др.), так и университетскими научно-исследовательскими центрами (Columbia University, Florida Institute of Technology, Purdue University, Ohio University и др.).

Способы выявления уязвимостей в программном коде

Сегодня применяются два основных подхода к выявлению уязвимостей кода.

Во-первых, это структурный статический и динамический анализ исходного кода, регламентируемый Руководящим документом (РД) Гостехкомиссии России по контролю над недекларированными возможностями [1].

Во-вторых, сигнатурно-эвристический анализ потенциально опасных операций, заключающийся в сканировании кода программы на наличие таких операций и последующем ручном или автоматическом анализе фрагмента кода для выявления реальной угрозы для программного обеспечения.

Очевидно, что второй подход лишен недостатков избыточной структуризации всего программного обеспечения и «проклятья размерности» полномаршрутного тестирования. Поскольку число потенциально опасных операций, как правило, не превышает 5-10% объема программного обеспечения, в десять-двадцать раз снижается время «ручного» анализа исходного текста. В отношении статического и динамического анализа следует подчеркнуть, что результаты статического анализа по сложности интерпретации сопоставимы с исходными текстами, а динамический анализ дополнительно требует составления и реализации соответствующих тестов маршрутов. Таким образом, сигнатурно-эвристический анализ сокращает временные затраты на поиск недекларированных возможностей в десятки раз. Опыт специалистов (Алексей Марков – руководитель, Сергей Миронов и Валентин Цирлов – ведущие эксперты Испытательного центра НПП «БИТ» (Москва)) показывает, что большинство выявленных на этапе сертификационных испытаний уязвимостей кода обнаружены благодаря использованию именно второго подхода.

Пример неординарного применения целенаправленного изменения программного кода

Человечество в стремлении получить доступ к открытым мировым информационным ресурсам, по сути своей, открыло «ящик Пандоры», так как не осознавая этого, открыла возможность для воздействия на общемировые политические, экономические и технологические процессы.

Развитие компьютерных технологий способствовало появлению новых, весьма своеобразных и мощных средств воздействия на информационную среду и психику человека, формирование общественного мнения.

Информационное оружие уже сейчас получает статус самостоятельного, а в будущем способно увеличить свой наступательный потенциал.

Информационная война – это широкомасштабное использование:

1) методов и средств информационно-психологического воздействия на личный состав и население (пропаганда, внушение, дезинформация и т.д.);
2) средств уничтожения, искажения или хищения информационных массивов, ограничения или недопущения законных пользователей технических средств к своим базам данных, дезорганизация работы технических

средств, полного вывода из строя радио- и телекоммуникационных сетей и компьютерных систем.

Интеллектуальное и психологическое состояние человека, производство и управление, оборона и безопасность, финансы и бизнес, наука и образование, средства массовой информации (СМИ) – главные объекты воздействия информационного оружия.

Проект Федерального Закона «Об информационно-психологической безопасности», подготовленный рабочей группой Комитета Государственной Думы РФ по безопасности, был внесен 3 декабря 1999 года, в порядке законодательной инициативы депутатом ГД РФ второго созыва Лопатиным В.Н., который возглавлял это направление законодательной работы в российском парламенте. Не прошел, задаем правильный вопрос: «Почему?», по моему мнению, политика по ведению информационной безопасности других государств, видимо, работает лучше.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Руководящий документ. Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей. – М.: Гостехкомиссия России, 1998.

2. *Калайда И.А.* Недекларированные возможности в программном обеспечении средств защиты информации. Jet Info, 2000, № 8.

3. *Марков А.С., Щербина С.А.* Испытания и контроль программных ресурсов. InformationSecurity, 2003, № 6.

4. *Дастин Э., Рэшке Д., Пол Д.* Автоматизированное тестирование программного обеспечения: внедрение, управление, эксплуатация. – М.: Лори, 2003.

5. ГОСТ Р ИСО/МЭК 15408-2002. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Части 1, 2, 3. – М.: ИПК «Изд-во стандартов», 2002.

6. Методическое руководство по оценке качества функционирования информационных систем. – М.: Изд-во 3 ЦНИИ МО РФ, 2003.

7. ГОСТ РВ 51719-2001. Испытания программной продукции. – М.: ИПК «Изд-во стандартов», 2001.

8. Руководящий документ. Безопасность информационных технологий. Критерии оценки безопасности информационных технологий. Части 1, 2, 3. – М.: Гостехкомиссия России, 2002.

9. Скрытые каналы // Тематический сборник. Jet Info, 2002, № 11.

10. *Хогланд Г., Мак-Гроу Г.* Взлом программного обеспечения: анализ и использование кода. – М.: Вильямс, 2005.

Харечкин П.В.
Научный руководитель: к.т.н., доцент Лепёшкин О.М.
г. Ставрополь, СГУ

Лозунг: Сложным системам – активное управление доступом!

ПОДХОД К ОПИСАНИЮ ДИНАМИЧЕСКОЙ СИСТЕМЫ УПРАВЛЕНИЯ ДОСТУПОМ В СОЦИОТЕХНИЧЕСКИХ СИСТЕМАХ НА ОСНОВЕ СЕТЕЙ ПЕТРИ

Управление доступом – это предоставление субъектам права оперировать над объектами. Сегодня правила и системы управления доступом являются статическими, они редко изменяются в течение времени. В статье показано, как в

социотехнических системах реализовать динамическое назначение полномочий посредством декомпозиции системы по функциям и задачам. Права доступа предоставляются согласно состоянию процесса выполнения потока задач. Таким образом, риск недопустимого использования данных уменьшен, что подтверждается моделью, представленной в статье. Потоки задач описываются сетями Петри, которые представляют собой надежный математический аппарат и хорошо подходят для описания дискретных моделей.

Управление доступом, социотехническая система, функциональная безопасность, задача, сети Петри.

С постоянным увеличением масштабов применения информационных технологий человек все чаще в своей деятельности сталкивается с социотехническими системами (СТС), которые представляют собой системное образование, включающее технико-технологическую подсистему и систему ролей и функций обслуживающего и управленческого персонала [1].

Среди СТС следует выделить критические СТС, выполняющие стратегически важные функции, для которых вопросы надежности, устойчивости и живучести имеют особое значение. Основу таких систем составляет взаимодействие человека и техники посредством информационных технологий, что приводит к необходимости обеспечения, с одной стороны, требований информационной безопасности (ИБ), а с другой – выполнения функциональных требований, то есть функциональной безопасности (ФБ).

Для создания эффективных механизмов обеспечения безопасных условий функционирования СТС недостаточно использовать подход, реализующий лишь методы ИБ и надежности, так как эти рамки ограничиваются задачами поддержания безопасных условий функционирования систем в части защиты от несанкционированного доступа (НСД), а также сохранения способности нормального функционирования СТС после дестабилизирующих воздействий. Таким образом, наряду с понятием ИБ встает понятие ФБ.

Задачей ФБ [2] является обеспечение такого состояния системы, при котором выполняется определенный набор функций системы в соответствии со своим назначением в реальном масштабе времени. На данный набор функций наложены соответствующие требования, и в случае неудовлетворения их, могут привести к отказам, последствиями которых будет недопустимый ущерб.

Для анализа СТС в аспекте ФБ необходимо провести декомпозицию системы по функциям и задачам, где функция F будет представлена совокупностью задач T, связанных отношениями последовательности и параллельности, что представлено на рис. 1.

Основные составные части задачи – действия, временной и логический порядок совершения которых задан управляющим воздействием. Чтобы описать действия, необходимо определить, каким субъектам позволяется выполнять действия и какие данные необходимы и создаются во время выполнения процесса.

Современная безопасность основывается на использовании моделей разграничения доступа [3], которые обеспечивают конфиденциальность и целостность информационных ресурсов в реальном масштабе времени. Обеспечение ИБ СТС осуществляется на основе систем управления доступом. Управление доступом – это определение возможности субъекта оперировать над объектом.

Субъекты могут быть ассоциированы с людьми, а также группами людей, ролями и компьютерными программами – потенциальными субъектами. Практически, понятие ролей и ролевых моделей широко распространено [4]. Выполнение

действия привязано к определенной роли, и служащий в компании может активизировать одну или более ролей.

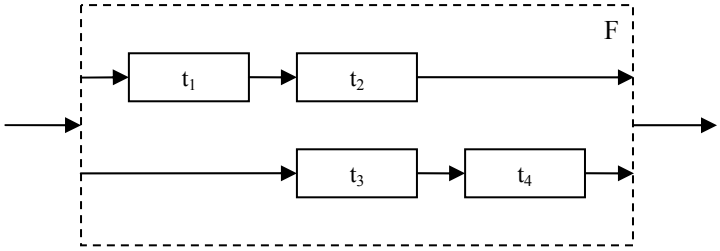


Рис. 1. Представление функции СТС в виде совокупности задач

Современные системы управления доступом разрабатываются на основе классических моделей разграничения доступа, таких как дискреционные, мандатные, ролевые [5]. Информационная система описывается на стадии проектирования, из-за чего правила разграничения доступа между субъектами *S* и данными *D* вводятся однократно ко всей системе в целом и не меняются при переходе пользователем от выполнения одной задачи к другой.

Таким образом, возникает противоречие между СТС и моделями управления доступом, заключающееся в отсутствии отображения данных моделей на модели процессов СТС.

Классический подход к управлению доступом можно расширить понятием потока задач – workflow. Если текущая задача учитывается при предоставлении прав доступа *P(t)*, то из всего множества полномочий пользователя *P*, отражающих права доступа к объектам всей СТС, ему доступны лишь те полномочия, которые необходимы для выполнения текущей задачи:

$$P = P(t) .$$

Таким образом, выполнение задачи становится более безопасным, потому что возможности для НСД к данным и неправильного использования данных, например несанкционированное чтение файла, уменьшены. Кроме того, в таком механизме управления доступом учитывается правило минимальных привилегий, потому что субъектам будут предоставлены только те полномочия, которые необходимы для текущей работы.

В соответствии с BPM (Business Performance Management) и workflow [6] поток задач может быть представлен в виде сети. Описание сетями Петри дает следующие преимущества:

1. Сети Петри хорошо подходят для описания дискретных динамических моделей. Процесс и его исполнение – это модель, потому что отсчеты во времени, соответствующие выполнению действий, формируют дискретный набор на оси времени.

2. Сети Петри – строго формализованный математический аппарат.

3. Если поток задач описан сетью Петри, такие свойства, как достижимость конечного состояния и живучесть, математически доказуемы. Существует большое количество методов анализа сетей Петри.

Сети Петри, автором которых является Карл Адам Петри [7], на данный момент широко используются.

Сеть Петри – тройка $N = \{P, T, F\}$. P – множество позиций, T – множество переходов, причем $P \cap T = \varnothing$. Отношение потоков F описывается следующим образом:

$$F \subseteq (P \times T) \cup (T \times P).$$

На рис. 2 представлен пример сети Петри. Сеть состоит из позиций $p_1,..., p_4$ и переходов $t_1,...t_5$.

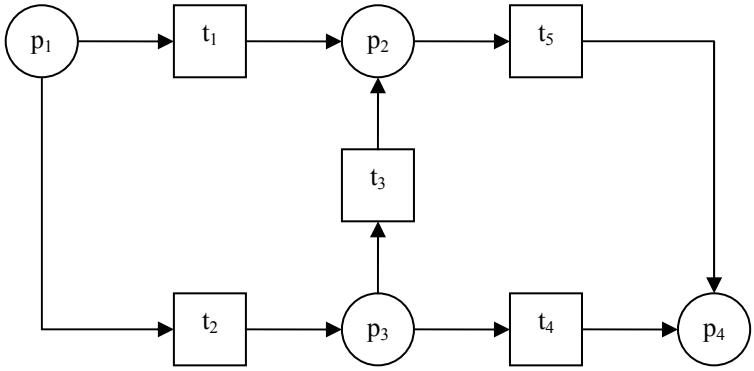


Рис. 2. Пример сети Петри

Множества определяются следующим образом:

$$P = \{p_1,..., p_4\},$$
$$T = \{t_1,..., t_5\},$$
$$F = \{(p_1, t_1), (p_1, t_2), (t_1, p_2), (t_2, p_3), (p_3, t_3), (t_3, p_2), (p_2, t_5), (t_5, p_4), (p_3, t_4), (t_4, p_4)\}.$$

Графическая интерпретация сети Петри – двудольный граф. Позиции могут быть соединены только с переходами, переходы могут быть соединены только с позициями. Позиции представлены графически как круги, переходы как прямоугольники. Графическая интерпретация элемента $(x, y) \in F$ – это стрелка от x к y. В примере $(t_1, p_2) \in F$. Поэтому стрелка соединяет переход t_1 с позицией p_2.

Поток задач характеризуется свойствами последовательности, параллельности и условности. Последовательность может быть смоделирована сетями Петри через переходы и позиции только с одной позицией входа и выхода или переходом. Параллельность базируется на переходах со множественными позициями выхода. Третье понятие – условность – основано на позициях со множественными переходами выхода. Позиции p_1 и p_3 на рис. 2 являются примерами.

Поток задач характеризуется следующими характеристиками:

1. Действия в процессах соответствуют переходам в сетях Петри. Выполнение действия соответствует срабатыванию перехода.

2. Маркировка сети Петри представляет текущее состояние процесса. Маркеры также называют фишками. Фишки представляют собой состояние процесса, т.е. показывают, какие действия активизированы. Отношения показывают, как маркеры могут двигаться в сети.

Пока не было показано, какие субъекты и данные связаны с каждым действием. Для более детального описания необходимо указать, какие субъекты могут выполнять действия и какие для этого данные необходимы и будут созданы при выполнении действий. Это осуществляется тремя функциями Sbj, D_{in} и D_{out}.

Функция Sbj отображает множество переходов на множество субъектов. Эти субъекты могут выполнять действия, связанные с переходом:

$$Sbj : T \to S,$$

где S – множество субъектов. С одной стороны, выполнение действия приводит к созданию данных. С другой стороны, чтобы выполнить действие, субъект использует данные, которые уже существуют. Множество входов данных и выходов перехода описывается через эти функции D_{in} и D_{out}:

$$D_{in}, D_{out} : T \to D,$$

где D – множество всех данных. Пустое множество – допустимое значение. В этом случае никакие данные не необходимы для выполнения, или никакие данные не созданы при выполнении действия.

Рис. 3 расширяет рис. 2, дополняя его атрибутами переходов в виде функций D_{in}, D_{out}, Sbj.

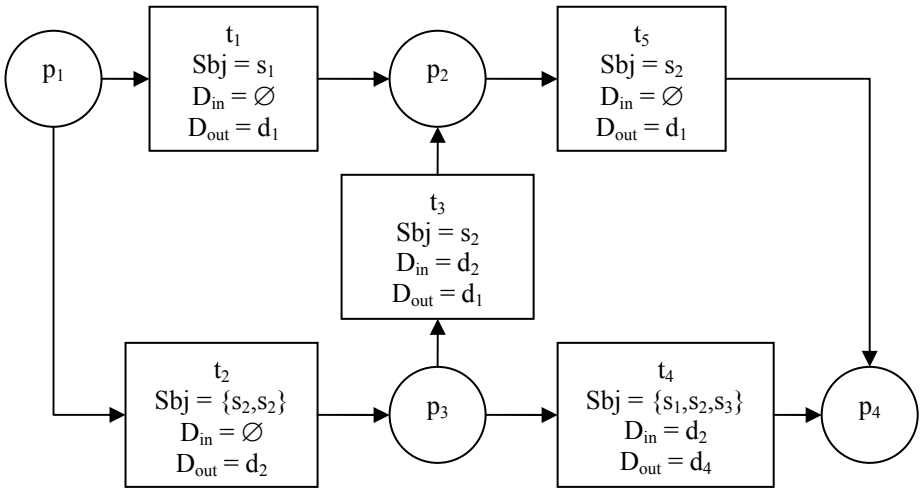


Рис. 3. Схема управления доступом на основе сетей Петри

В рис. 3 два множества S, D являются $S = \{s_1, s_2, s_3\}$, $D = \{d_1, ..., d_5\}$. Переход t_1 может только быть выполнен субъектом s_1. Нескольким субъектам можно

назначить переход, например, переход t_4 может быть выполнен тремя субъектами s_1, s_2, s_3. Субъекту s_2 необходимы данные d_2, чтобы выполнить переход t_3. При выполнении t_3 s_2 создает d_3. Значения D_{in} и D_{out} могут быть пустыми: $D_{in}(t_2) = \varnothing$. В этом случае никакие данные не нужны, чтобы выполнить действия, связанные с переходом.

Определить, выполнима ли функциональная сеть полностью, т.е., достижима ли конечная маркировка из маркировки начала, – нетривиальная задача. Она связана с проблемой достижимости в сетях Петри, для которой были введены такие понятия, как живучесть, инвариантность, тупиковые состояния [7].

Цель управления доступом состоит в том, чтобы предоставить доступ к данным только уполномоченным субъектам. Обычно используются права доступа на чтение (*r*) и запись (*w*):

$$P = S \times D .$$

Если права доступа предоставляют согласно текущему состоянию сети, то изменение прав доступа происходит при маркировке сети Петри. Права доступа, предоставляемые субъекту, теперь зависят от выполняемой задачи и называются динамическими полномочиями P_{dyn}:

$$P_{dyn} = S \times D \times T .$$

Пусть субъект *s* запрашивает право на чтение данных *d*. Модель управления доступом проверяет текущую маркировку сети для всех задействованных переходов. Пусть *t* является задействованным переходом и $s \in Sbj(t)$.

1. Если $r \in P_{dyn}(s, d, t)$, право *r* предоставляется. Если *t* не задействован более, то право *r* отменяется.

2. Если $r \notin P_{dyn}(s, d, t)$, право *r* не предоставляется.

Таким образом, классическое и динамическое управления доступом связаны следующим соотношением:

$$P(s, d) = \bigcup_{t \in T} P_{dyn}(s, d, t) .$$

Если имеет место применение *rw*-политики безопасности, правила управления доступом выражены следующим образом:

$$P_{dyn} = \{r\},\ \textit{если}\ \ s \in Sbj(t)\ \ u\ \ d \in D_{in}(t),$$
$$P_{dyn} = \{w\},\ \textit{если}\ \ s \in Sbj(t)\ \ u\ \ d \in D_{out}(t),$$

где $s \in S$, $d \in D$ и $t \in T$. $P_{dyn}(s, d, t) = \{r, w\}$ недопустимы, потому что данные должны быть записаны прежде, чем прочитаны.

Представленный динамический подход обладает следующими преимуществами:

– ограничение полномочий в зависимости от текущей задачи и состояния системы в целом;

– оценка состояния и функционирования системы на основе контроля информационных потоков между задачами;

– анализ живучести и достижимости конечной цели системы в режиме реального масштаба времени.

Динамический подход к описанию системы управления доступом на основе сетей Петри позволит описывать функционирование СТС в режиме реального времени и обеспечивать разграничение доступа субъектов к данным согласно правилу минимальных привилегий.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Волобуев С.В.* Философия безопасности социотехнических систем. – М.: Вузовская книга, 2004. – 360 с.

2. *Липаев В.В.* Функциональная безопасность программных средств. – М.: СИНТЕГ, 2004. – 348 с.

3. *Девянин П.Н.* Модели безопасности компьютерных систем. – М.: Издательский центр «Академия», 2005. – 144 с.

4. *Баранов А.П., Борисенко Н.П., Зегжда Д.П., Корт С.С., Ростовцев А.Г.* Математические основы информационной безопасности. – Орел: Военный институт правительственной связи, 1997. – 284 с.

5. *Гайдамакин Н.А.* Разграничение доступа к информации в компьютерных системах. – Издательство Уральского ун-та, 2003. – 328 с.

6. Thomas R.K., Sandhu R.S. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enteprise-oriented Authorization Management. In Proceedings of the IFIP WG11.3 Workshop on Database Security, Lake Tahoe, California, August 11-13, 1997.

7. *Котов В.Е.* Сети Петри. – М.: Наука. Главная редакция физико-математической литературы, 1984. – 160 с.

Гонцов Д.В.
Научный руководитель: к.т.н., доцент Макаров Ю.Н.
г. Челябинск, ЮУрГУ

Лозунг: Всем порталам для grid-сетей – защита по-максимуму!

МОДЕЛЬ БЕЗОПАСНОСТИ ВЕБ-ПОРТАЛА РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ GRID-СЕТИ

В статье рассматриваются основные механизмы системы безопасности веб-порталов для grid-сетей. Дается обзор основных угроз для подобных веб-порталов, а также предлагается модель построения защищенного портала для grid-сети.

Grid, веб-портал, распределенные вычисления, параллельные вычисления.

Технологии создания и использования систем высокопроизводительных вычислений постоянно совершенствуются, но наиболее перспективными с точки зрения соотношения конечного результата и трудозатрат на создание и поддержку благодаря своей уникальной структуре являются grid-сети. Технология grid-вычислений представляет собой виртуализированную параллельную распределенную вычислительную среду. Целью создания подобной среды является получение возможности динамического изменения цикла программы, распределения и объединения (географически) разделенных автономных ресурсов, основываясь на их возможностях, параметрах доступности, производительности и стоимости, а также

на основе специфичности задачи организации и/или экстренной необходимости в вычислительных ресурсах.

На сегодняшний день одним из самых перспективных направлений развития grid-сред является создание grid-порталов, которые позволят всем пользователям высокопроизводительных вычислительных ресурсов получать быстрый и удобный доступ к ним через тонкий интерфейс web-оболочки.

Grid-портал – это стандартный подход к обеспечению пользователям интерфейса для различных grid-приложений. Используя стандартные веб-технологии, портал является простейшей графической средой доступа к сервисам grid-сети. Портал работает на самом высоком уровне относительно всех остальных составляющих сети grid (см. рис. 1).

1. Существует множество способов для организации подобных порталов, и практически все порталы на сегодняшний день используют в качестве базовой инфраструктуры grid-инструментарий Globus Toolkit.

2. Система безопасности grid-портала должна решать следующие высокоуровневые задачи:

- Единая точка для создания учетных записей пользователей в хранилище идентификационных данных сети grid (Single Sine On).
- Аутентификация пользователей на портале, позволяющая им в дальнейшем работать с ресурсами сети.
- Система управления единым доступом (Single Sine On). Система управления идентификационными данными пользователя для их автоматической передачи всем основным службам сети grid.
- Авторизация. Система, позволяющая администратору изменять права доступа пользователей на пользование ресурсами в глобальном списке прав пользователей.

3. Следует отметить, что данный перечень функций представляет собой минимально допустимый набор служб для функционирования веб-портала распределенной вычислительной сети.

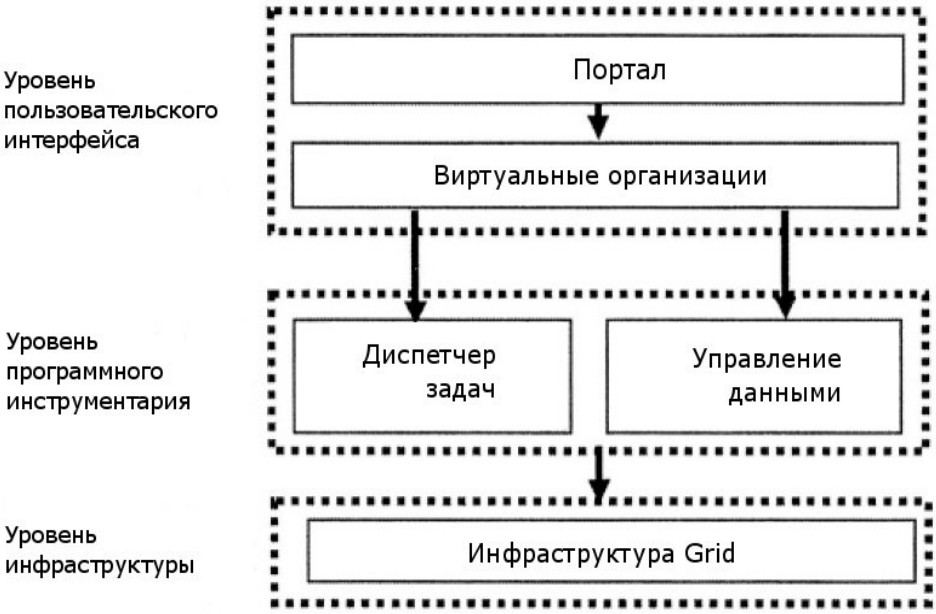


Рис. 1. Трехуровневая схема среды Grid

Архитектура простейшего grid-портала изображена на рис. 2. То есть grid-портал является веб-оболочкой для доступа к базовым сервисам сети grid, которые включают в себя инфраструктуру безопасности grid (основанную в свою очередь на инфраструктуре PKI), службу постановки задач в общую очередь, службу GSI-FTP для доступа пользователя к своим данным и службу диспетчера для ресурсов хранения данных. Инфраструктура Grid Security Infrastructure (GSI) обеспечивает базовые механизмы для функционирования распределенной вычислительной сети и решает вопросы PKI-аутентификации, защиты канала передачи данных внутри сети и создания временных идентификационных данных для пользователей.

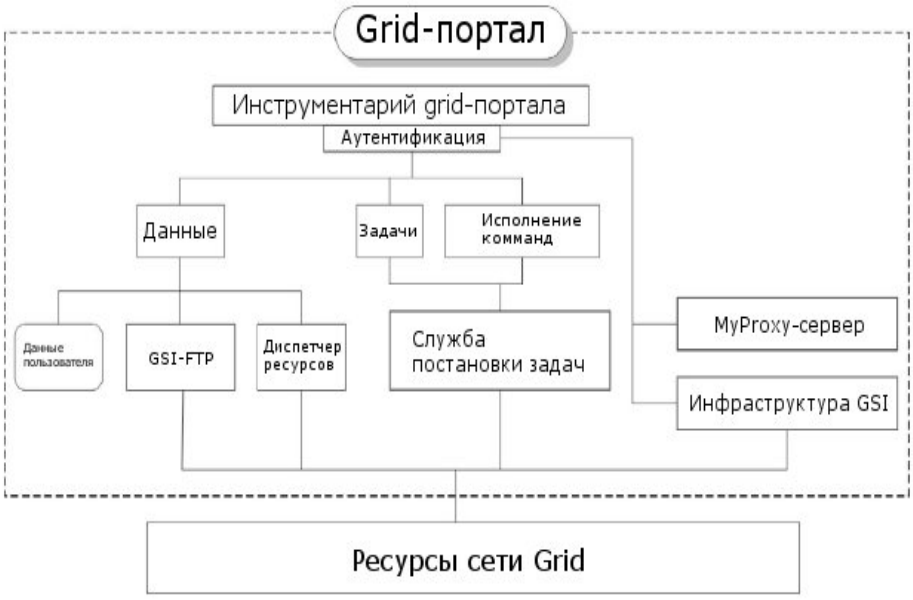


Рис. 2. Архитектура grid-портала

4. Главной проблемой существующих веб-порталов является то, что ни один из них не является полноценной и полностью интегрированной составляющей среды grid. И с точки зрения злоумышленника, не обладающего физическим доступом ни к одному узлу сети, он будет являться самым слабым звеном в защите сети, а его атака – самым простым способом получения контроля над ее ресурсами.

5. Основными типами уязвимостей, характерных для grid-портала (как частного случая веб-приложения), являются:

- Непроверенные на допустимость получаемые от пользователей данные. Система может начать работу в соответствии с введенными пользователем в web-форму данными, не проверив их на допустимость, что может привести к непредсказуемым результатам.
- Ошибки в системе контроля доступа. Критично для интерфейсов администрирования.
- Ошибки в подсистеме аутентификации и управления сессиями.
- Инъекции вредоносного кода.
- Неправильная обработка ошибок.
- Небезопасное хранилище идентификационных данных.
- Ошибки класса «Отказ в обслуживании».
- Переполнение буфера.

- Небезопасные механизм изменения конфигурации.

6. Основная особенность веб-портала, с которой связанны его дополнительные уязвимости. Это то, что grid-портал сам управляет идентификационными данными grid от имени своих пользователей. Вследствие чего в общем случае злоумышленник, получив управление порталом, не имеет прямого доступа к ресурсам всей вычислительной сети, но он может получить полный доступ к идентификационным данным пользователей, что в свою очередь является критической угрозой безопасности всей распределенной вычислительной сети.

Для полноценной защиты необходим механизм связи подсистем безопасности веб-портала и основного программного инструментария среды grid. То есть включение непосредственно в портал модуля безопасности для обмена данными со службой GSI, чтобы она в свою очередь могла определять состояние портала с точки зрения безопасности. То есть осуществлять мониторинг сессий, журналирование событий и проверку целостности основных модулей.

Для реализации данной модели был создан веб-портал, в качестве технической базы которого был выбран инструментарий GridSphere 3.1. Для создания базовой инфраструктуры grid-сети использовано программное обеспечение Globus Toolkit 4.2.1. Таким образом, реализация модели представляет собой модуль PortalSecurityIntegration для инструментария GridSphere на программном языке Java. Данный модуль реализует механизм дополнительного взаимодействия системы безопасности портала со службой GSI. Модуль решает следующие задачи:

- Мониторинг сессий всех пользователей на портале. Отслеживание действий пользователей с целью выявить попытки захвата контроля на порталом или увеличения своих полномочий в сети grid.
- Журналирование событий, происходящих на портале в глобальном журнале событий GSI.
- Проверку целостности основных модулей сайта. Проверка целостности программного кода портала и выявление возможного использования злоумышленниками своего вредоносного кода.
- Проверка правомочности действий пользователей. Используется алгоритм для анализа всех совершаемых операции через портал с точки зрения текущей системы управления доверием, права на доступ на определенный ресурс, квоты, очередности поставленных на выполнение задач. Сам анализ осуществляется с помощью информации от подсистемы безопасности GSI о допустимости данных действий.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Enterprise Grid Security Requirements. Enterprise Grid Alliance Security Working Group, 2005. -37 p.

2. *Zhou Q, Yang G*. A Grid Portal Model Based on Security and Storage Resource Proxy.// Proc. Computational and information science, CIS 2004, pp. 13–18.

3. *Minoli D.* A Networking Approach to Grid Computing. –John Wiley & Sons, 2005. – 396 p

4. *Chakrabarti A.* Grid Computing Security. – Springer, 2007. – 331p.

Климов С.М., Климов М.К.
Ростов-на-Дону, РВИ РВ

Лозунг: Уровни защиты систем электропитания

МНОГОУРОВНЕВАЯ ЗАЩИТА ИНФОРМАЦИОННЫХ ПРОЦЕССОВ В КОМПЬЮТЕРИЗИРОВАННЫХ СИСТЕМАХ ЭЛЕКТРОСНАБЖЕНИЯ СЛОЖНЫХ ТЕХНОЛОГИЧЕСКИХ КОМПЛЕКСОВ

В статье рассматриваются предложения, направленные на разработку и создание многоуровневой системы защиты информационных процессов в компьютеризированных электроснабжения сложных технологических комплексов (СТК). Реализация рассмотренных предложений обеспечивает возможность повышения эффективности функционирования электроснабжения СТК.

Автоматизированное рабочее место, система электроснабжения сложных технологических комплексов, химические источники тока, информационный процесс, специальное программное обеспечение.

По сообщениям журнала The Wall Street Journal (США), американская энергосистема состоит из трех отдельных сегментов, каждый из которых включает в себя тысячи миль проводов, а также электростанции и подстанции. По мере интеграции интернет-технологий в систему управления электросетями возрастает и их уязвимость перед хакерскими атаками. Зафиксирован случай проникновения в компьютеры американской системы энергоснабжения и установления там программы, с помощью которой, в принципе, можно было нарушить работу этой системы [1].

Известные события выхода из строя только одной системы электропитания в таком крупном мегаполисе, как г. Москва показали, что следствием такого техногенного нарушения явилась потеря управляемости наземным, подземным транспортом и связью. В условиях интенсивной компьютеризации и усложнения систем электропитания возникает угроза нарушения их функционирования при воздействии компьютерных атак внутреннего нарушителя или через уязвимые места для несанкционированного доступа по беспроводному каналу связи. Масштабы ущерба от такого воздействия сравнимы с техническим выходом из строя системы электропитания.

Следует отметить, что на практике в контур автоматизированного управления систем электроснабжения (СЭС) внедряется стандартное компьютерное и коммуникационное оборудование, программное обеспечение общего пользования с присущими им уязвимостями и угрозами.

Одними из динамически развивающихся систем являются современные системы электроснабжения сложных технологических комплексов (СЭС СТК), которые предназначены для гарантированного электроснабжения переменным и постоянным токами объектов специального назначения.

Они состоят из электроустановок, включающих электроагрегаты, преобразовательные устройства, химические источники тока (ХИТ) и средства автоматизированного управления и защиты.

Для более гибкого автоматизированного управления, обеспечения высокой надежности функционирования, снижения энергопотребления осуществляется компьютеризация средств управления СЭС СТК.

Для этих целей разрабатывается специальное программное обеспечение (СПО) с использованием цифровой аппаратуры.

В ходе эксплуатации средств СЭС СТК осуществляются следующие информационные процессы (рис. 1):

- контроль с помощью датчиков значений параметров СЭС СТК, например, напряжения, частоты тока, мощности, скорости вращения приводов, давления масла и температуры охлаждающей жидкости в магистралях, времени работы электроустановок (реализуется с помощью программно-аппаратных датчиков СЭС СТК);
- сбор и обработка технологической информации, получаемой от датчиков СЭС СТК, сравнение текущих значений параметров с базой эталонных данных (осуществляется сервером сбора информации от датчиков);
- формирование разностных сигналов управления и передача по технологическим каналам на автоматизированные рабочие места (АРМ) пункта управления СЭС СТК (осуществляется средствами электронной почты и управления СЭС СТК).

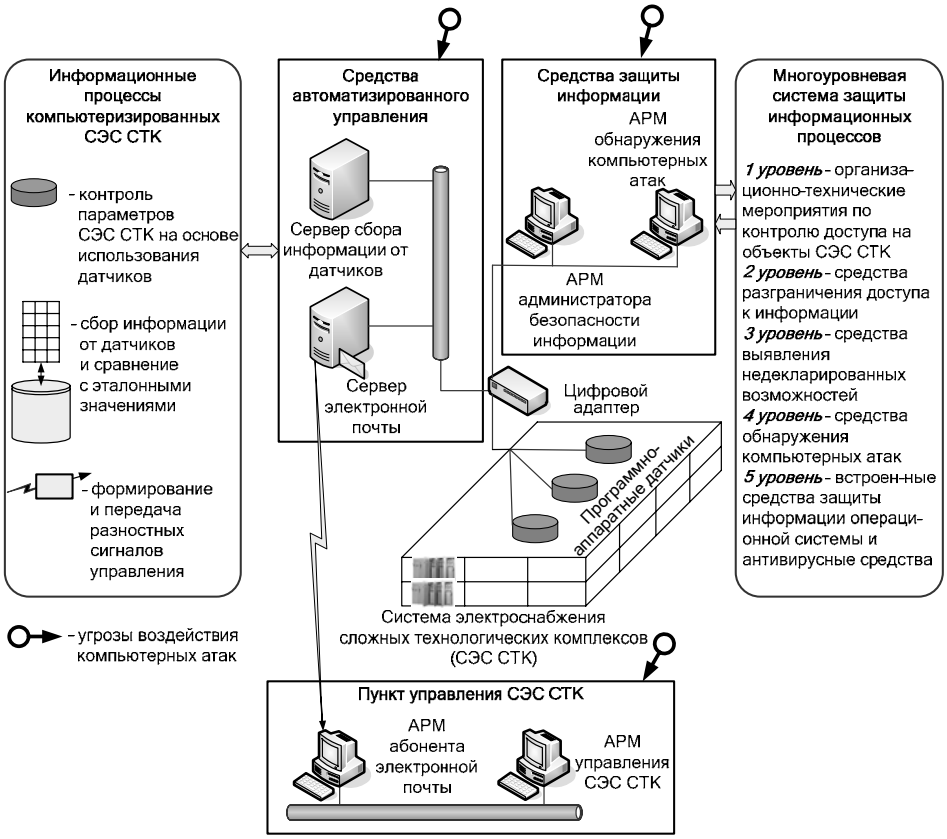


Рис. 1. Структура многоуровневой защиты информационных процессов в компьютеризированных СЭС СТК

Содержание информационных процессов о сигналах управления СЭС СТК зависит от режима ее работы.

В дежурном режиме, когда электроустановки СЭС СТК отключены и электроснабжение переменным током осуществляется от внешних сетей, а постоянным током – от управляемых выпрямителей либо кратковременно – от ХИТ, наиболее

опасной угрозой является несанкционированное их включение путем подачи на АРМ пункта управления СЭС СТК ложного (искаженного) сигнала.

В автономном режиме, когда электроустановки СЭС СТК включены, возможно, нештатное их выключение внедрением внутренних помех путем формирования ложных эталонных параметров и воздействия компьютерных атак на сигналы управления СЭС СТК.

Объектом защиты информации о сигналах управления СЭС СТК являются информационные процессы, формализуемые в виде:

- данных, поступающих от датчиков контроля величины напряжения, подаваемого от систем внешнего электроснабжения, в дежурном режиме работы СЭС СТК;

- данных, получаемых с АРМ-управления СЭС СТК и передаваемых на него по различным каналам связи, обеспечивающим функционирование ее в дежурном и автономном режимах работы;

- данных, доставляемых от датчиков текущего и эталонного контроля параметров напряжения, частоты тока и других параметров жизнеобеспечения СЭС СТК при работе ее в автономном режиме.

Надежное функционирование СЭС СТК в условиях потенциальной возможности воздействия компьютерных атак обеспечивается разработкой многоуровневой системы информационных процессов, реализуемой на следующих уровнях (см. рис. 1):

1-й уровень: организация мероприятий по недопущению посторонних лиц к элементам СЭС СТК в дежурном режиме и при техническом регламентированном ее обслуживании;

2-й уровень: установка программно-технических средств (ПТС) разграничения доступа к информационным сигналам управления СЭС СТК при передаче их по технологическим и другим каналам связи во всех режимах работы (средства защиты от несанкционированного доступа);

3-й уровень: дистанционный контроль с пункта управления микропроцессорных устройств и адаптеров на предмет выявления недекларированных возможностей;

4-й уровень: внедрение программно-технических средств защиты цифровых информационных сигналов от компьютерных атак на программное обеспечение, цифровое оборудование и базы данных эталонных контролируемых параметров (средства обнаружения компьютерных атак);

5-й уровень: настройка средств защиты цифровых информационных сигналов в операционной системе, а также средств защиты от воздействия компьютерных вирусов при работе СЭС СТК во всех режимах работы.

При реализации уровней защиты информационных процессов управления СЭС СТК контролируемыми параметрами защиты информации в системе являются: допустимые значения напряжения, частоты тока, мощности и других параметров жизнедеятельности системы.

Особо важно при обеспечении безопасности информационных процессов осуществлять динамическую проверку текущего значения контролируемого напряжения, подаваемого на СЭС СТК от внешней системы электроснабжения в дежурном и штатном режимах. Его можно уменьшить или увеличить до критически недопустимого значения несанкционированным воздействием по различным каналам связи и тем самым преждевременно включать или выключать СЭС СТК.

В автономном режиме работы СЭС СТК важную роль в защите сигналов управления играют контролируемые эталонные параметры, снимаемые с базы датчиков. Несанкционированное воздействие компьютерными атаками на них по ка-

налам связи также приводит к частичному либо полному нарушению режима работы СЭС СТК.

Таким образом, угроза воздействия компьютерных атак на цифровую информацию о сигналах управления СЭС СТК при работе ее в различных режимах является несанкционированное воздействие на контролируемые параметры жизнедеятельности системы, приводящее к нештатному включению или выключению СЭС СТК.

Поэтому при разработке средств защиты информационных процессов управления СЭС СТК целесообразно учитывать в комплексе возможные угрозы внешнего и внутреннего воздействия на информацию, программы и цифровое оборудование, используемые для управления СЭС СТК.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. http://inopressa.ru/article/08Apr2009/wsj/spy.html.

Климов С.М.
г. Юбилейный Московской области, ЗАО «ЭКА»

Лозунг: Безопасность и защита информационных систем

ТЕХНОЛОГИЯ ИСПЫТАНИЙ КРИТИЧЕСКИ ВАЖНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ В УСЛОВИЯХ КОМПЬЮТЕРНЫХ АТАК

В статье рассматривается технология испытаний программных средств критически важных информационных систем (КВИС) в условиях воздействия компьютерных атак. Реализация предложенной технологии испытаний обеспечит возможность реализации в КВИС эффективных мер противодействия компьютерным атакам.

Критически важная информационная система, программные средства, компьютерная атака, автоматизированное рабочее место, устойчивость функционирования, тестирование и верификация.

В настоящее время существует потенциальная угроза нарушения функционирования программных средств критически важных информационных систем (КВИС) при массированном воздействии компьютерных атак на их уязвимости. Применяемые информационные технологии и традиционные средства защиты информации не обеспечивают необходимого уровня защищенности и устойчивости функционирования.

Основными причинами опасности нарушения функционирования программных средств КВИС являются:

1. Появление новых уязвимостей в КВИС в связи с их существенной модернизацией и оснащением новыми информационными технологиями и цифровым коммуникационным оборудованием, как правило, зарубежного производства или с зарубежной элементной базой.

2. Создание программно-аппаратных средств, реализующих целенаправленные компьютерные атаки на КВИС, приводящие к их интеллектуальному выводу из строя.

3. Отсутствие нормативной базы, методов и средств испытаний и контроля реальной защищённости и устойчивости функционирования КВИС в условиях воздействия компьютерных атак.

Последствиями нарушения устойчивости функционирования КВИС при воздействии компьютерных атак может быть полная потеря управляемости в реальном масштабе времени и компрометация информации у потребителя.

Тенденция развития КВИС характеризуется тем, что осуществляется переход от «клиент-серверной» структуры КВИС к «сервис-ориентированной» глобальной информационной сети. К особенностям перспективных КВИС относятся:

1. Сетецентрическое построение и цифровизация КВИС.

2. Децентрализованный доступ «в любое время, в любом месте» к информационным ресурсам КВИС в соответствии с правилами разграничения доступа.

3. Новые уязвимости в программных средствах КВИС (потенциальные точки несанкционированного доступа).

Особенности функционирования этих систем, представляющих собой «цифровую матрицу» с уязвимыми местами, которые могут быть подвержены массированному воздействию компьютерных атак, обусловили дополнительные требования к обеспечению устойчивости функционирования в условиях компьютерных атак и необходимость разработки соответствующих средств испытаний.

Учитывая то, что достигнутый уровень защищенности КВИС оценивался по устаревшим относительно современных требований нормативным документам и не проверялся в условиях массированных компьютерных атак, разработка технологий испытаний для автоматизированного тестирования и оценки устойчивости функционирования программных средств является актуальной.

Тестирование, испытания и оценка устойчивости функционирования КВИС в условиях компьютерных атак являются важнейшими этапами их жизненного цикла, по результатам которых появляется возможность принять объективное решение о реальном уровне защищенности и устойчивости функционирования программных средств.

Под испытанием программных средств КВИС понимается процесс проверки соответствия реальных характеристик КВИС заданным требованиям на установленном интервале времени в условиях моделирования компьютерных атак при выполнении технологических циклов управления.

Для обеспечения тестирования и оценки устойчивости функционирования КВИС в условиях компьютерных атак необходимо решить следующие задачи:

- определить технические требования к тестированию и оценке устойчивости функционирования КВИС в условиях компьютерных атак;

- разработать методики тестирования и оценки устойчивости функционирования КВИС в условиях компьютерных атак;

- разработать средства функционального тестирования и верификации КВИС в условиях компьютерных атак.

Анализ зарубежного опыта испытаний КВИС в условиях компьютерных атак показывает, что в настоящее время в США и Евросоюзе принимаются активные меры по обеспечению качества аппаратно-программных средств и контролю характеристик защищенности на основе комплексного тестирования этих средств в условиях компьютерных атак.

Существующая система испытаний программных средств КВИС реализуется:

- в рамках комплекса испытаний на соответствие требованиям Единой системы программной документации (ЕСПД), требованиям к качеству программных средств ГОСТ-28195-89, ГОСТ Р ИСО/МЭК 12119-2000 и организуется в соответствии с ГОСТ 34.603-92, ГОСТ Р 51189-98, ГОСТ РВ 15.210-78, ГОСТ РВ 15.211-78;

- в рамках системы сертификации средств защиты информации по требованиям информационной безопасности, в которой программные средства проверяются

на соответствие уровню защиты от несанкционированного доступа к информации и по уровню контроля отсутствия недекларированных возможностей.

Однако нормативные документы, лежащие в основе этой системы испытаний, устарели и не учитывают особенности современных и перспективных КВИС, отсутствуют сертифицированные инструментальные средства и стенды для испытаний программных средств в условиях компьютерных атак.

Модель уязвимостей вычислительных сетей КВИС должна быть основана на 7-уровневой эталонной модели взаимодействия открытых систем. Уязвимые места в КВИС представляют собой санкционированные и несанкционированные точки удаленного доступа, ошибки в программах и администрировании информационной безопасности, неправильные настройки цифрового коммуникационного оборудования и человеческий фактор, реализуемый через потенциального внутреннего нарушителя с полномочиями штатного оператора. Кроме того, в программно-аппаратных средствах потенциально существуют не декларированные возможности или «закладки». Классификацию уязвимостей целесообразно проводить на основе анализа возможностей доступа средств реализации компьютерных атак к сетевым сервисам ЭМ ВОС и нарушения установленного порядка информационного взаимодействия абонентов по протоколам передачи данных.

Технология испытаний программных средств КВИС предусматривает решение двух основных задач (рис. 1):

1. Испытания самих программных средств на соответствие технических характеристик образца установленным требованиям.

2. Испытания средств противодействия компьютерным атакам совместно с программными средствами в реальных условиях применения.

В ходе испытаний взаимосвязано рассматриваются преднамеренные и непреднамеренные компьютерные атаки.

В формах традиционных промышленных испытаний добавляются научно-технические эксперименты, компьютерные игры и компьютерные учения.

В документации испытаний: сценарий и технический паспорт компьютерных атак.

Инструментальные средства испытаний практически все новые:

– сама испытательная база стендового полигона;
– средства сканирования сети;
– средства моделирования компьютерных атак;
– средства предупреждения и обнаружения компьютерных атак;
– средства оценки эффективности противодействия компьютерным атакам;
– средства тестирования АРМ КВИС.

Пять дополнительных испытаний программных средств КВИС в условиях компьютерных атак приведены на рис. 1.

В целом технология испытаний КВИС в условиях компьютерных атак основана на создании стендового полигона оценки защищенности и устойчивости функционирования КВИС по принципу «электронного» полигона на основе распределенной структуры стендов, объединённых вычислительной сетью. Стендовый полигон позволяет натурно моделировать КВИС, тестировать программные средства и проводить оценку защищённости устойчивости их функционирования в условиях моделирования воздействий компьютерных атак.

В интересах выработки адекватных мер по обеспечению безопасного функционирования КВИС в едином информационном пространстве в условиях воздействия компьютерных атак и отработки согласованных действий по нейтрализации этой угрозы информационной безопасности предлагается предусмотреть такую форму испытаний программных средств КВИС, как компьютерные учения. На

основе компьютерных учений существует возможность выработки действенного механизма принятия решений по выбору методов, моделей и средств противодействия атакам, а также по устранению уязвимостей в программных средствах.

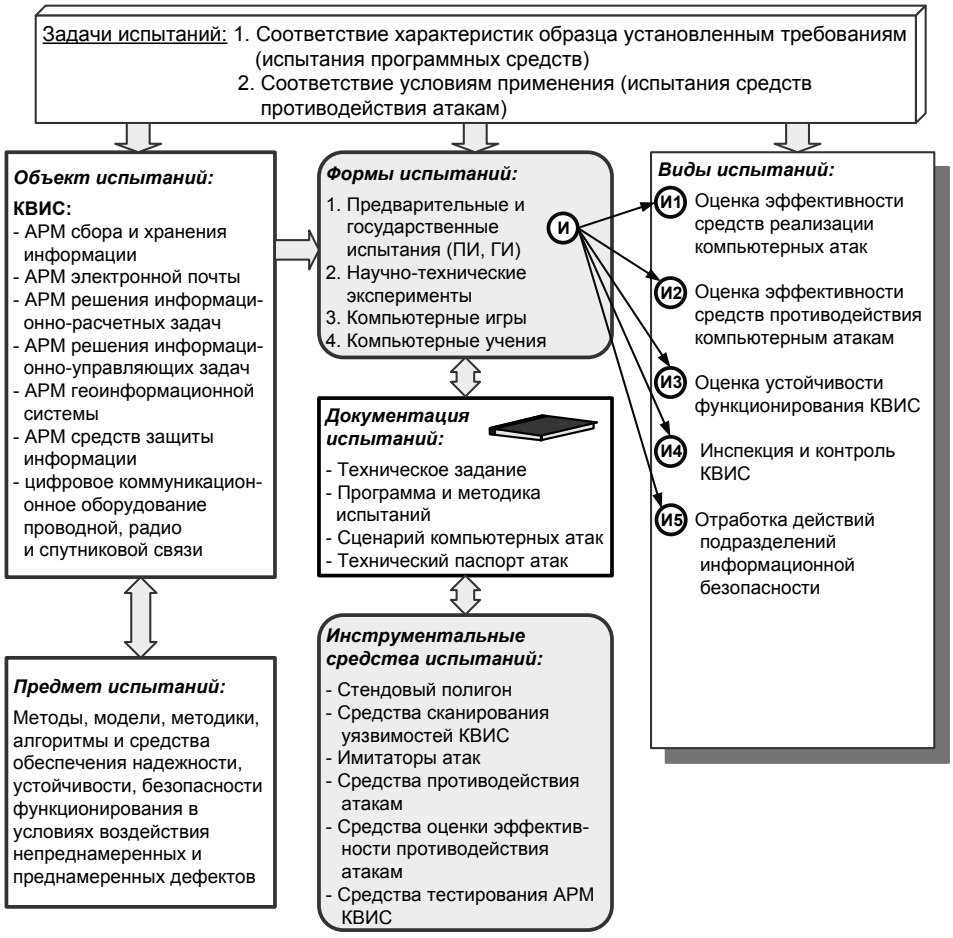


Рис. 1. Схема технологии испытаний КВИС в условиях компьютерных атак

Для проведения испытаний программных средств КВИС необходим комплекс инструментальных средств автоматизированного тестирования и верификации программ.

На рис. 2 приведена технологическая схема функционального тестирования КВИС. Средства функционального тестирования и верификации специального программного обеспечения и телекоммуникационных систем в условиях компьютерных атак являются одними из наиболее сложных комплексов программ для проведения испытаний программных средств КВИС. Проведение тестирования рассматриваемых программных средств в условиях компьютерных атак связано с необходимостью проверки и контроля значительного числа параметров и показателей. В результате проведения тестирования должны быть промоделированы условия реального применения программных средств КВИС в различных режимах функционирования, проимитированы средства программно-аппаратного воздейст-

вия и средства защиты от них, а также собраны, обработаны и наглядно представлены результаты экспериментальных исследований.

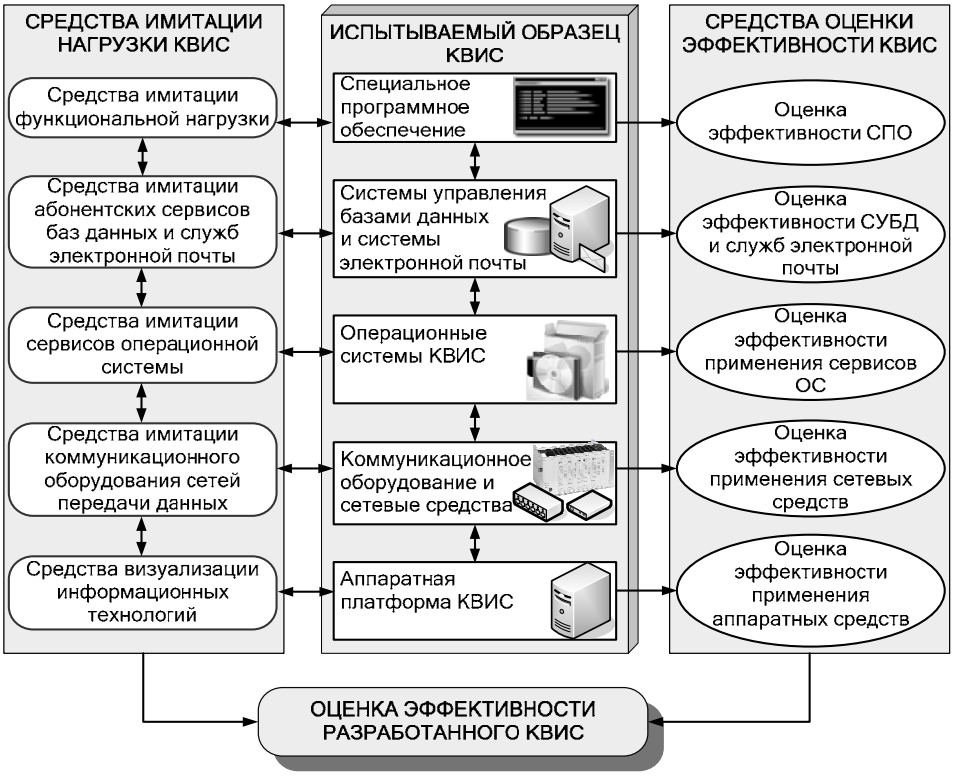


Рис. 2. Технологическая схема функционального тестирования КВИС

С целью проведения интегральной оценки характеристик КВИС при их реальном применении предлагается комплексно и взаимосвязано оценить надежность, пропускную способность и устойчивость функционирования КВИС в условиях компьютерных атак (рис. 3). При таком подходе на временном интервале выполнения технологического цикла управления КВИС можно определить полосу устойчивого функционирования, которая и характеризует реальные возможности КВИС по сохранению работоспособного состояния при воздействии преднамеренных и непреднамеренных дефектов.

Таким образом, реализация предложений по технологии испытаний КВИС в условиях компьютерных атак позволит эффективно реализовать меры противодействия угрозам нарушения функционирования КВИС при целенаправленном и массированном воздействии компьютерных атак.

Организация системы стендовых полигонов испытаний программных средств КВИС в условиях компьютерных атак и проведение компьютерных учений даёт возможность априорной и детальной оценки характеристик защищённости и устойчивости функционирования программных средств при имитации реальных условий применения на основе использования комплекса средств автоматизированного тестирования и верификации программного обеспечения.

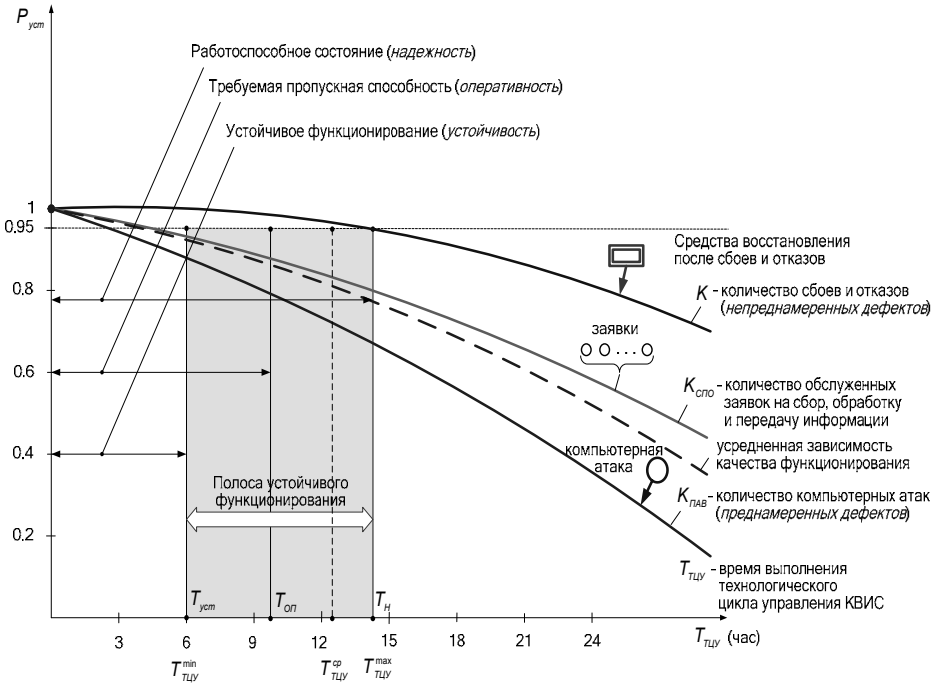


Рис. 3. Результаты комплексной оценки надежности, пропускной способности и устойчивости функционирования КВИС в условиях компьютерных атак

Рыжов Б.С.
г. Юбилейный Московской области, ЗАО «ЭКА»

Лозунг: Защищенность системы определяется самым слабым её элементом

МЕТОДОЛОГИЯ ОЦЕНКИ ЗАЩИЩЕННОСТИ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ

В статье рассматривается методология оценки защищенности, основанная на использовании методов анализа и оценки активов распределенных вычислительных сетей, уязвимостей, угроз информационной безопасности, возможных атак и целей безопасности.

Оценка защищенности, профиль защиты, распределенная вычислительная сеть, информационная безопасность.

Анализ работ, проведенных в области оценки информационной безопасности, показал, что оценка информационной безопасности проводится на основе стандарта ISO/IEC 15408 и соответствующим профилям защиты. Стандарт ISO/IEC 15408 определяет защищенность как совокупность функциональных и гарантийных требований, позволяющих реализовать систему защиты с необходимым уровнем информационной безопасности (ИБ). Методология оценки защищенности основана на использовании методов анализа и оценки активов распределенных вычислительных сетей (РВС), уязвимостей, угроз ИБ, возможных атак и целей безопасно-

сти. Атака на РВС возможна, если существует угроза Y_{Ri} использования уязвимости Y_{Zi} объекта оценки защищенности. Это приводит к риску R потери активов РВС. Снижение риска потери активов возможно при правильном выборе функциональных требований, политик безопасности РВС $П_{Bi}$ и разумных предположений $П_{Pi}$ относительно свойств объекта оценки и его среды функционирования.

В литературе отсутствуют сведения о вероятностях отдельных угроз, вероятностях атак, эффективности отдельных политик безопасности. В связи с этим для получения количественной оценки риска предлагается использовать экспертные оценки [2], основанные на использовании кластера исходов (рис. 1), представляющего собой дерево иерархий с вершинами $E1, \ldots Ei, \ldots En$. Каждая из вершин кластера исходов соответствует элементу множества значений анализируемого показателя $\{(Y_Z)_i\}, \{(Y_R)_i\}, \{\Pi_{Bi}\}, \{\Pi_{Pi}\}$.

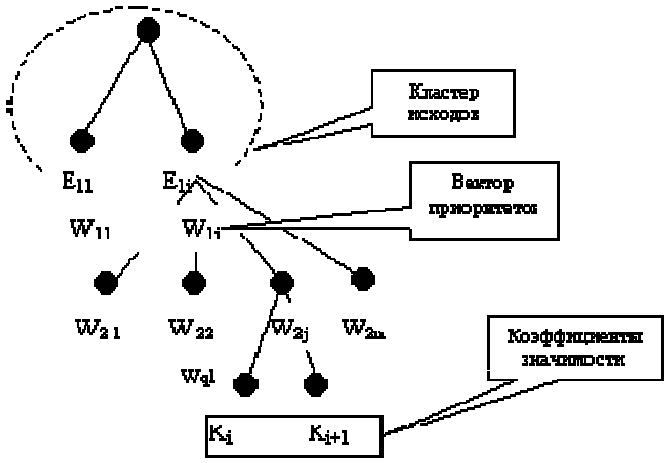


Рис. 1. Иерархия кластеров

Значимость элементов кластера определяется на основе матрицы парных сравнений [E] и вектора приоритетов W_i, определяющего ранжировку элементов кластера исходов [2]. Каждый путь в дереве иерархий соответствует элементу множества значений отдельных показателей $\{(Y_R)_i\}, \{\Pi_{Bi}\}, \{\Pi_{Pi}\}$ и для этого пути можно определить коэффициент значимости K_i как произведение значений векторов приоритетов кластеров исходов, соответствующих этому пути.

$$K_i = W_{1i} * W_{2i} * W_{3i} * \ldots * W_{gi} . \qquad (1)$$

При этом $\sum_{\forall i} K_i = 1$.

Выбор разработчиком защищенности конкретных элементов множеств $\{(Y_R)_i\}, \{\Pi_{Bi}\}, \{\Pi_{Pi}\}$ равносилен исключению из соответствующих деревьев иерархии отдельных элементов, что уменьшает суммарное значение коэффициентов значимости K_i с 1 до некоторой величины $g < 1$. Точка с координатами

$\{g_Y, g_{ПВ}, g_{ПР}\}$ определяет исходное состояние защищенности РВС. Данная точка соответствует в процессе проектирования РВС этапу «Среда безопасности» [1], когда выбраны уязвимости и соответствующие им угрозы, определены политики безопасности и сформулированы предположения.

Выбор конкретных элементов множеств $\{(Y_Z / Y_R)_i\}, \{\Pi_{Bi}\}, \{\Pi_{Pi}\}$ всегда связан с риском того, что выбранные множества не обеспечат нужной совокупности целей безопасности, вследствие чего появится возможность проведения нарушителем направленных атак и произойдет потеря активов РВС.

Условиями, способствующими риску, являются:

- недостаток, имеющейся у разработчика информации (список уязвимостей не полон, часть угроз не идентифицирована, политики безопасности в силу экономических, юридических особенностей могут быть выполнены лишь частично);
- недостаток времени и имеющихся ресурсов.

Специфика разрабатываемой оценки защищенности требует введения новых целей и разработки для них специальных функциональных требований безопасности. А это всегда связано с дополнительными и не малыми временными, ресурсными и стоимостными затратами.

Риск исходного состояния защищенности можно представить вектором с координатами $\{0, 0, 0\}$ и $\{g_Y, g_{ПВ}, g_{ПР}\}$, а мерой риска может служить длина этого вектора (рис. 2), определяемая как

$$R = \sqrt{(\{g_Y^2 + g_{ПВ}^2 + g_{ПР}^2\})} . \quad (2)$$

В данном случае R определяет верхнюю границу риска, которая априорно существует до определения целей защищенности. При выборе целей защищенности, исходя из уязвимостей, угроз, политик безопасности и предположений, разработчику по защищенности рекомендуется использовать таблицы соответствия: «Детальная политика безопасности – Цели безопасности» и таблицу соответствия «Угрозы – Атаки – Цели безопасности» из профилирующей базы знаний «CC Profiling Knowledge Base». Профилирующая база знаний включает 30 угроз, 34 детальные политики безопасности, 36 предположений и 153 цели безопасности. Методика формирования и последующей оценки защищенности основывается на утверждении: вариант защищенности включает совокупность целей защищенности, покрывающих все атаки и политики безопасности. На рис. 2 показана последовательность формирования целей защищённости для вариантов защищенности, исходя из существующих уязвимостей, угроз, атак. Для исходного состояния защищенности рассчитываются суммарный коэффициент значимости атак g_a и суммарный коэффициент значимости политик безопасности $g_{ПБ}$ (соотношения (3), (4)):

$$g_a = \sum_{i=1}^{n} K_{ai} , \quad (3)$$

$$g_{ПБ} = \sum_{j=1}^{m} K_{ПБi} . \quad (4)$$

Стандарт ISO/IEC 15408 позволяет выбрать цели безопасности, нейтрализующие возможные атаки, способствующие реализации политик безопасности и, в конечном счете, снижающие величины g_a, и $g_{\text{ПБ}}$ на величины, определяемые соотношениями (5) и (6):

$$\Delta g_a = \sum_{j=1}^{n-1} \left(\prod_{i=1}^{n} (K_{ai} * K_{\text{ц}ai1}) \right)_j , \tag{5}$$

$$\Delta g_{\text{ПБ}} = \sum_{j=1}^{n-1} \left(\prod_{i=1}^{m} (K_{ПБi} * K_{\text{ц}ПБi1}) \right)_j . \tag{6}$$

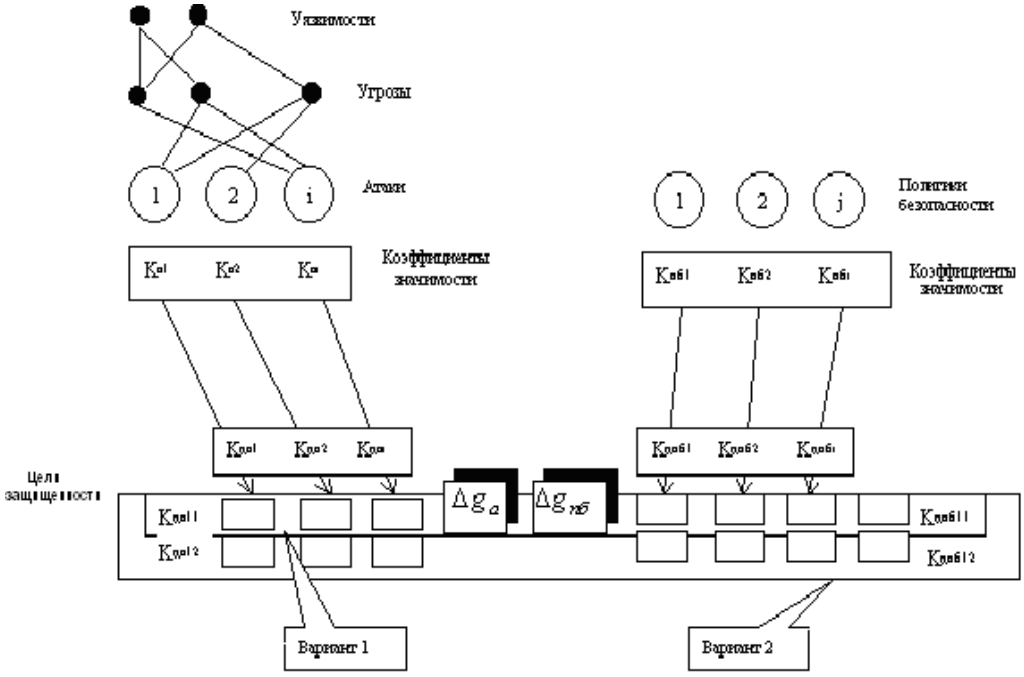


Рис. 2. Формирование вариантов защищенности РВС

Риск потерь активов для защищенности РВС можно оценить соотношением (7)

$$R_1 = \sqrt{((g_a - \Delta g_a)^2 + (g_{\text{ПБ}} - \Delta g_{ПБ})^2)} \tag{7}$$

На рис. 3 показана графическая интерпретация процесса целенаправленного выбора варианта защищенности, имеющего минимальное значение риска потери активов.

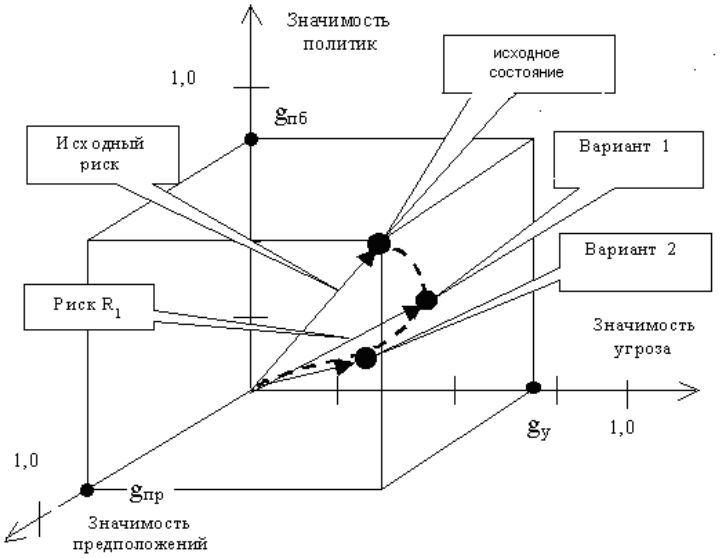


Рис. 3. Графическая интерпретации риска потери активов защищенности РВС

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ISO/IEC 15408-2. Information technology - Security techniques - Evaluation criteria for IT security - Part 2: Security functional requirements.

2. *Саати Т.* Принятие решений. Метод анализа иерархий / Пер. с англ. – М.: Радио и связь, 1989.

Апшацева Л.Р.
Научный руководитель: к.т.н., доцент Брюхомицкий Ю.А.
г. Нальчик, ИИПРУ КБНЦ РАН

Лозунг: Доверяй, но проверяй

КОНТРИНСАЙДЕРСКИЙ КЛАВИАТУРНЫЙ МОНИТОРИНГ АВТОМАТИЗИРОВАННЫХ ИНФОРМАЦИОННЫХ И УПРАВЛЯЮЩИХ СИСТЕМ КРИТИЧЕСКИХ ПРИЛОЖЕНИЙ

С целью предотвращения негативных последствий от возможной реализации угрозы злонамеренных воздействий на автоматизированные информационные и управляющие системы критических приложений со стороны внутренних злоумышленников – инсайдеров – обсуждается возможность организации и ведения скрытного клавиатурного мониторинга операторов таких систем. Рассматриваются возможности и методы повышения точности систем скрытного клавиатурного мониторинга, основанные на использовании многосвязного представления клавиатурных параметров операторов.

Информационная безопасность, операторы автоматизированных информационных и управляющих систем критических приложений, инсайдеры, скрытный клавиатурный мониторинг, биометрическая идентификация, классификация, точность представления клавиатурных параметров, методические ошибки.

Автоматизированные информационные и управляющие системы критических приложений (АИУС КП), используемые в оборонном комплексе, центрах управления полетами космических объектов, крупных энергетических системах (особенно на атомных электростанциях), химических производствах, диспетчерских центрах управления авиационным и железнодорожным транспортом, характеризуются чрезвычайно высокой ценой ошибки и, очевидно, должны обладать предельно высокой степенью надежности и информационной защиты. Одна из наиболее серьезных угроз нарушения информационной безопасности АИУС КП исходит от действий злоумышленников, которые могут быть как внешними, так и внутренними. Современные АИУС КП, как правило, имеют достаточно развитые системы информационной защиты по всем трем составляющим ИТ-безопасности: конфиденциальности, целостности и доступности. Вместе с тем преимущественное внимание традиционно уделяется информационной защите от внешних злоумышленников, когда большая часть средств, выделяемых на обеспечение ИТ-безопасности АИУС КП, направляются на защиту уязвимых точек внешнего периметра безопасности, и гораздо меньше внимания уделяется угрозам, исходящих изнутри, от собственных сотрудников – инсайдеров. Между тем инсайдеры (от англ. *inside* – внутри), находясь внутри организации, эксплуатирующей АИУС КП, легально наделены определенными (в том числе и очень высокими) полномочиями, пользуясь которыми гораздо проще получить доступ к внутрисистемной информации, чем каким бы то ни было лицам со стороны.

По усредненному мнению зарубежных и отечественных специалистов в области ИТ-безопасности наиболее опасной угрозой для АИУС КП сегодня являются кража информации инсайдерами (64 %). На самом деле инсайдерский фактор еще более весомый, так как он влияет фактически и на большую часть остальных угроз (вредоносные программы, хакерские атаки, халатность сотрудников, спам и т.д.). Эффективные вредоносные программы невозможно создать без помощи инсайдеров, поскольку такие программы должны быть исключительно целенаправленными, учитывающими эксклюзивные «тонкости» поражаемых фрагментов конкретного объекта информатизации. Практически все успешные и широко известные хакерские атаки так или иначе были осуществлены с участием инсайдеров. Халатность сотрудников является, с одной стороны, благоприятной средой для эффективной деятельности инсайдеров, с другой – благоприятной базой для вербовки новых инсайдеров, а также появления инсайдеров-инициативников. Даже спам, имея чисто внешнее происхождение, может быть использован инсайдерами в своих целях, например, для маскировки их деятельности [1].

Для АИУС КП, в которых цена ошибки операторов оказывается чрезвычайно высокой, актуальным является, в частности, решение следующих задач в области информационной безопасности:

- скрытного выявления в реальном масштабе времени факта подмены инсайдером штатного легального оператора АИУС КП, открывшего доступ в систему и затем покинувшего свое рабочее место;
- скрытного выявления операторов-инсайдеров, совершающих злоупотребления и атаки на АИУС КП, а также действия, выходящие за рамки их полномочий;
- проведения тестирования, в том числе скрытного, для выявления операторов-инсайдеров АИУС КП, заподозренных в злоумышленной деятельности;
- проведения скрытного тестирования в том числе скрытного, операторов АИУС КП для решения вопроса о их допуске к работе.

Решение указанных задач возможно на основе использовании методов непрерывной скрытной биометрической идентификации операторов АИУС КП по их клавиатурному почерку (скрытный клавиатурный мониторинг – СКМ).

Как известно, биометрическая идентификация оператора информационной системы по клавиатурному почерку возможна двумя способами: по набору ключевой фразы и по набору произвольного текста (текстонезависимая идентификация).

Большинство имеющихся на рынке систем биометрической идентификации оператора по клавиатурному почерку осуществляют аутентификацию по фиксированной ключевой фразе, в качестве которой используется пароль или имя входа в систему (например, система BioPassword компании Net Nanny Software International). Единственным продуктом, позволяющим проводить текстонезависимую идентификацию (СКМ) операторов информационных систем, является система Complex Electronic Signature Lock (CESL) компании Electronic Signature Lock Corporation.

Суть СКМ заключается в проведении скрытного анализа клавиатурного почерка работающего оператора и его сравнении с эталоном, предварительно созданным и зарегистрированным для данного оператора.

Известные методы СКМ сводятся к прямому измерению параметров клавиатурного ввода, их усреднению и хранению в качестве эталонов [2]. Далее осуществляется непрерывная (периодическая) идентификация (классификация) личности работающего оператора по принципу «свой - чужой», которая заключается в сравнении текущих клавиатурных параметров с эталоном, ранее зарегистрированным для данного оператора. Недостатком этих методов является низкая конечная точность классификации, которая обусловлена двумя основными группами ошибок:

- методическими ошибками, обусловленными недостаточной информативностью принятого представления клавиатурных параметров операторов;
- собственными методическими ошибками используемых классификаторов «свой - чужой».

Следует отметить, что способы построения классификаторов «свой - чужой», а, следовательно, и их собственные методические ошибки в значительной степени зависят от принятого представления клавиатурных параметров операторов. Поэтому стремление повысить конечную точность процедуры классификации «свой - чужой» в системах СКМ связано, в первую очередь, с необходимостью повышения точности представления клавиатурных параметров операторов.

В работах [3-4] впервые были предложены методы повышения точности систем СКМ, основанные на так называемом многосвязном представлении клавиатурных параметров операторов. В настоящей работе обсуждаются преимущества этих методов и возможности их использования в области инсайдерологии.

Для СКМ русскоязычных текстов временные параметры сильно зависят от «вхождения» символов в контекст. Другими словами, скорость набора разных фраз одинаковой длины, при условии, что каждая из них содержит одинаковое число одноименных символов, но расположенных по-разному, будет различной.

Содержательно это можно объяснить несколькими причинами.

- Объективным фактором, влияющим на скорость клавиатурного набора, является принятая раскладка символов на клавиатуре. Суммарное время, затрачиваемое на перенос пальцев из одной позиции в другую при клавиатурном наборе различных последовательностей одного и того же состава символов, будет существенно разниться.
- Субъективным фактором является контекст. Осмысленные для данного оператора слова и фразы набираются им быстрее.
- Субъективным фактором является уровень «клавиатурного» знакомства с данной предметной областью. Слова и фразы, которые оператору приходится набирать чаще, набираются им быстрее.

- Объективный фактор, отмеченный в п. 1, выступает одновременно и субъективным фактором. Определенные последовательности символов и слов для данной личности могут быть более «благозвучными». Это можно объяснить тем, что набору некоторого слова, словосочетания или фразы на клавиатуре предшествует «проигрывание» этой конструкции в мозге оператора. От того, насколько более «благозвучно» для данного оператора такое проигрывание, зависит как общее время набора фразы, так и раскладка этого времени по входящим в нее символам и паузам между ними.

На самом деле все отмеченные причины выступают в тесной взаимосвязи и их разделение полезно лишь с методической точки зрения.

Основываясь на приведенном перечне причин, можно предположить, что общие показатели скорости клавиатурного набора при прочих равных условиях оказываются выше, если:

- сумма расстояний между последовательно удерживаемыми клавишами меньшая;
- набираемые слова и фразы являются осмысленными для данного оператора;
- набираемые слова и фразы являются составными понятиями предметной области, хорошо знакомой данному оператору;
- набираемые слова и фразы являются более «благозвучными» для восприятия данным оператором.

На основании вышеизложенного, можно констатировать следующее:

1. Низкая точность используемых систем СКМ объясняется, прежде всего, большой методической погрешностью представления клавиатурных параметров.
2. В части ошибок описания и представления процесса СКМ, большая методическая погрешность является следствием интегрального подхода к регистрации временных параметров СКМ, при котором разброс регистрируемых временных параметров клавиатурного почерка фиксируется независимо от «вхождения» символов в определенные сочетания с другими символами.

Вывод. Точность систем СКМ может быть повышена, в первую очередь, за счет снижения методической погрешности представления клавиатурных параметров. Для этого предлагается регистрировать временные параметры клавиатурного набора не только отдельных символов и их двухбуквенных сочетаний, но и (по возможности) более длинных цепочек.

Пример. В процессе клавиатурного набора с использованием двух клавиш А и Б при регистрации времен их удержания кроме времен $\tau_А$ и $\tau_Б$ (классический метод), можно анализировать времена $\tau_{А/А}$, $\tau_{А/Б}$, $\tau_{Б/Б}$, $\tau_{Б/А}$. Указанные времена определяются текущим вхождением. То есть время удержания данной клавиши зависит от того, какой была предшествующая клавиша в наборе. При этом есть основания полагать, что времена удержаний клавиш А и Б во всех указанных сочетаниях будет существенно различными. При накоплении достаточной статистики, скорее всего, будет наблюдаться следующая картина:

- время удержания клавиши А (параметр τА) будет иметь не один, а два выраженных центра распределения, соответствующих двум вхождениям клавиши А (τА/А, τА/Б);
- время удержания клавиши Б (параметр τБ) будет иметь также два выраженных центра распределения, соответствующих двум вхождениям клавиши Б (τ Б/Б, τ Б/А);

- указанные распределения будут близки к нормальным.

Дальнейшее развитие этого подхода в направлении разработки соответствующих методов, алгоритмов, программных моделей и методик позволяет рассчитывать на эффективное решение задачи контринсайдерского клавиатурного мониторинга АИУС КП.

Работа выполнена при поддержке гранта РФФИ № 08-07-00117-а

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Бородакий Ю.В., Добродеев А.Ю., Пальчун Б.П., Болдина М.Н.* Инсайдерология – наука о нелегитимности в компьютерной инфосфере // Изв. ЮФУ. Технические науки. – 2008. – №8. – С. 55 – 64.

2. *Широчин В.П., Кулик А.В., Марченко В.В.* Динамическая аутентификация на основе анализа клавиатурного почерка. – http://www.masters.donntu.edu.ua/2002/fvti/aslamov/files/bio_autentification.htm.

3. *Брюхомицкий Ю.А., Казарин М.Н.* Система скрытного клавиатурного мониторинга / Известия ТРТУ. Специальный выпуск. Технические науки. Материалы LII научно-технической конференции. – Таганрог: Изд-во ТРТУ, 2006. № 9(64). – С. 153-154.

4. *Брюхомицкий Ю.А., Казарин М.Н.* Методы многосвязного представления клавиатурного почерка / Материалы III Международной конференции «Нелокальные краевые задачи и родственные проблемы математической биологии, информатики и физики. – Нальчик, 5-8 декабря 2006 г. – С. 68-69.

Васильев Е.С.
Научный руководитель: д.т.н. Шепитько Г.Е.
г. Москва, МФЮА

ПРОБЛЕМЫ И МЕТОД ОПТИМИЗАЦИИ СОВРЕМЕННЫХ ILD&P СИСТЕМ

В работе описываются правовые, этические и экономические аспекты применения ILD&P в секторе коммерческих предприятий. Основной целью исследования является выработка алгоритмов оптимизации систем управления инсайдерскими рисками и создание статистической базы для дальнейшего моделирования внутренних угроз безопасности.

Управление рисками, утечка информации, инсайдер, внутренний нарушитель.

Распространенность угроз информации привела к появлению на рынке массы компаний-разработчиков (вендоров), предлагающих готовые решения проблем безопасности того или иного типа. К концу девяностых годов коньюктура рынка стала меняться – вендоры в поисках новой точки приложения своих производственных ресурсов выбрали инсайдерские угрозы как новый тренд в сфере информационной безопасности.

Маркетинговый потенциал новой ниши, безусловно, подтвержден растущим количеством инсайдерских инцидентов [1,2].

Не лишним будет отметить, что российские разработчики и интеграторы, имея большой опыт и квалифицированные кадры, при постановке проблемы ориентируются, в основном, на западную аналитику тенденций информационной безопасности, зачастую пренебрегая отечественными реалиями. Причиной тому служит отсутствие механизмов централизованного сбора статистики компьютер-

ных нарушений и утечки информации, а также недостаточность регулирующих норм в этой сфере.

Текущая дискуссия вендоров и системных интеграторов сводится к обсуждению программно-аппаратных каналов утечки информации из компьютерных систем – на данный момент итогом этой дискуссии стало принятие бизнес-сообществом класса специализированных ILD&P-продуктов [3] (Information Leakage Detection and Prevention – решения для обнаружения и предотвращения утечек информации).

Продукты класса ILD&P разделяются на две категории. В соответствии с распространенными каналами утечки информации – это сетевые решения, контролирующие передачу информации на уровне локальной сети предприятия, и решения на базе настольных ПК.

И сетевые, и настольные средства используют методы контентной фильтрации по контрольным суммам, вхождению слов или цифровым отпечаткам для отслеживания перемещения индексированных документов. Эти методы несовершенны и при больших объемах передаваемой информации велико количество ложных срабатываний [4]. Для эффективной оперативной аналитики при этом требуется постоянное дежурство офицера безопасности, что означает дополнительные финансовые затраты. Кроме того, возникают проблемы морально-этического и правового характера в случаях, когда фильтр срабатывает на персональную информацию пользователя, и уполномоченный сотрудник обязан просмотреть её. Такие действия могут трактоваться как нарушение ст. 23 Конституции РФ и считаться перлюстрацией согласно ст. 138 УК РФ. Радикальным подходом в таких ситуациях является запрет на использование информационных систем компании в личных целях и включение в трудовой договор соответствующего пункта, при подписании которого сотрудник соглашается на мониторинг процессов, предусмотренных его рабочими полномочиями. Но такой подход не снимает правовой проблемы, так как сотрудник, использующий компьютерную сеть компании в личных целях хоть и считается нарушителем режима компании, но вынуждает офицера безопасности идти на непреднамеренное нарушение УК РФ. Кроме того, такие требования и ограничения для всех сотрудников снижают лояльность к компании.

Оптимальным вариантом решения проблемы представляется внедрение системы гибких индивидуальных политик и процедур безопасности, основанных на оценках дисциплинированности, лояльности и грамотности персонала.

Автором предлагаются базовые принципы построения системы управления внутренними рисками, основными элементами которой являются методика их оценки и система сигнатурного мониторинга.

Определение допуска пользователей к группе информации производится согласно рабочей необходимости с использованием матрицы доступа.

Для получения условного балла недоверия предлагается использовать данные теневого сигнатурного мониторинга действий пользователя АРМ в течение определенного срока. Для каждого нарушения допущенного пользователем в период мониторинга устанавливается коэффициент, основанный на статистических данных. Основанием для применения инструментария теневого аудита и контекстного анализа будет высокий бал недоверия к сотруднику при обращении к информации, соответствующей критичности.

Таким образом, применение изложенной методики может снизить поток информации, необходимой для исполнения процедуры безопасности, а соответственно, снизить требования к аналитике и количество ошибок второго рода, изначально применив более лояльные правила для групп незначительного риска.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Исследование Института компьютерной безопасности (США) «2007 CSI Computer Crime and Security Survey».

2. http://www.securitylab.ru/news/354109.phpИсследование компании Secure Computing.

3. Статья IDC «White Paper - Combating Insider Threats».

4. *Волков Д.* «Спецобзор: DLP-системы на страже корпоративных данных». Ежемесячный журнал ITСпец №11 (Ноябрь 2008).

5. *Васильев Е.С.* «Оценка риска инсайдерских угроз на предприятии». / Труды XVI Международной конференции «Проблемы управления безопасностью сложных систем». – М.: ИЦ РГГУ, 2008.

Михеев В.А.
Научный руководитель: д.т.н., профессор Верба В.С.
г. Москва, ОАО «Концерн «Вега»

Лозунг: Избыточность – основной метод повышения надежности многофункциональной информационной системы

АНАЛИЗ МЕТОДОВ ПОВЫШЕНИЯ НАДЕЖНОСТИ МНОГОФУНКЦИОНАЛЬНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Рассмотрена проблема повышения надежности многофункциональной информационной системы холдинга вследствие возникновения угроз потери или модификации (искажения) информации, циркулирующей в системе. Проведен анализ существующих подходов и методов повышения надежности комплекса программных и технических средств. Обоснована необходимость использования методов повышения надежности в зависимости от этапа жизненного цикла многофункциональной информационной системы. На основе этих методов выработаны и проанализированы направления и принципы повышения надежности функционирования МИС.

Информационная система, надежность, отказ, избыточность.

Многофункциональная территориально распределенная информационная система (МИС) – взаимосвязанная совокупность кампусных ЛВС и сетей передачи данных, предназначенная для создания единого защищенного интегрированного информационного пространства информационных ресурсов предприятий холдинга. Многофункциональность системы определяется наличием в ней целого ряда функциональных систем и подсистем, а также комплекса технических (аппаратных) и программных средств.

По мере развития МИС и усложнения используемых программно-аппаратных средств происходит повышение уязвимости МИС. Основными факторами, способствующими повышению уязвимости, являются:

- увеличение объемов информации, хранимой, обрабатываемой и передаваемой в МИС;
- расширение круга пользователей, имеющих непосредственный доступ в соответствии с ролями и полномочиями доступа к ресурсам МИС;
- сосредоточение в единой базе данных разнородной информации;

- усложнение режимов функционирования МИС, связанных с непрерывной обработкой данных.

Таким образом, возникает угроза потери или модификации (искажения) информации, циркулирующей в МИС. В связи с этим методы обеспечения и повышения надежности приобретают ключевой характер на всех этапах жизненного цикла системы [1].

В соответствии с [2] надежность трактуется как свойство объекта сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях применения, технического обслуживания, ремонта, хранения и транспортирования.

Под надежностью МИС понимается её защищенность от случайных или преднамеренных вмешательств в нормальный процесс её функционирования, выражающаяся в потере или модификации (искажении) информации.

К угрозе потери или модификации (искажения) информации приводят сбои и отказы в работе МИС.

Классифицировать отказы можно в зависимости от характера и особенностей, от момента возникновения следующим образом [2]:

По характеру изменения параметра до момента возникновения отказа:

- внезапный отказ;
- постепенный отказ.

По связи с другими отказами:

- независимый отказ;
- зависимый отказ.

По возможности последующего использования после возникновения отказа:

- полный отказ;
- частичный отказ.

По характеру устранения отказа:

- устойчивый отказ;
- самоустраняющийся отказ (сбой или перемежающийся отказ).

По наличию внешних проявлений:

- очевидный (явный) отказ;
- скрытый (неявный) отказ.

По причине возникновения:

- конструкционный отказ;
- технологический отказ;
- эксплуатационный отказ.

По природе происхождения:

- естественный отказ;
- искусственный отказ (вызываемый намеренно).

По времени возникновения отказов:

- отказ при испытаниях;
- отказ периода приработки;
- отказ периода нормальной эксплуатации;
- отказ последнего периода эксплуатации.

Необходимо отметить, что отказы относят к конструкционным, технологическим или эксплуатационным с целью установления, на каком этапе жизненного цикла МИС следует провести мероприятия, предупреждающие и устраняющие причины отказов.

Таким образом, применительно к рассматриваемой системе, отказы можно разбить на две группы [3]:

- нарушения в работе комплекса технических средств (КТС) – аппаратный сбой;
- нарушения в работе комплекса программных средств (КПС) – программный сбой.

Методы повышения надежности КТС. В настоящее время известно множество различных методов повышения надежности КТС информационных систем, однако не все из них могут найти применение в рамках рассматриваемой системы.

Исходя из этого, методы повышения надежности КТС можно ограничить разбивкой на четыре группы:

- введение избыточности КТС (внутриэлементной, структурной, информационной, алгоритмической);
- применение более надежных компонентов;
- улучшение условий эксплуатации;
- организация интенсивного профилактического обслуживания как всей системы, так и отдельных ее элементов.

Применение второй группы методов не позволит обеспечить необходимый уровень надежности МИС вследствие того, что он требует применения фактически нового КТС (имеющего более высокие показателями надежности) и значительного увеличения финансовых затрат.

Третья группа методов предполагает приведение условий эксплуатации КТС в соответствие с требованиями, при которых гарантируются паспортные данные технических средств по надежности. Дальнейшее же улучшение условий эксплуатации не может существенно повысить надежность функционирования КТС.

Очевидно, что методы первой и второй групп необходимо реализовывать на этапе разработки МИС, а третьей и четвертой – на этапе эксплуатации.

Основным методом повышения надежности КТС МИС является применение структурной избыточности. Другие виды избыточности (временная, алгоритмическая и т.д.), также способные повышать надежность КТС, но по своим свойствам во многом сходны со структурной.

Структурная избыточность вводится в комплекс технических средств с целью улучшения надежностных характеристик и показателей. В настоящее время известно много методов целенаправленного введения структурной избыточности (горячее и холодное резервирование, мажоритарные структуры и т.п.), что позволяет обеспечить требуемый уровень надежности системы на основе применения ненадежных элементов.

Методы повышения надежности КПС. Надежность МИС определяется не только отказами КТС, но и отказами КПС, вызываемыми ошибками в программах. Если отказы КТС зависят от времени и не зависят от обрабатываемой информации, то отказы КПС, наоборот, не зависят от времени, зато зависят от обрабатываемой входной информации, а также от текущего состояния системы.

Оценка надежности КПС может быть получена непосредственно на основе свойств используемого в рамках МИС программного обеспечения (ПО).

Для сокращения до минимума количества ошибок, встречающихся в ПО, необходимо знание совокупности факторов, определяющих надежность ПО, которые можно разделить на три группы [4,5]:

- общие факторы;
- факторы, связанные с разработкой ПО;
- эксплуатационные факторы.

К общим факторам относятся:

- процедуры управления разработкой ПО;
- подготовка управления разработкой ПО;
- архитектура вычислительной системы;
- языки программирования.

К факторам, связанным с разработкой ПО, относятся:

- конструктивные (разряды и стоимость разрабатываемой системы, структура построения программы, наличие опыта разработки, степень выполнения последовательности работ);
- технологические (техника программирования);
- организационные (управление надежностью в процессе разработки, степень обучения и информативности персонала, микроклимат в группе, выполняющей разработку, временные ограничения на выполнение работ).

К эксплуатационным факторам относятся:

- полнота и качество эксплуатационной документации;
- степень адаптации документации;
- простота изучения и использования системы ПО;
- качество обучения пользователей, ответственных за эксплуатацию ПО;
- степень выполнения стандартов на эксплуатацию ПО.

Для достижения заданного уровня надежности КПС необходимо [6]:

- избегать ошибок, возникающих в процессе создания ПО;
- обнаруживать ошибки;
- исправлять допущенные ошибки;
- предусматривать допуск ошибок.

К средствам избегания ошибок в процессе проектирования относятся такие, целью которых является предупреждение появления ошибок в программе, а именно:

- средства и приемы минимизации сложности как основной причины ошибок трансляции;
- средства и методы совершенствования информационных связей разработчиков;
- средства и приемы немедленного обнаружения и удаления ошибок трансляции после каждого ее шага, а не после завершения написания программы.

К средствам обнаружения ошибок относятся:

- организация проверки значений входных данных на известное ограничение;
- проверка на совместимость входных данных;
- введение необходимой избыточности входных данных;
- организация сравнения входных данных с некоторым набором внутренних данных.

Эффективные средства исправления ошибок пока не разработаны.

Средства допущения ошибок обеспечивают возможность функционирования ПО в случае, когда в нем присутствуют ошибки. В составе этих средств можно выделить динамическую избыточность, вспомогательные методы и изоляцию ошибок.

Очевидно, что при прочих равных условиях (квалификация разработчиков, период эксплуатации, устранение ошибок после их обнаружения и т.д.) через не-

который промежуток времени более надежными становятся ПО массового использования.

Следовательно, для повышения надежности КПС необходимо применять ПО массового использования, а разработку собственного ПО вести с учетом рассмотренной совокупности факторов, определяющих надежность ПО.

Таким образом, при минимизации применения в МИС ПО собственной разработки проблема повышения надежности КТС имеет больший приоритет по сравнению с проблемой повышения надежности КПС, а основным методом повышения надежности КТС является метод применения структурной избыточности, при этом другие рассмотренные методы повышения надежности также могут быть использованы в качестве дополненных к основному.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Михеев В.А.* Методология разработки и аттестации автоматизированных систем в защищенном исполнении // Материалы IX Международной научно-практической конференции «Информационная безопасность-2007» (Часть I). – Таганрог: Изд-во ТТИ ЮФУ, 2007.

2. ГОСТ 27.002-89 «Надежность в технике. Основные понятия. Термины и определения».

3. *Морозов Ю. Д., Ильин И. И.* Методы обеспечения качества и надежности проектов автоматизированных систем. – М.: МЭСИ, 1990.

4. *Майерс Г.* Надежность программного обеспечения. - М.: Мир, 1980.

5. *Леонтьев Е. А.* Надежность экономических информационных систем. – Тамбов:ТГТУ, 2002.

6. *Кульба В.В., Ковалевский С.С., Шелков А.Б.* Достоверность и сохранность информации в АСУ. – М.:СИНТЕГ, 2003.

Михеев В.А.
Научный руководитель: д.т.н., профессор Верба В.С.
г. Москва, ОАО «Концерн «Вега»

Лозунг: ***Контурирование – основа создания единого защищенного интегрированного информационного пространства многофункциональной информационной системы***

РАЗРАБОТКА МОДЕЛИ БЕЗОПАСНОСТИ МНОГОФУНКЦИОНАЛЬНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Рассмотрена модель безопасности многофункциональной информационной системы холдинга. Предложен метод контурирования системы по категориям обрабатываемой информации. Разработан «модуль безопасности» и сформулированы ограничения для задачи синтеза оптимального варианта комплекса СЗИ.

Информационная система, модель безопасности, метод контурирования, контур безопасности.

Многофункциональная территориально распределенная информационная система (МИС) – взаимосвязанная совокупность кампусных локальных вычислительных сетей (ЛВС) и корпоративной сети передачи данных (КСПД), предназначенная для создания единого защищенного интегрированного информационного пространства информационных ресурсов предприятий холдинга. В связи с необходимостью циркуляции в МИС как общедоступной информации, так и информации

ограниченного доступа, возникают требования по разработке модели безопасности системы, а также особых подходов к проектированию и построению МИС как к автоматизированной системе в защищенном исполнении, отвечающей требованиям безопасности информации [1].

Разработка модели безопасности МИС производится на основе подходов и системотехнических принципов, сформулированных в [2,3].

Учитывая территориальную распределенность МИС и существующие требования по защите информации, обоснованным решением при проектировании и построении МИС является применение метода «контурирования» системы по категориям обрабатываемой информации.

Применение в модели безопасности МИС метода «контурирования» позволяет разбить МИС на три защищенных контура обработки информации (контуры безопасности), представленных на рис. 1:

- публичный информационный контур (ПИК) – выделяется для обработки, передачи и представления общедоступной информации в сетях международного информационного обмена (включая сеть Интернет);
- внутренний информационный контур (ВИК) – выделяется для обработки конфиденциальной информации, включающей коммерческую тайну, персональные данные и другие сведения, которые могут быть отнесены к конфиденциальным;
- защищенный информационный контур (ЗИК) – выделяется для обработки информации, содержащей сведения, составляющие государственную тайну.

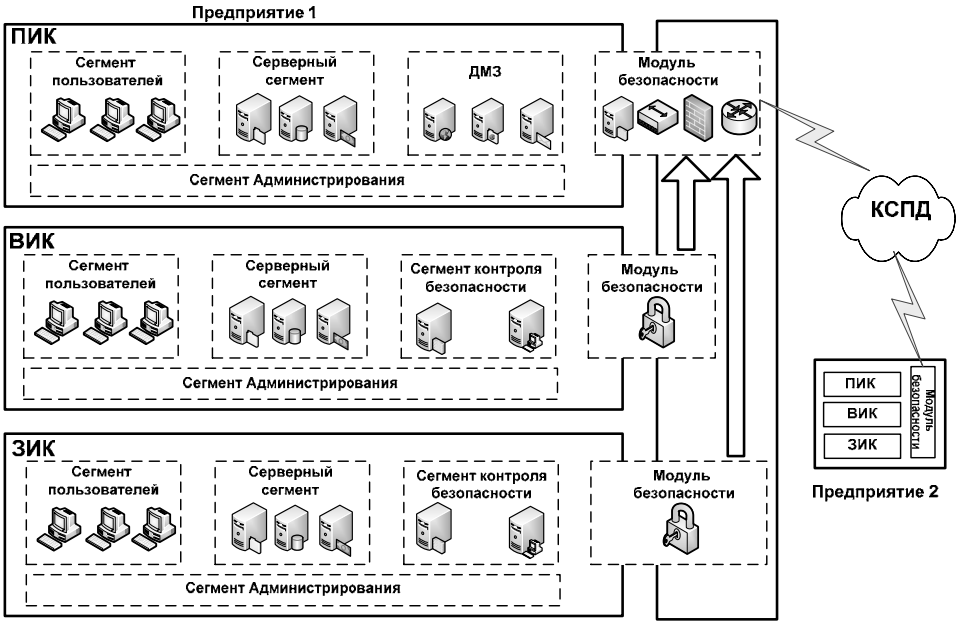


Рис. 1. Контуры обработки информации (контуры безопасности)

Объединение кампусных ЛВС предприятий холдинга в единое информационное пространство посредством КСПД происходит путем объединения контуров безопасности.

Угрозы, возникающие при таком объединении, решаются разработкой и внедрением в каждом контуре «модуля безопасности» – программно-аппаратного комплекса, обеспечивающего выполнение требований по защите информации (разных для каждого контура).

Публичный информационный контур обеспечивает доступ к публичным сервисам сети «Интернет». Требования по защите информации для ПИК регламентируются внутренними документами холдинга и обеспечиваются выполнением комплекса организационно-технических мероприятий. Подключение к сети «Интернет» осуществляется через модуль безопасности ПИК, при этом обеспечивается организация «демилитаризованной зоны» (ДМЗ), в которой размещаются публичные ресурсы, такие как интернет-портал, почтовый сервер, DNS-сервер и др.

Внутренний информационный контур обеспечивает хранение, обработку и обмен конфиденциальной информацией. Требования по защите конфиденциальной информации для ВИК регламентируются руководящими документами ФСТЭК и ФСБ и обеспечиваются комплексом организационно-технических мероприятий и проведением аттестации ВИК по требованиям защиты информации. В ВИК применяется целый ряд средств защиты информации (СЗИ), позволяющих обеспечить требования по защите информации. Конкретный перечень СЗИ определяется в соответствии с классификацией ВИК, проводимой на основании РД ФСТЭК [4].

Защищённый информационный контур обеспечивает циркуляцию информации, содержащей сведения, составляющие государственную тайну. Требования по защите информации, составляющей государственную тайну, регламентируются руководящими документами ФСТЭК и ФСБ, обеспечиваются комплексом организационно-технических мероприятий и проведением аттестации ЗИК по требованиям защиты информации.

В модулях безопасности ВИК и ЗИК применяются сертифицированные по требованиям безопасности информации программно-аппаратные комплексы криптографической защиты информации (шифраторы). Эти устройства выполняют функции шифрования, межсетевого экранирования, скрытия архитектуры защищаемого контура, тем самым обеспечивая требуемый уровень защиты контуров.

Любое взаимодействие между контурами безопасности до криптографического преобразования информации в модулях безопасности ВИК и ЗИК исключается путем физического разделения контуров. Потоки информации от всех трех контуров консолидируются в модуле безопасности ПИК, куда данные из ВИК и ЗИК поступают в зашифрованном виде, тем самым исключая их компрометацию.

Таким образом, предлагаемый в статье метод, основанный на контурах безопасности, позволяет решить задачу построения единого защищенного интегрированного информационного пространства МИС, при необходимости обработки в ней как общедоступной информации, так и информации ограниченного доступа.

Дальнейшую задачу проектирования модулей безопасности в каждом контуре можно сформулировать как задачу синтеза оптимального варианта СЗИ [5] при наличии следующих ограничений: уровеня защищенности, надежности, стоимости. Совокупность ограничений вызывает необходимость использования многокритериального подхода для синтеза СЗИ.

В общем случае математически задача синтеза комплекса СЗИ на основе многокритериального подхода формулируется следующим образом: в множестве возможных вариантов найти такую СЗИ $\vec{X} \in \Pi \subset D$, для которой

$$Q(\vec{X}) = (q_1(\vec{X}), q_2(\vec{X}), \ldots, q_m(\vec{X})) \to \underset{\vec{X} \in \Pi \subset D}{extr},$$

$$D : G(\vec{X}) \le 0,$$

где Π – множество Парето – оптимальных решений; D – множество допустимых решений, в пределах которого выполняются ограничения $G(\vec{X}) \le 0$.

В число частных критериев $q_i(\vec{X}), i = 1,2,\ldots,m$ могут входить как предложенные в статье, так и другие.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Михеев В.А.* Методология разработки и аттестации автоматизированных систем в защищенном исполнении//Материалы IX Международной научно-практической конференции «Информационная безопасность». – Таганрог: Изд-во ТТИ ЮФУ, 2007.

2. *Михеев В.А.* Основы построения подсистемы защиты информации многофункциональной информационной системы. «Известия ЮФУ. Технические науки». – Таганрог: Изд-во ТТИ ЮФУ, 2008.

3. *Николаев А.В., Михеев В.А.* Перспективы многофакторных систем // Защита информации INSIDE, Санкт-Петербург: ИД Афина, 2008. Вып. 2.

4. Руководящий документ «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации». М.: Военное изд-во, 1992.

5. *Кащенко А.Г.* Задача многокритериального синтеза структуры и параметров системы защиты информации // Информация и безопасность: региональный научно–технический журнал. Вып. 2. – Воронеж, 2006.

Мотиенко О.В.
Научный руководитель: д.т.н., профессор Макаревич О.Б.
г. Таганрог, ТТИ ЮФУ

Лозунг: Если поищешь в книгах мудрости внимательно,
то найдешь большую пользу для души своей
(с) Нестор-Летописец

МИНИМИЗАЦИЯ ТРАФИКА ПРИ РЕЗЕРВНОМ КОПИРОВАНИИ ВИРТУАЛЬНЫХ ВЫДЕЛЕННЫХ СЕРВЕРОВ ЧЕРЕЗ ИНТЕРНЕТ

В данной статье рассматривается метод минимизации трафика при резервном копировании данных через Интернет с использованием rsync. Приведено описание метода и экспериментальные данные.

Резервное копирование, Интернет, VDS, виртуальные выделенные сервера, Linux, FreeBSD, ssh, rsync.

В последнее время всё более популярным стало использование виртуальных выделенных серверов (от англ. VDS/VPS – Virtual Private/Dedicated Server) – технологии, построенной на идее разделения ресурсов физического сервера многими пользователями, дающей полный контроль над пользовательскими приложениями и практически полный контроль над операционной системой, по аналогии с выделенным физическим сервером.

Виртуализация – это общий термин, охватывающий абстракцию ресурсов для многих аспектов вычислений. Наиболее распространены в настоящее время следующие типы виртуализации:

- Виртуальная машина – это окружение, которое представляется для «гостевой» операционной системы как аппаратное. Однако на самом деле это программное окружение, которое симулируется программным обеспечением основной операционной системы. Эта симуляция должна быть достаточно надёжной, чтобы драйверы гостевой системы могли стабильно работать. Данный тип виртуализации чаще всего используется для виртуализации рабочих станций и реализуется следующим ПО: Bochs, KVM, Parallels, Qemu,VirtualBox, Xen,Virtual Iron, Microsoft VirtualPC,VMware.
- Паравиртуализация – технология виртуализации, при которой гостевые операционные системы подготавливаются для исполнения в виртуализированной среде, для чего их ядро незначительно модифицируется. Наиболее популярным ПО, реализующим эту технологию, является Xen. При использовании паравиртуализации, виртуальная машина не симулирует аппаратное обеспечение, а вместо этого предлагает использовать специальное API.
- Виртуализация на уровне операционной системы – виртуализирует физический сервер на уровне ОС, позволяя запускать изолированные и безопасные виртуальные серверы на одном физическом сервере. Данный тип виртуализации чаще всего используется для виртуализации серверов и реализуется следующим ПО: Solaris Containers / Zones, FreeBSD Jail, Linux-VServer, FreeVPS, OpenVZ / Virtuozzo .

В России наибольшее распространение получили VDS на базе ОС GNU/Linux с вирутализацией на уровне ОС, например, на основе OpenVZ. Многие компании предлагают коммерческую аренду виртуальных серверов по доступной цене, включающую помимо аренды сервера дополнительные услуги, в том числе периодическое резервное копирование данных (обычно ежемесячное или еженедельное). Однако для повышения надежности проекта иногда требуется проводить резервное копирование с меньшей периодичностью и/или размещать резервные копии на компьютерах пользователя. В последнем случае резервное копирование производится через Интернет, поэтому необходимо обеспечить шифрование передаваемых данных и минимизировать их объем. В связи с этим был разработан и реализован метод резервного копирования на основе открытых технологий, применяемых в UNIX-подобных системах: OpenSSH и Rsync.

OpenSSH (открытый безопасный shell) – набор программ, предоставляющих шифрование сеансов связи по компьютерным сетям с использованием протокола SSH. Он был создан командой OpenBSD под руководством Teo de Raadt (Тэо де Раадт) как открытая альтернатива проприетарного ПО от SSH Communications Security., который все еще является проприетарным ПО. Разработчики OpenSSH утверждают, что он более безопасен чем оригинальный Secure Shell, благодаря их политике чистки и аудита кода. OpenSSH является стандартом de facto для UNIX-подобных систем.

Rsync (англ. Remote Synchronization) – это программа для UNIX-подобных систем, которая выполняет синхронизацию файлов и каталогов в двух местах с минимизированием трафика, используя кодировку данных при необходимости. Важным отличием rsync от многих других программ/протоколов является то, что зеркалирование осуществляется одним потоком в каждом направлении (а не по

одному или несколько потоков на каждый файл). Rsync может копировать или отображать содержимое каталога и копировать файлы, опционально используя сжатие и рекурсию. Выпущен под лицензией GNU GPL, rsync является свободным программным обеспечением. Утилита rsync использует алгоритм, разработанный австралийским программистом Эндрю Триджеллом, для эффективной передачи структур (например, файлов) по коммуникационным соединениям в том случае, когда принимающий компьютер уже имеет отличающуюся версию этой структуры.

Принимающий компьютер разделяет свою копию файла на неперекрывающиеся куски фиксированного размера S и вычисляет контрольную сумму для каждого куска: MD4 хэш и более слабый 'rolling checksum', и отправляет их серверу, с которым синхронизируется.

Сервер, с которым синхронизируются, вычисляет контрольные суммы для каждого кусочка размера S в своей версии файла, в том числе перекрывающиеся куски. Это может быть эффективно подсчитано ввиду особого свойства rolling checksum: если rolling checksum байт от n до n+S-1 равняется R, то rolling checksum байт от n+1 до n+S может быть посчитан исходя из R, байта n и байта n+S без необходимости учитывать байты, лежащие внутри этого интервала. Таким образом, если уже подсчитана rolling checksum байт 1-25, то для подсчета rolling checksum байт 2-26 используется предыдущая контрольная сумма и байты 1 и 26.

В представленном методе Rsync использует OpenSSH (с алгоритмом аутентификации DSA для беспарольной аутентификации) для организации шифрованного туннеля через Интернет.

Основное отличие представленного метода от «классических» методов резервного копирование через сеть – создание и периодическая синхронизация локального «зеркала» файловой системы VDS при помощи rsync с включенной компрессией данных (библиотека zlib). Это позволяет минимизировать трафик, передаваемый через Интернет. После синхронизации «зеркала» можно производить его резервное копирование с помощью любого подходящего ПО, как и в случае локальной файловой системы. Как наиболее популярный был выбран метод инкрементального (добавочного) резервирования (англ. Incremental Backup).

Метод состоит из двух частей: подготовка VDS к резервному копированию и собственно периодическое резервное копирование данных.

Во время подготовки необходимо настроить OpenSSH на беспарольную аутентификацию путем генерации и размещения DSA ключей на передающей и принимающей стороне. Также необходимо установить программу rsync, если она отсутствует в стандартом наборе программ. Также необходимо сделать начальную синхронизацию «зеркала» файловой системы VDS.

Периодическое резервное копирование производится следующим образом:

1. Производится синхронизация локального «зеркала» с помощью rsync.
2. В случае успешного завершения rsync производится вызов программы резервного копирования.
3. Программой резервного копирования производится создание архива файловой системы «зеркала» VDS «классическим» методом, например методом инкрементального резервирования.

В табл. 1 приводятся результаты, полученные при тестировании метода резервного копирования данных из VDS, размещенного в г. Москве, на сервер, размещенный в г. Таганроге. Для сравнения использовался «классический» инкрементальный метод с периодом в 7 дней с использованием сжатия gzip и bzip2. В качестве архиватора использовалась программа GNU tar. Были рассмотрены данные, полученные в первые 10 дней после начала резервного копирования, для

удобства дни пронумерованы от 0 до 9. Во время тестирования учитывались следующие параметры: общий объем данных, объем файлов, выбранных для резервного копирования (измененные и новые файлы), объем данных, переданных по сети, и время их передачи, а также средняя нагрузка на передающий сервер (Load average).

Таблица 1

Экспериментальные данные

День	0	1	2	3	4	5	6	7	8	9
Общий объем данных, Мб	1347	1351	1352	1354	1421	1422	1428	1431	1433	1435
Объем файлов, выбранных для резервного копирования, Мб	1347	114	124	95	171	118	102	1431	112	121
Резервное копирование «классическим методом», со сжатием gzip										
Объем данных, переданных по сети, Мб	916	102	106	84	148	101	94	927	103	108
Время резервного копирования по сети, с	3718	458	514	382	682	395	436	4214	464	489
Средняя нагрузка	0.36	0.34	0.35	0.36	0.34	0.35	0.35	0.35	0.34	0.35
Резервное копирование «классическим методом», со сжатием bzip2										
Объем данных, переданных по сети, Мб	871	96	101	78	136	94	89	881	96	99
Время резервного копирования по сети, с	3553	439	446	371	631	433	414	4004	438	448
Средняя нагрузка	0.74	0.73	0.74	0.75	0.74	0.74	0.73	0.74	0.75	0.74
Резервное копирование разработанным методом, со сжатием gzip										
Объем данных, переданных по сети, Мб	927	14	12	7	82	16	14	16	12	15
Время резервного копирования по сети, с	3820	61	57	32	324	64	59	63	58	61
Средняя нагрузка	0.73	0.78	0.76	0.77	0.75	0.77	0.78	0.77	0.77	0.77

На основе полученных данных можно сделать следующие выводы:

1. При использовании «классических» методов производится периодическая передача всех файлов по сети, что даже при использовании сжатия создаёт большой трафик и увеличивает время резервного копирования. При использовании разработанного метода полная передача всех файлов производится 1 раз (см. день 0).
2. При использовании «классических» методов производится передача всех измененных файлов целиком. При использовании разработанного метода производится передача только самих изменений в файлах.
3. Разработанный метод более эффективен в случае, если производится резервное копирование файлов, в которых происходят изменения, и менее эффективен, если производится копирование вновь созданных файлов (см. день 4).
4. Использование rsync вызывает повышение средней нагрузки, однако нагрузка сравнима с использованием «классического» метода с со сжатием bzip2, при большей эффективности разработанного метода.
5. К недостаткам разработанного метода можно отнести требование

дополнительного объема дискового пространства на принимающем сервере, равного общему объему резервируемых данных, и дополнительного времени, необходимого для архивации полученного «зеркала». Однако уменьшение издержек на передачу данных по сети компенсирует эти недостатки.

Аналогичные данные были получены и при тестировании метода на других серверах. Данный метод успешно применяется компанией «Мегалинк» (г. Таганрог) для резервирования сайтов и баз данных, размещенных на VDS в г. Москве и г. Санкт-Петербурге. Кроме того, возможно применение этого метода не только для резервного копирования VDS, но и также и других ресурсов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Daniel J. Barrett, Richard E. Silverman, and Robert G. Byrnes – SSH: The Secure Shell (The Definitive Guide), O'Reilly 2005 (2nd edition). ISBN 0-596-00895-3
2. Michael Stahnke – Pro OpenSSH, Apress 2005. ISBN 1-59059-476-2
3. Шредер К., Linux. Сборник рецептов. – Питер, Издательский дом, 2006 г. ISBN: 0-596-00640-3, 5-469-01188-7.
4. W.Curtis Preston "UNIX Backup and Recovery" O`Reilly, ISBN: 1-56592-642-0

Нархов К.Г.
Научный руководитель: к.ф.-м. н., Грюнталь А.И.
г. Москва, НИИСИ РАН

Лозунг: Модели должны работать!

ПРОГРАММИРОВАНИЕ В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ: АВТОМАТИЗАЦИЯ И БЕЗОПАСНОСТЬ

Настоящая статья посвящена практическим вопросам технологии разработки прикладных программ реального времени с использованием средств моделирования и генерации исходного кода. В статье рассмотрены принципы реализации генератора, основные идеи формализации алгоритмических языков программирования, вопросы безопасности и надежности, а также перспективы развития.

Программирование, системы реального времени, генерация исходных текстов, XML, UML, моделирование, модель, надежность, многопоточность, исходный код.

Безопасность разработки программ для систем реального времени

Увеличение сложности вычислительных систем реального времени подняло на новый уровень проблему обеспечения их информационной безопасности. В настоящий момент требования к механизмам информационной безопасности стали обязательными функциональными требованиями, предъявляемыми к приложениям реального времени еще на этапе постановки задачи по их проектированию. При этом для систем жесткого реального времени всегда были и остаются особо актуальными вопросы функциональной безопасности – способности вычислительной системы не создавать угрозы жизни и здоровью людей или сохранности материальных активов. Таким образом, надежность и отказоустойчивость являются важными аспектами информационной безопасности программного обеспечения реального времени [2].

Одна из основных задач разработки программ реального времени – это обеспечение эффективности и надежности приложений уже на этапе проектирования. Безусловно, это требует высокого профессионального уровня разработчиков и наличия соответствующих инструментов, как то среды разработки, отладки и моделирования приложений. Эти инструменты должны учитывать требования стандартов (ГОСТ Р ИСО/МЭК 1540 [12], ГОСТ Р 51904-2002 [11] и т. д.), в частности, классификации категорий отказов, а также вытекающих из нее классификации уровней программного обеспечения и требований к тестированию этого программного обеспечения.

Следует отметить, что одним из направлений по повышению отказоустойчивости приложений является применение инструментов проектирования на основе современных инноваций – унифицированного языка моделирования (UML) [7] и расширяемого языка разметки (XML) [3]. Использование перечисленных технологий позволяет повысить надежность программ за счет применения типовых программных компонентов, а также уменьшить трудоемкость разработки.

Средства автоматизации разработки приложений реального времени

Как было сказано ранее, повышение надежности может быть достигнуто применением в процессе создания приложений реального времени автоматизированных средств построения и анализа моделей вычислительных систем реального времени (ВС РВ), автоматизированной генерации прототипов типовых программных модулей на языке программирования (например, Си) для компонентов ВС РВ, имитационного моделирования работы ВС РВ.

Автоматизированные средства построения и анализа моделей представляют собой гибридный XML/UML-редактор, с помощью которого возможно как создание UML-диаграмм, так и специфицирование узлов и элементов этих диаграмм с помощью XML-документов. Документы, которые, по сути, описывают диаграмму на новом уровне абстракции, должны удовлетворять некоторой XML-схеме. По этой схеме можно провести проверку (валидацию) узлов диаграммы, и уже на этапе проектирования иметь информацию о работоспособности будущей программы.

Особое место занимает переход от визуального проектирования к конкретному исходному коду на алгоритмическом языке. Для решения данной задачи используется опять же технология XML. С ее помощью формализуется синтаксис некоторого алгоритмического языка и составляется соответствующая XML-схема [4]. В дальнейшем на основании этой схемы создаются базовые конструкции, которые впоследствии используются в качестве представлений UML-диаграмм.

Проектировщику предоставляется возможность «программировать» визуальными средствами (диаграммы), при этом исходный код, стоящий за тем или иным узлом диаграммы, является выверенным и предоставляется в качестве библиотеки конструкций (возможно даже типовых скелетов программ) к инструментарию проектирования.

Принципиальная схема средств разработки приложений реального времени представлена на рис. 1.

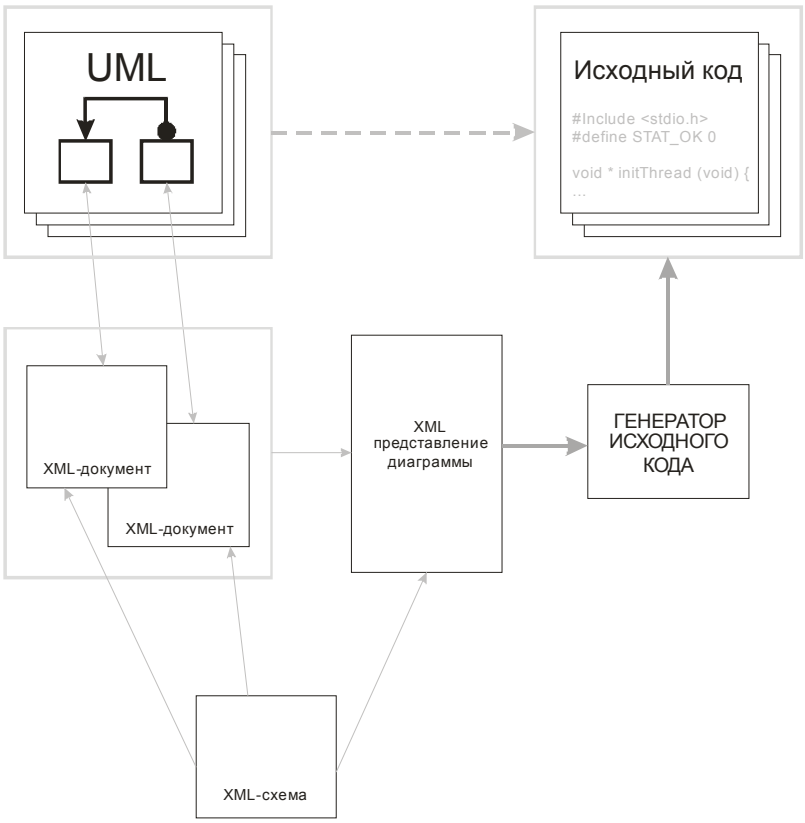


Рис. 1. Принципиальная схема средств разработки

Формализация языка программирования посредством XML

Существует несколько подходов к формализации языка программирования посредством XML. Среди них: составление формального представления произвольного множества конструкций языка и составление формального представления на основании стандарта на язык программирования.

Формальное представление произвольного множества конструкций алгоритмического языка основано на выделении из множества его конструкций некоторого подмножества, составлении иерархии элементов этого подмножества, а также спецификации отношений (связей) между этими элементами. Например, для любого языка программирования можно выделить элементы «программа», «операция», «условный переход», «цикл», «вызов подпрограммы». Такое формальное представление имеет ряд преимуществ. Во-первых, оно является минимально достаточным с точки зрения результатов моделирования, так как в блоки выделяются только требуемые элементы программы. Во-вторых, программы, используемые для обработки формального представления произвольного множества конструкций языка, являются простыми в силу того, что подмножество формализованных элементов обычно намного меньше множества конструкций алгоритмического языка. Тем не менее формальное представление произвольного множества конструкций языка не предоставляет в полном объеме возможностей для контроля синтаксических ошибок внешними средствами. Более того, модели программ, основанные на

таком формальном представлении, являются ограниченными вследствие ограниченности самого формального представления.

Представление синтаксиса Си в терминах XML на основании стандарта выполняется путем полного описания синтаксиса алгоритмического языка Си, изложенного в стандарте [6]. В приложении А («Annex A») стандарта приведена иерархия элементов синтаксиса, а также специфицированы отношения (связи) между этим элементами.

Представление на основе стандарта является полным, следовательно, может быть использовано для представления любой Си-программы, кроме того, такое представление языка используется в средствах контроля синтаксических ошибок. С другой стороны, эта формализация избыточна: при моделировании программного обеспечения вряд ли потребуется оперировать такими элементами, как, например, «символ» или «escape-последовательность».

Выбор подхода к формализации не является однозначным и напрямую зависит от требований, предъявленных к формальному представлению. Отметим, что формальное представление произвольного множества конструкций алгоритмического языка удобно использовать, например, в системах имитационного моделирования. В этих системах с каждым элементом ассоциированы те или иные расчетные характеристики: быстродействие, объем требуемой оперативной памяти, интервал процессорного времени и т. д.

Формальное представление на основе стандарта языка программирования Си рекомендуется к использованию в системах, где одними из важных требований являются проектирование произвольных отдельных участков исходного кода, контроль синтаксических ошибок и реинженеринг готовых программ.

Генератор исходного кода

Генератор исходного кода (анализатор XML) работает по событийно-поточному принципу разбора документов [9]. Использование стратегии, основанной на потоках, подразумевает непрерывную передачу XML-данных генератору. Генератор функционирует подобно каналу, в качестве исходных данных для которого выступает XML-разметка. Этот канал называется потоком событий. Происхождение подобного названия связано с тем, что каждый фрагмент анализируемых данных сигнализирует о некотором событии, происшедшем в потоке входных данных. Например, к важным событиям можно отнести начало нового элемента. В этом случае выполняется поиск инструкций по обработке, которые могут содержаться либо в самой разметке, либо во внешнем хранилище данных (например, в базе данных). В результате выполнения каждого нового обновления потока событий (чтение XML-документа, прием информации из сокета и т. д.), генератор выполняет новую трансляцию данных (смещение блоков данных в канале). Затем выполняется проверка текущего блока данных на предмет наличия специфического содержимого, которое передается соответствующему обработчику и в последующем помещается во внутренние информационные структуры (например, дерево представления входного XML-документа или стек событий).

Принципиальная схема генератора исходных текстов представлена на рис. 2.

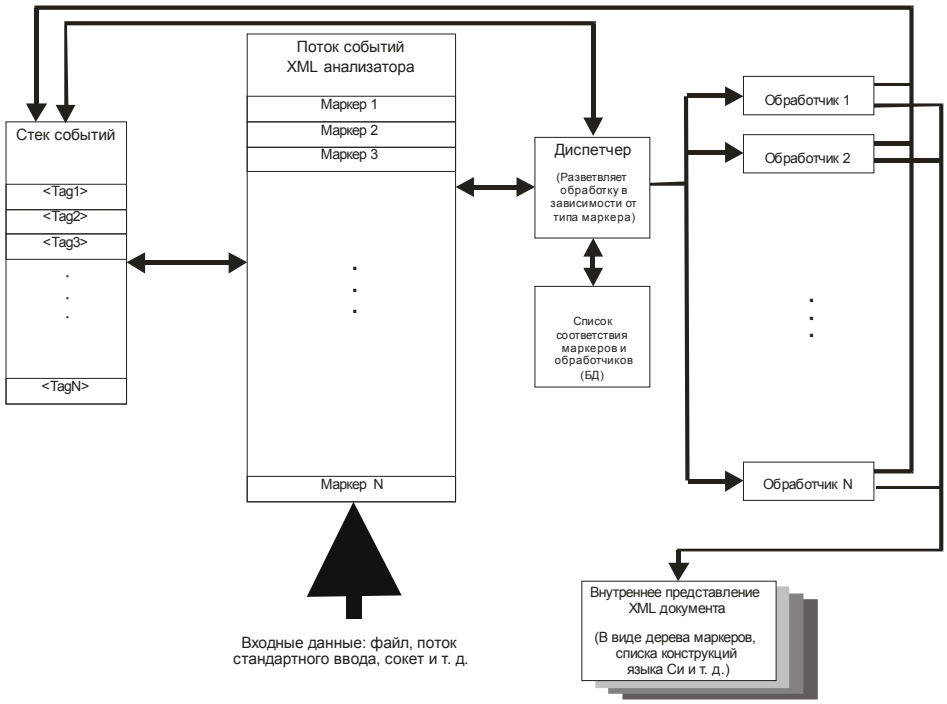


Рис. 2. Принципиальная схема генератора исходного кода (XML анализатора)

Потоки событий генератора являются группами данных. Каждая группа данных, которая именуется маркером, представляет собой набор из одного или большего количества символов. Каждый маркер соответствует типу разметки, например, начальному либо конечному тегу элемента. Благодаря этому упрощается разбор XML-кода с применением анализаторов, а также минимизируются требуемые ресурсы и время.

Особенность XML-потоков заключается в содержимом, присущем каждому маркеру; маркер должен содержать XML-код, который отвечал бы требованиям формальной корректности. Данные, которые не подчиняются требованиям формальной корректности, будут исключены из обработки генератором.

Упомянутые контекстные правила применимы в отношении к анализаторам, равно как и к процессорам, выполняющим обработку данных (обработчикам).

Язык XML иерархичен по своей структуре (элементы содержат другие элементы), в связи с этим невозможно выделять отдельные элементы и использовать их в качестве маркеров потока. В документе, отвечающем правилам формальной корректности, все элементы содержатся в одном корневом элементе. Корневой элемент, содержащий весь документ, сам по себе не является потоком. Поэтому трудно представить поток, формирующий завершенный элемент в виде маркера (если речь не идет о пустом элементе).

В силу упомянутых причин XML-потоки составляются на основе событий. Любое событие является сигналом, свидетельствующим об изменении состояния документа. Например, если генератор обнаруживает начальный тег элемента, сообщается, что был открыт другой элемент, и изменилось состояние синтаксического разбора. Завершающий тег влияет на состояние, закрывая последний открытый элемент. Обработчики событий XML могут отслеживать открытые элементы в структуре стека данных. При этом в стек помещаются сведения о только что от-

крытых элементах и исключаются данные о закрытых элементах. В любой момент синтаксического разбора генератор и соответствующие обработчики могут оценить глубину своего нахождения в документе путем анализа стека.

После обработки данные помещаются в информационные структуры, которые в совокупности являются внутренним представлением XML-документа. Эти структуры могут быть как деревьями, так и прочими представлениями данных (массивы, списки, очереди). Основные критерии выбора структуры для внутреннего представления – сохранение логических сущностей и связей между ними, а также удобство анализа и обработки. На основании внутреннего представления выполняется генерация исходных текстов программ на алгоритмическом языке.

Требования к генератору исходных текстов:

- генератор исходных текстов программ должен генерировать программные модули, содержащие прототипы процессов и потоков управления, пригодные для последующей компиляции, сборки и выполнения на аппаратных средствах проектируемой системы. Генерация исходных текстов программ должна выполняться на основе типовых заготовок таких программ с возможным интерактивным вмешательством программиста;
- генератор исходных текстов программ должен обеспечивать встраивание в генерируемые программы дополнительных операторов (на основе типовых заготовок и/или встраиваемых в интерактивном режиме), обеспечивающих протоколирование работы макета системы и имитацию поведения реальных прикладных программ при прогоне макета на имитационной модели.

Перспективы развития

Генератор исходных текстов может применяться как в виде отельного полнофункционального приложения, так и в составе системы проектирования. В настоящее время активно развиваются так называемые IDE-среды (Integrated development environment), предоставляющие интерфейсы для разработки собственных расширений (add-on) и плагинов (plug-in). Таким образом, генератор исходных текстов может быть реализован для систем типа Eclipse, NetBeans и т. д.

Не менее важна перспектива создания XML-библиотек конструкций для популярных языков типа Java, C#, VisualBasic, а также реализация средств автоматизированного документирования сгенерированного исходного кода.

Особое место занимает подготовка скелетов программ, которые могут быть включены в общую XML-библиотеку. Это могут быть скелеты потоков управления, процессов, критические участки кода, обработчики исключений, прерываний и т. д.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Отчет о научно-исследовательской работе «Анализ выполнимости приложений реального времени с заданным структурированием» по программе фундаментальных научных исследований ОИТВС РАН «Оптимизация вычислительных архитектур под конкретные классы задач, информационная безопасность сетевых технологий», направление «Разработка информационно-безопасных распределенных вычислительных систем». – Москва, НИИСИ РАН, 2006.

2. *В.Б. Бетелин, В.А. Галатенко, А.Н. Годунов, А.И. Грюнталь.* Анализ информационной безопасности систем на платформе ОС РВ БАГЕТ / Безопасность информационных технологий. – 2002. – № 4. – С. 65–72.

3. Extensible Markup Language (XML) 1.1. W3C Recommendation 04 February 2004, edited in place 15 April 2004.

4. XML Schema Part 1: Structures Second Edition (W3C Recommendation 2004-10-28).

5. XML Schema Part 2: Datatypes Second Edition (W3C Recommendation 2004-10-28).
6. International Standard ISO/IEC 9899:1999. Programming languages – C.
7. Unified Modeling Language: Infrastructure, version 2.0, formal/05-07-05, Object Management Group specification.
8. Unified Modeling Language: Superstructure, version 2.0, formal/05-07-04, Object Management Group specification.
9. Perl & XML. Библиотека программиста / Э. Рей, Дж. Макинтош. – СПб.: Питер, 2003. – 208 с.: ил.
10. ГОСТ Р ИСО/МЭК 8825-4 2006 Правила кодирования АСН.1, часть 4. Правила XML-кодирования (XER).
11. ГОСТ 51904-2002 Программное обеспечение встроенных систем. Общие требования к разработке и документированию.
12. ГОСТ Р ИСО/МЭК 15408-1-2002 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель.

Абрамов Г.Э.
Научный руководитель: д.т.н., профессор Макаревич О.Б.
г. Таганрог, ТТИ ЮФУ

Лозунг: Модель аномального поведения

МОДЕЛЬ АНОМАЛЬНОГО ПОВЕДЕНИЯ СИСТЕМЫ НА ОСНОВЕ ВЕРОЯТНОСТНЫХ СУФФИКСНЫХ ДЕРЕВЬЕВ

Описывается метод применения вероятностных суффиксных деревьев для обнаружения аномального поведения программ. Используется «отпечаток» нормального поведения приложений с целью в дальнейшем обнаружить аномальное поведение как нечто, отклоняющееся от модели. В качестве основной модели используется вероятностные суффиксные деревья.

Вероятностное суффиксное дерево, probabilistic suffix tree, PST, обнаружение аномального поведения.

Классификация активности приложений на основе вероятностных суффиксных деревьев

Вероятностное суффиксное дерево (probabilistic suffix tree, PST), описанное в [1], это n-арное дерево, в котором узлы организованы так, что корневой узел представляет безусловную вероятность каждого символа в алфавите, в то время как узлы на следующих уровнях представляют вероятности возникновения следующего символа при условии, что уже наблюдалась комбинация одного или большего числа символов (т.е. при наличии истории). Вероятности – это относительные оценки, вычисляющие частоту возникновения символа в обучающем примере или примерах. PST также имеют свойство порядка, соответствующее глубине дерева, так что PST порядка k содержит k+1 уровень. Для иллюстрации, рис. 1 показывает PST третьего порядка, порожденное из образца «2 1 1 3 1 3 1 3 3 4», где символы «2» и «4» однозначно используются как начало и конец строки.

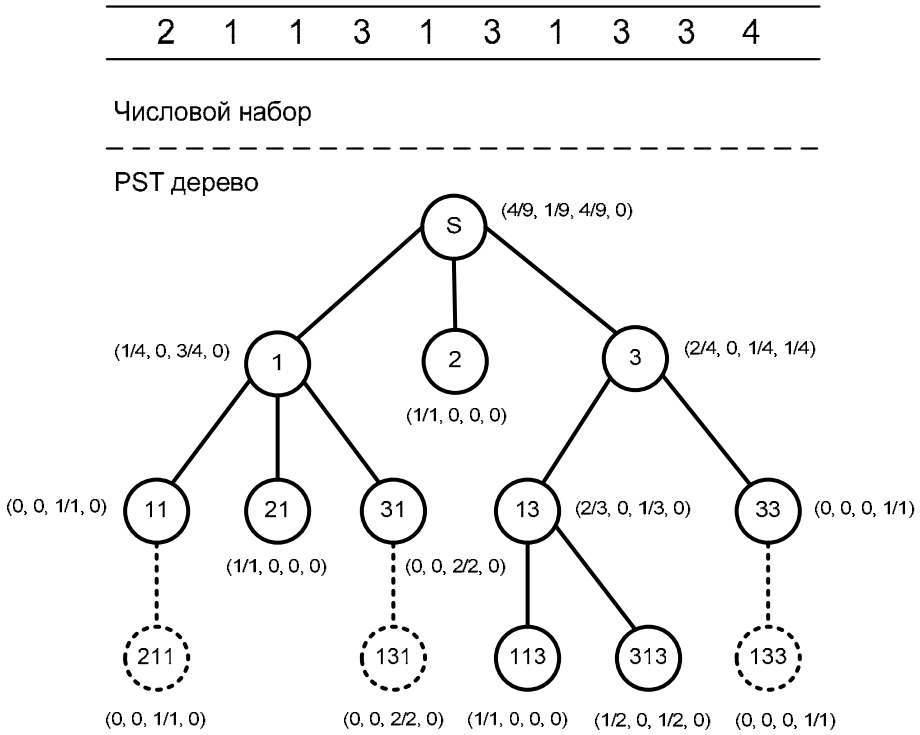


Рис. 1. Пример построения PST

Начиная с корневого узла, который представляет пустую строку, каждый узел является суффиксом всех своих потомков, поэтому модель получила название суффиксного дерева. Корень, представляющий пустую строку, является непосредственным суффиксом всех единичных символов. Числа в круглых скобках, расположенные рядом с узлами, являются условными вероятностями следующих символов, за исключением того, что распределение вероятностей следующих символов для корневого узла имеет вероятность, равную нулю для конечного символа «4», поскольку этот символ сам не имеет следующих символов в обучающем примере. Необходимо также отметить, что переходы к следующему символу осуществляются от одной ветки к другой, а не от родительского узла к потомку, вследствие формата суффикса узла.

Ниже представлен рекурсивный алгоритм для построения дерева. Пример для построения сканируется для определения вероятности каждого символа. Каждый символ с ненулевой вероятностью становится потомком корневого символа, и для каждого такого узла пример пересканируется для определения распределения вероятностей следующих символов. Это может создать новые листья и добавить новый уровень к PST. Построение рекурсивно добавляет дополнительные уровни путем проверки каждого текущего листа для определения, могут ли быть созданы новые листья среди потомков данного узла. Информация о символах, предшествующих данному символу, сохраняется, поскольку эти символы могут встречаться в суффиксной информации потомков данного узла. Узел добавляется только если подстрока символов, которая соответствует метке узла есть в обучающем примере и для него существует ненулевое распределение вероятности следующего символа. Многие ветви отмирают прежде, чем распространяются до максимума глуби-

ны. Глубина (т.е. порядок) дерева может быть установлена в максимальное допустимое значение, или по умолчанию – это длина входной последовательности. Для каждого узла дерева память должна выделяться динамически.

Далее рекурсивный восходящий процесс усечения должен удалить ветви, которые предоставляют ту же информацию, что и родительские узлы. Удаление этих узлов, «аналогичных родительским», снижает избыточность в структуре дерева и таким образом, создает экономичную модель. Пунктирные линии и фигуры на рис. 1 показывают отсеченные узлы и соответствующие ветви. Отсечение базируется на вероятностной информации.

Главная роль PST в нашем случае – это стохастическое представление обучающих данных в древовидной структуре. Несколько PST, потенциально представляющих разные или подобные обучающие примеры, могут быть слиты в одно дерево, которое может быть получено другим путем при использовании всех индивидуальных обучающих последовательностей. Эта операция позволяет проводить прямые манипуляции с самими PST без потери дополнительных вычислительных мощностей для разборки большого дерева PST. В отношении реконструкции, как упомянуто выше, определенные узлы убираются из PST, если они предоставляют ту же стохастическую информацию, как и их предки. Существует алгоритм для реконструкции вероятностной информации следующих символов для удаленных узлов и, таким образом, реконструкции или заполнения узлов в PST. Эта техника может использоваться для выполнения аналитических операций с PST, таких как сравнение двух деревьев. Есть несколько вариантов алгоритмов нахождения расстояния между двумя PST: для нахождения степени похожести или непохожести между PST.

Основываясь на изложенной выше информации, приведём далее алгоритм построения PST.

1. Исходный образец сканируется для определения вероятности каждого символа.
2. Каждый символ с ненулевой вероятностью становится дочерним узлом корневого узла, и для каждого такого узла исходный образец сканируется заново для определения вероятности следующего символа.
3. Рекурсивно добавляются новые уровни путём анализа каждого конкретного узла. Определяется, необходимо ли создавать новые узлы-потомки.
4. Информация о предшествующих символах запоминается, так как они могут появиться как дети потомков самих себя.
5. Узел добавляется только тогда, если подстрока символов, относящихся к узлу, находится в исследуемом исходном образце и если для него существует ненулевая вероятность.
6. Глубина (порядок) может быть изменена на максимально возможную, или, по умолчанию, равна длине входного исходного образца.
7. После этого проходом снизу вверх удаляются узлы, содержащие такую же статистическую информацию (вероятности), как и их родители. Этим устраняется избыточность структуры дерева.

Для использования суффиксных деревьев в задачах обнаружения атак необходимо иметь механизм сравнения различных PST. В этом случае процедура сравнения просматривает образец с использованием PST, для того чтобы вычислить кумулятивную вероятность возникновения символов в образце.

Для сравнения образцов с использованием PST начинаем с первого символа и смотрим на корневой узел, чтобы определить вероятность возникновения данного символа. Затем продолжаем просматривать последовательность, добавляя следующий символ к предшествующему, получая таким образом новую метку. Далее

необходимо найти узел, основанный на вновь созданной метке, исключая последний символ. Например, если мы пытаемся найти "1 3 2", мы пытаемся найти узел "1 3", поскольку он содержит вероятность "2", возникающей после этого. Если узел не существует, мы удаляем один символ с внешней стороны узловой метки до тех пор, пока мы не столкнемся с узлом, который существует в дереве. Этот процесс назван "backtracking", поскольку происходит перемещение по дереву к корню, пока не будет найден правильный узел для того, чтобы продолжать процесс сравнения. Отметим, что в наихудшем случае необходимо возвращаться в корень, поскольку он должен содержать вероятность следующего символа для того, чтобы приступать к алгоритму сравнения.

Суммарная вероятность сравнения получается путем умножения вероятности в каждой точке сравнения на текущую суммарную вероятность. Например, если мы хотим сравнить пример "1 3 2" с PST для последовательности «1 2 3 1 2 3 2 1 3», показанным на рис. 2 должны быть выполнены следующие шаги.

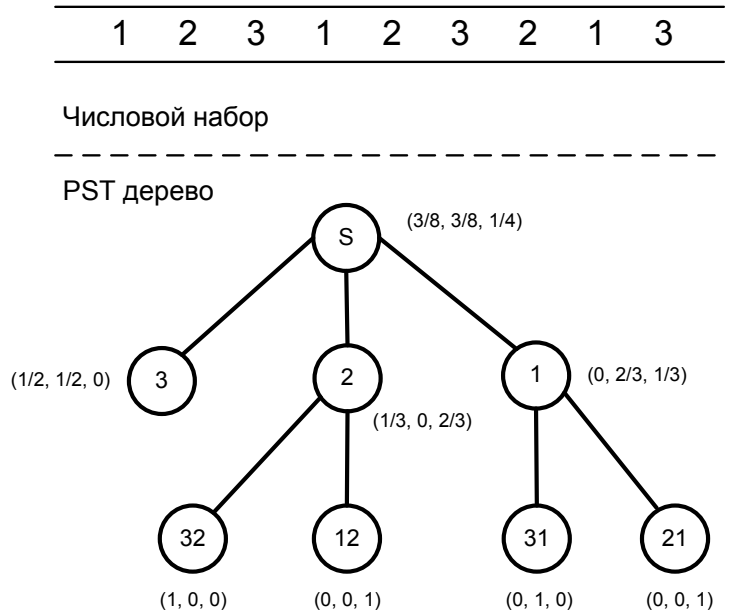


Рис. 2. Пример для иллюстрации сравнения PST

Шаг 1: корневой узел дает вероятность 3/8 при обнаружении "1" .

Шаг 2: узел "1" дает условную вероятность 1/3 при обнаружении "3" , так что наша совокупная вероятность теперь 3/24.

Шаг 3: узел "1 3" не существует, поэтому мы удаляем символ с внешней стороны, которая дает "3". Узел "3" дает вероятность 1/2 при обнаружении "2". Совокупная вероятность сопоставления для образца – 3/48.

Чем выше вероятность сравнения, тем более вероятно, что данный образец был использован при построении суффиксного дерева.

Например, при сравнении "1 2" с PST на рис. 2, вероятность сравнения – 2/8. Это может быть проверено, поскольку из возможных восьми случаев "1 2" возникает в обучающих данных дважды.

При сравнении "1 2" – более вероятно использовано для того, чтобы построить суффиксное дерево, чем "1 3 2" с совокупной вероятностью 3/48.

Сравнение суффиксных деревьев может быть основано на расстоянии (т.е. значении отличия) между ними. Расстояние между деревьями определено как евклидово расстояние между узлами каждого дерева.

Расстояние между двумя узлами определяется следующим образом:

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 + ... + (k_1 - k_2)^2 . \quad (1)$$

Расстояние равно 0 тогда и только тогда, когда два узла идентичны.

Есть два типа сравнения: семантическое и потоковое. Семантическое сравнение включает только те узлы, которые существуют в обоих деревьях. В зависимости от обучающих данных, одно дерево может содержать больше узлов, чем второе, что делает процесс сравнения более сложным. Для того, чтобы решить эту проблему, нужно учесть, что если данный узел не существует в дереве, то оно может иметь ту же вероятность, что и родитель. Можно затопить дерево ненужными узлами для того, что бы выполнить сравнение, путем создания такого количества дочерних узлов, которое необходимо. При этом копируются распределения вероятностей принадлежащие родительскому узлу, существующему в обоих деревьях. Например, если «1 3» существует в первом дереве, но не существует во втором, мы можем «залить» общий суффиксный узел, «3», во втором дереве путем копирования распределения вероятностей в новый узел, маркированный «1 3». Теперь, когда оба дерева содержат «1 3», операция сравнения может быть успешно применена.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *D. Ron, Y. Singer, and N. Tishby.* The power of amnesia: Learning probabilisitc automata with variable memory length. Machine Learning, 25:117–142, 1996.

2. *M. Schonlau, W. DuMouchel, W. Ju, A. Karr, M. Theus, and Y. Vardi.* Computer intrusion: Detecting masquerades. Statistical Science, 16:1–17, 2001.

Емельянов К.И.
Научный руководитель: д.т.н., профессор Макаревич О.Б.
г. Таганрог, ТТИ ЮФУ

Лозунг: Безопасность беспроводных сетей

РЕАЛИЗАЦИЯ АТАКИ НА ПРОТОКОЛ WPA2

В статье описывается практическая реализация атаки на подсистему обеспечения безопасности беспроводных протоколов TKIP (Temporal Key Integrity Protocol), использующуюся в WPA/WPA2.

Безопасность беспроводных протоколов, WPA, WPA2, TKIP, атака, точка доступа.

Анализ безопасности протокола WPA

Протокол WPA хоть и является более защищённым, чем WEP, но имеет уязвимости, обусловленные используемым алгоритмом шифрования RC4 и использованием алгоритма CRC32, которые не дотягивают по требованиям до криптографической хеш-функции.

На рис. 1 блок Network Data, идет после MAC Header (чек-сумма FCS здесь не показана, так как ее обработка ведется на нижележащем уровне модели OSI). IV –

это тот же 24-битовый WEP Initialization Vector, только смысл его несколько иной, структура данных же пакета значительно усложнилась.

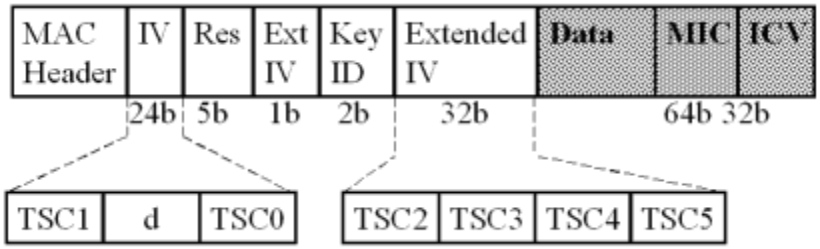


Рис. 1. Формат TKI- пакета

Чтобы понять, что означают представленные поля, рассмотрим протокол TKIP (рис. 2).

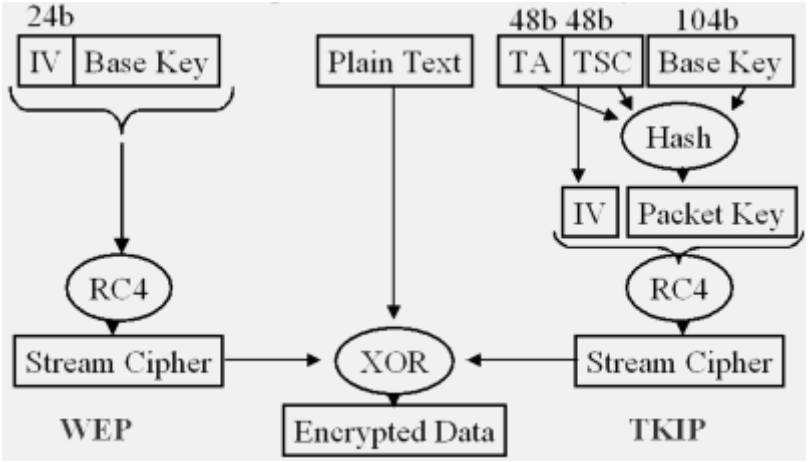


Рис. 2. Алгоритм TKIP

TKIP (Temporal Key Integrity Protocol) реализует три подхода к улучшению безопасности радиопротоколов семейства IEEE 802.11. Во-первых, это функция смешения ключей, которая комбинирует секретный основной ключ PTK (см. ниже) с вектором инициализации перед тем, как передать его в качестве ключа алгоритму RC4. Во-вторых, это последовательный счетчик (TSC – TKIP Sequence Counter) 48-битной длины, значение которого растет с каждым переданным пакетом. Пакеты, полученные в неверном порядке, будут отвергнуты (а именно, будут отвергнуты пакеты с более ранним TSC), что позволяет защищаться от так называемых replay-атак. Наконец, это 64-битный код защиты сообщения MIC (Message Integrity Code).

TKIP реализует также механизм rekeying'а – смены сессионных ключей и гарантирует, что каждый пакет будет передан с уникальным RC4-ключом.

Основной сессионный ключ PTK (Pairwise Transient Key) – это набор из 4-х 128-битных ключей, используемый для TKIP для шифрования отдельных пакетов в периоды между сменами ключей. Заметим, что PTK – это не пароль WEP и ни он, ни механизм его генерации частью TKIP не являются.

В PTK входят 4 ключа: один для шифрования данных в TKIP (назовем его TK), один для вычисления MIC (а его MTK) и еще два так называемых EAPOL ключа, которых я касаться не буду. Ключи генерируются с использованием пароля WPA, известного обеим сторонам в процессе четырехэтапного «рукопожатия».

На рис. 2 видим, что все, что между MAC Header и Data, по сути, – счетчик TSC плюс некоторые служебные поля, служащие для защиты от слабых ключей и говорящие, о том, что используется Extended IV. TSC устанавливается в нуль при начале сессии и монотонно увеличивается с каждым переданным данным устройством пакетом, емкости в 48 бит хватит на ~250 триллионов пакетов, что можно считать достаточным с учетом периодической реинициализации сессионных ключей (в отличие от 24-битного IV в WEP). Уникальный ключ для шифрования пакета вычисляется из TK, TA (Transmittor Address – MAC адрес передатчика) и TSC путем определенного двухфазного механизма хеширования, который дает на выходе 104-битовую строку, к которой добавляется WEP IV для получения 128-битового ключа.

MIC создан в основном для борьбы с поддлекой (forgery) пакетов. Код вычисляется от данных всего сообщения (плюс адреса передатчика и приемника) еще до фрагментации и возможной смены порядка пакетов с помощью алгоритма под названием MICHAEL, генерирующего подпись длины 64 бит. Алгоритм, кроме данных, использует ключ, а именно вышеупомянутый MTK. Что немаловажно, алгоритм в некотором смысле обратим, т.е. зная данные и MIC подпись, можно вычислить ключ MTK.

Борьба с подделкой со стороны Access Point происходит так: если пришел пакет с неверным MIC (при этом он валиден по остальным признакам, т.е. имеет допустимое значение TSC и ICV – сумма прошла проверку), то отправителю посылается уведомление и, если есть возможность, данное событие фиксируется в журнале как попытка взлома. Если в течение 60 секунд приходит еще один пакет с неверным MIC, точка доступа инициализирует rekeying с данным отправителем. Таким образом поддельные пакеты относительно безнаказанно можно слать не чаще, чем раз в минуту.

Практическая атака на WPA

Из внушительного списка атак, к которым уязвим WEP, рассмотрим так называемую Chop-Chop атаку (от англ. chop, вольный перевод – «отрезать ломтик»). Эта атака позволяет узнать plaintext сообщения (т.е. данные сообщения до шифрования), а значит и RC4 keystream пакета (plaintext + keystream = ciphertext => keystream = ciphertext + plaintext). Атака использует тот факт, что CRC32 чек-сумма отнюдь не проходит по требованиям в криптографическую хеш-функцию [3].

CRC есть остаток от деления исходной строки S, представленной в виде многочлена (от X) с коэффициентами, равными соответствующим разрядам в ее двоичной записи, на предопределенный многочлен PCRC(X), причем арифметические операции выполняются в поле GF(2) (подробнее, например, в википедии). Однако в такой форме CRC нечувствителен к нулям в начале и в конце исходной строки, поэтому на практике в начало и конец дописываются специальные строки (длины, равной количеству разрядов CRC), которые обозначим как Li (начало) и Lf (конец). Обычно обе они состоят из 32-х двоичных единиц. Таким образом, для CRC32:

$$CRC = (X^{32} * S + L_i * X^{n+32} + L_f) \bmod P_{CRC}, \quad (1)$$

где n – длина S в битах.

Примечательно, что если к исходной строке S приписать справа ее CRC, то CRC от полученной строки будет постоянна и равна CRC пустой строки, которую обозначим за P_{zero} (доказывается очень просто, достаточно помнить, что в GF(2) сложение и вычитание – одна и та же операция – XOR), или в виде формулы

$$(X^{32} * (X^{32} * S + CRC) + L_i * X^{n+64} + L_f) \bmod P_{CRC} = P_{zero}. \quad (2)$$

В WEP-пакете последние байты данных (Network Data) – это зашифрованный ICV, т.е. CRC от сообщения. Забудем на время о шифровании и рассмотрим, что будет, если убрать последний байт, который обозначим как R, из пакета. Пусть Q – пакет без последнего байта, тогда Q уже навряд ли будет иметь нужный остаток (т.е. CRC). Но оказывается, что к Q можно прибавить некий многочлен M так, чтобы исправить CRC. Подставив сначала $S_O = Q * X^8 + R$**,** а затем $S_1 = Q + M$ в (2)**,** легко найти, что $M = (X^{32})^{-1} * (1 + (X^8)^{-1}) * (P_{zero} + L_f) + (X^8)^{-1} * R$.

Возвести многочлен P в минус 1-ю степень означает найти такой многочлен P', что $P * P' = 1 \bmod P_{CRC}$, что всегда возможно, так как P_{CRC} неприводим и многочлены с операциями по модулю P_{CRC} образуют поле. Кстати, степень M не превышает 32, так как M достаточно взять по модулю P_{CRC}.

Если перебирать все байты R от 0 до 255 и отправлять их в сеть точке доступа, то, в конце концов, можно наткнуться на правильный. Понять, что попали в байт в реальности также несложно: пакеты с неверным ICV отбрасываются как переданные с ошибкой; если же CRC верна, мы вправе ожидать ответный пакет, например, в случае WPA, это обычно будет пакет, информирующий о неверном MIC.

Тот факт, что пакет шифрован, значения не имеет, так как шифрование RC4 алгоритмом сводится к XOR-иванию данных с keystream'ом, но прибавление M – это тот же XOR, а так как операция XOR коммутативна и ассоциативна, то и неважно, прибавлять M к исходным данным, а потом шифровать, или к уже зашифрованным.

Описанный процесс можно повторять дальше, беря по байту от сообщения, и теоретически можно расшифровать WEP-пакет произвольной длины.

Рассмотрим на практике Chop-Chop атаку на сеть с WPA-защитой. TKIP имеет, в основном, 2 средства для защиты от подобных атак:

а) пакеты с неверной ICV отбрасываются. Если ICV угадана, но неверен MIC-код, атакующий должен выдержать минуту, чтобы не спровоцировать смену ключей;

б) если пакет принят, TSC-счетчик для этого канала (TSC-счетчики существуют отдельно для каждого канала, в которые данное устройство может слать пакеты) увеличивается на 1. Пакеты с меньшим либо равным TSC с этого момента отбрасываются.

Первый пункт частично на руку атакующему, так как позволяет понять, когда он угадал очередной байт. Со вторым сложнее: действительно, пусть перехвачен ARP пакет, посланный одним из устройств в сети, его же получила и Access Point, пересылать этот пакет ей еще раз бессмысленно, ведь TSC уже инкрементирован, плюс то первое устройство, возможно, наслало еще пакетов, увеличив счетчик еще больше. Авторы рассматриваемой атаки нашли выход в спецификации IEEE 802.11e, определяющей улучшения в QoS для сетей Wi-Fi [4]. Сухой остаток здесь таков: устройства Wi-Fi поддерживают несколько очередей пакетов, по словам одного из авторов, Эрика Тьюза, предполагалось использовать 4 канала, в стандарте их 8, в реальности авторы обнаруживали до 16. Каналы обыкновенно не исполь-

зуются одновременно, экономя пропускную способность для важных пакетов. В ненагруженной сети часто весь трафик идет в один канал, таким образом пакет мы скорее всего поймаем на канале с высоким значением счетчика TSC, перепосылать же его можно, переключившись на менее нагруженный канал. Важно, что, опять же ссылаясь на авторов, при отсылке сообщения о неправильном MIC-счетчик канала не увеличивается, таким образом дальше каналы можно не переключать.

Если в сети не поддерживаются QoS расширения, атака, в принципе, также осуществима, если удастся предотвратить попадание выбранного для дешифровки пакета на AP и отключить пославшее его устройство от сети.

Понятно, что пытаться расшифровать сколько-нибудь длинные пакеты по байту в минуту, дело довольно-таки безнадежное – стандартный Wi-Fi пакет имеет размер ~2300 байт, т.е. на него уйдет порядка полутора суток, а за это время или TSC-счетчик увеличится, или ключи сменятся, или перезагрузится точка доступа. Причем, имея одно устройство, ломать можно только один пакет.

Поэтому имеет смысл направлять усилия на расшифровку коротких пакетов, например, ARP, которые легко идентифицировать по их длине (14 байт). Собственно, атакующий знает большую часть содержимого APR-пакета, а именно заголовки и MAC-адреса. Если можно также сделать и некоторые предположения об IP-структуре взламываемой сети (в сети, созданной из под Windows, например, можно ожидать, что IP-адреса будут иметь вид 192.168.0.x), то угадать останется совсем немного. То есть, угадав 12 байт ICV + MIC, остальное можно просто подобрать, используя ICV-сумму.

Расшифровав один пакет, атакующий узнает RC4-keystream пакета (использовать его, правда, можно только, пока TSC не изменился) и, что более важно, сессионный MTK – а MIC обратим, и по данным и подписи можно установить ключ. Последнее означает, что до следующей смены ключей нет необходимости больше угадывать MIC, и, например, чтобы взломать следующий ARP-пакет (с теми же предположениями о структуре сети), уйдет 4-5 минут. Использовать дешифрованные данные для посылки forged-пакетов можно, в зависимости от количества QoS-каналов, от 7 до 15 раз (меньше, если трафик идет по более, чем одному каналу), дальше придется анализировать другой пакет.

Атака позволяет расшифровать отдельные короткие пакеты, с которыми ведется работа, затрачивая в самом лучшем случае по 4-5 минут на пакет, и, используя результаты дешифровки, инжектировать очень ограниченное число столь же коротких пакетов обратно в сеть.

Атака не может быть использована ни для подключения к домашней или корпоративной сети, ни для того, чтобы отслеживать в них трафик.

Тем не менее с её помощью можно «отравить» ARP- и DNS-кеш и прочитать некоторый объем приватного трафика, обмануть некоторые фаерволы.

Хотя шифры и не взламываются окончательно, администраторы беспроводных сетей должны переосмыслить использование WPA и TKIP. Многие компании уже столкнулись с необходимостью модернизации беспроводных сетей для обеспечения их соответствия требованиям стандарта PCI DSS 1.2, который недавно появился для замены WEP в качестве меры защиты в беспроводных сетях, работающих с конфиденциальной информацией.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin*. Breaking 104 bit WEP in less than 60 seconds [Электронный ресурс] / Режим доступа: http://eprint.iacr.org/2007/120.pdf, свободный. – Загл. с экрана.

2. Aircrack manual [Электронный ресурс] / Режим доступа: http://www.aircrack-ng.org/doku.php, свободный. – Загл. с экрана.

3. *Adam Stubble_eld, John Ioannidis, and Aviel D. Rubin.* A key recovery attack on the 802.11b wired equivalent privacy protocol (WEP). ACM Transactions on Information and System Security, 7(2):319{332, May 2004.

4. *Erik Tews.* Attacks on the wep protocol. Cryptology ePrint Archive, Report 2007/471, 2007. [Электронный ресурс] / Режим доступа: http://eprint.iacr.org/. - свободный. – Загл. с экрана.

Половко И.Ю.
Научный руководитель: д.т.н., профессор Макаревич О.Б.
г. Таганрог, ТТИ ЮФУ

Лозунг: Правда о системах обнаружения атак

МЕТОДЫ ТЕСТИРОВАНИЯ ПРОИЗВОДИТЕЛЬНОСТИ СЕТЕВЫХ СОА

Описываются подходы к тестированию производительности сетевых систем обнаружения атак, приводятся результаты тестов и их анализ.

Системы обнаружения атак, СОА, IDS, производительность, эффективность СОА, тестирование СОА.

Цели тестирования производительности

Тесты производительности проводятся в условиях, когда атакующий, жертва и СОА располагаются в одной сети. В этом случае тестам не мешают устройства, ограничивающие пропускную способность. Цель тестов производительности – определить характеристики работы СОА с пакетами. Рис. 1 иллюстрирует топологию для таких тестов.

Методы тестирования производительности

Необходимо проводить несколько серий тестов при различной загрузке сети. Как минимум, необходимы две серии при загрузке 100 Mbps сети 6100 пакетов в секунду (~3%) и 25000 пакетов в секунду (~15%). Во всех тестах должны использоваться пакеты длиной 64 байта. Пакеты, содержащие атаки, генерируются с заданной скоростью 100 пакетов в секунду.

Размеры пакета должны быть выбраны таким способом, чтобы генерировать максимальное количество пакетов в секунду, что обычно является самым жестким режимом работы сетевого анализатора. Размер пакета, при котором отправлено максимальное количество пакетов, равен 101 байту.

Генератор трафика обычно использует всю доступную пропускную способность, не давая возможности передачи другого трафика. Для повышения точности результатов, тесты были проведены несколько раз, после чего были выведены средние значения.

Трафик посылается несуществующему хосту, таким образом исключается любое взаимодействии между рабочими станциями. На втором хосте интерфейс устанавливается в различные режимы и производится захват трафика с использованием Win-Dump/TCPdump в различных конфигурациях. В зависимости от типа теста, захваченные пакеты могут сохраняться на диск, выводиться на экран или пропускаться.

Для получения значения загрузки системы используются следующие программы: netperf в UNIX, task manager в Windows XP. Первая утилита разработана

автором, вторая поставляется вместе с операционной системой. В тестировании были использованы: WinDump версии 4.0; TCPdump версии 3.4, libpcap версии 0.4.

Даже если в испытаниях суметь изолировать воздействие каждой из подсистем (BPF и фильтрация BPF, передача наверх), результаты не способны продемонстрировать производительность каждого компонента. Это происходит из-за различий в архитектуре разных версий и невозможности изолировать каждый компонент от взаимодействия с другими и операционной системой. Самый представительный тест – это тест номер 3, который измеряет производительность "в целом", включая драйвер пакета, libpcap, WinDump, а также операционную систему (обработка ядром, запись данных на диск и т.д). Причина этого в том, что производительность "целой системы" наиболее интересна пользователям.

Критерии оценки для тестов на производительность

Критерии оценки производительности следующие:

а) *Пропускная способность* – позволяет оценить способность СОА перехватывать пакеты, не вызывая их потерю. Для этого теста был использован только чистый трафик, не содержащий атак.

б) *Сборка пакетов* – тест для определения производительности сенсора при сборке пакетов. Для теста использовалась атака TearDrop, в базу сигнатур была загружена сигнатура только этой атаки.

в) *Эффективность фильтрации* – тест предназначен для оценки общей эффективности системы при решении задачи перехвата, разбора пакета и реагирования на атаку. Для тестирования использовалась атака LAND.

г) *Влияние на производительность системы* – тест для оценки влияния работы СОА на загруженность центрального процессора и памяти и общую производительность хоста.

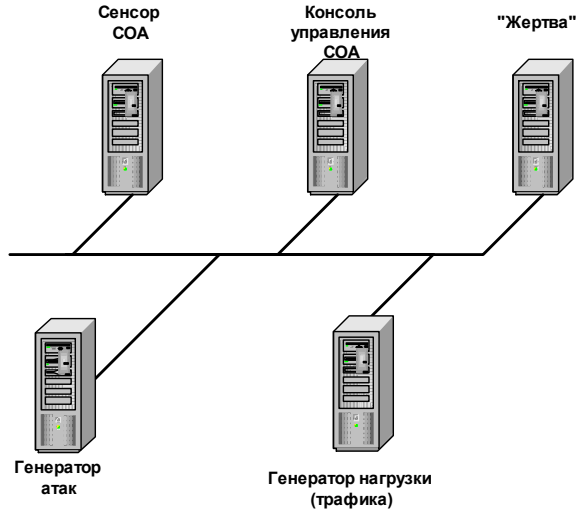


Рис. 1. Топология стенда для проведения тестов производительности

Описание тестов производительности

Тест 1: Производительность фильтра

Этот тест измеряет воздействие BPF-фильтра на процесс захвата. Пакеты, полученные из сети, перехватываются и проверяются фильтром BPF. Фильтр полу-

чает и обрабатывает все посланные пакеты. WinDump/TCPdump запущен со следующей командной строкой:

`tcpdump 'filter'`,

где `'filter'` – фильтр пакетов в формате TCPdump. Этот тест выполняется с двумя различными фильтрами:

– `'udp'`: принимать только UDP пакеты. Это сделано 5-ю командами.

– `'src host 1.1.1.1 and dst host 2.2.2.2'`: принимать пакеты от 1.1.1.1 для 2.2.2.2. Этот фильтр немного более сложен, и устанавливается 13-ю командами.

Так как нет пакетов, удовлетворяющих этому фильтру (потому что все пакеты сгенерированы повреждёнными специальным образом), фильтр отклоняет все пакеты. Таким образом, они не будут скопированы, и не будет взаимодействия с приложением. Только функции фильтрации будут использовать системные ресурсы.

Функция фильтрации не использует память, так что интересным является увидеть использование ею процессора.

Отличия между различным ОС очень ограничены. Это показывает, что реализация BPF в Windows оправдана и вполне способна конкурировать с оригинальным BPF. Загрузка центрального процессора изменяется на различных платформах, однако остается на приемлемых уровнях.

Это из-за того, что NDIS обычно вызывает функцию `packet_tap` прежде, чем DMA-передача пакета закончена, давая небольшое преимущество BPF, в то время как `bpf_tap`, вызывается в конце DMA-передачи.

Значения производительности для двух фильтров очень похожи. Это подтверждает, что машина BPF-фильтрации хорошо оптимизирована и что ее эффективность увеличивается с увеличением длины фильтра.

Тест 2: Производительность драйвера

Для второго теста используется программа перехвата трафика. Это приложение получает все пакеты от драйвера (установка фильтра – принимать все), но дальнейшая обработка их не осуществляется, потому что libpcap – функция `'callback'` пустая. Все пакеты обрабатываются на основных сетевых уровнях, затем драйвером, но на пользовательском уровне обработки нет. Этот тест показывает глобальную оценку эффективности драйвера, включая процесс копирования из интерфейса драйвера в буфер ядра и затем в пользовательский буфер.

UNIX имеет лучшую производительность, чем Windows главным образом потому, что `tap`-функция в UNIX очень проста и быстра, в Windows более сложная и медленная. Для более «длинных» пакетов использование центрального процессора под FreeBSD уменьшается, чего не происходит в Windows. Так получается из-за возможности UNIX BPF “отсроченной записи”. При высокой интенсивности трафика загрузка центрального процессора различных операционных систем подобна. Однако частота системных вызовов (отсюда и загрузка центрального процессора) под UNIX значительно уменьшается, когда размер входящих пакетов увеличивается (т.е. частота уменьшается), в то время как в Windows остается устойчивой.

Такое поведение – не проблема для Windows – реализации, потому что время ЦП используется только, когда есть возможность. Все системы, теряют малое количество пакетов.

Тест 3: Производительность захвата трафика

Это самый важный тест, потому что измеряет полный процесс сбора данных, для чего используется tcpdump. Фильтр не определен. Захваченные пакеты сохраняются на диск, для чего используем опцию `“-w”` tcpdump.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Thomas H. Ptacek, Timothy N. Newsham [Электронный ресурс] / Режим доступа: http://insecure.org/stf/secnet_ids/secnet_ids.html, свободный. – Загл. с экрана.
2. Network Based Intrusion Detection. A review of technologies. [Электронный ресурс] / Режим доступа: http://linkinghub.elsevier.com/retrieve/pii/S016740489980131X, свободный. – Загл. с экрана.
3. Benchmarking network IDS. [Электронный ресурс] / Режим доступа: http://archives.neohapsis.com/archives/sf/ids/2000-q4/0244.html, свободный. – Загл. с экрана.

Сергеев Д.В.
Научный руководитель: д.т.н., профессор Аграновский А.В.
г. Ростов-на-Дону, ФГНУ НИИ “Спецвузавтоматика”

Лозунг: Обман тоже защита

ТЕХНОЛОГИЯ HONEYPOT – ПРИМАНКА ДЛЯ ЗЛОУМЫШЛЕННИКОВ

В статье раскрываются основные аспекты использования технологии honeypot в современных компьютерных сетях. Исходя из значимости, размещения в сети, преимуществ и недостатков средств honeypot, выясняется, какую роль они играют в обеспечении информационной безопасности.

Информационная безопасность, компьютерная сеть, технология honeypot, обнаружение вторжений.

Возросшая сложность компьютерных сетей, увеличение количества уязвимостей в программном и аппаратном обеспечении, а также возможностей по реализации сетевых атак, обуславливают необходимость применения интеллектуальных автоматизированных механизмов защиты. Классические средства обеспечения безопасности, такие как межсетевые экраны, системы обнаружения и предотвращения вторжений, антивирусы, механизмы шифрования, призваны решать конкретные задачи. Защита этих средств основана на предупреждении или принятии контрмер против наиболее распространенных видов атак. Технология honeypot (от английского honey pot – "бочонок с медом"), являясь более гибким инструментом, позволяет взглянуть на проблемы обеспечения безопасности с другой стороны.

Под технологией honeypot будем понимать идею, заложенную в основу информационной системы, задача которой подвергнуться атаке или несанкционированному исследованию, что позволит изучить стратегию злоумышленника и определить круг средств, с помощью которых могут быть нанесены удары по реальным объектам безопасности [1]. Такую информационную систему будем называть системой-приманкой. Также технология honeypot включает в себя совокупность методов и средств, применяемых для обеспечения безопасности компьютерной сети в соответствии с вышеописанной идеей. Понятие "средство honeypot" будем применять для обозначения ресурса компьютерной сети, используемого для достижения результата при описании, построении и реализации систем-приманок.

Реализация средств honeypot не принципиальна, это может быть как полноценная операционная система, развернутая на специально выделенном сервере, так и имитируемый сетевой сервис, запущенный на рабочей станции. Смысл функционирования такого ресурса сети заключается в привлечении внимания зло-

умышленников с последующим изучением его поведения, используемых технологий и инструментов.

Средствам honeypot также не требуется какая-нибудь особая поддержка, и устанавливаться они могут в зависимости от того, поведение какого типа злоумышленников (внутренние, внешние) необходимо изучить (рис. 1).

Размещение средств honeypot перед межсетевым экраном позволяет протоколировать действия злоумышленников, действующих из сети Интернет. Очевидно, размещение средств honeypot в локальной сети предоставляет возможность контролировать атакующие действия внутри этой сети. Размещение же средств honeypot в демилитаризованной зоне (DMZ) используется для фиксирования подозрительной активности, проявляемой как со стороны внешних, так и внутренних злоумышленников в этом сегменте сети.

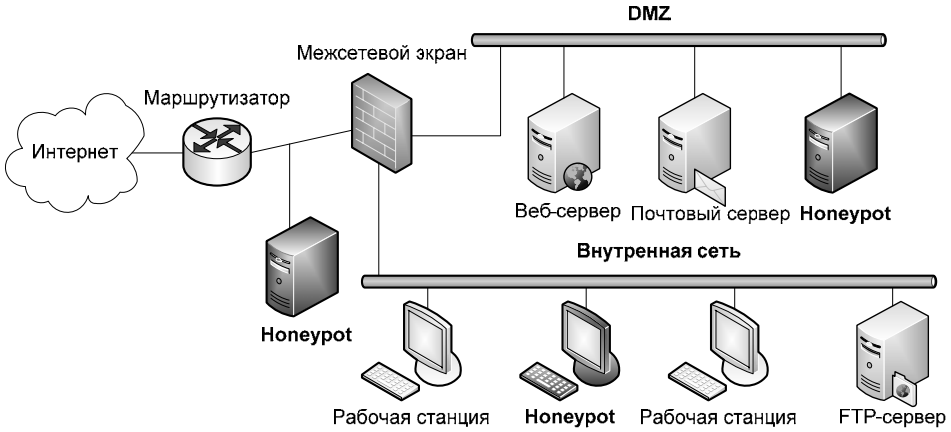


Рис. 1. Размещение средств honeypot в компьютерной сети

Для того чтобы оценить значимость технологии honeypot, рассмотрим модель информационной безопасности, предложенной Брюсом Шнайером [2], которая подразумевает три уровня защиты: предотвращение, обнаружение и ответная реакция. Информационные системы-приманки могут быть задействованы на всех трех уровнях. Так, например, на уровне предотвращения системы-приманки применяются для сдерживания злоумышленников путем обмана, что ведет к замедлению или полной остановке вторжения, атаки или дальнейшего исследования компьютерной сети. Используются системы-приманки и для обнаружения несанкционированной деятельности, при этом они вносят наиболее ощутимый вклад в решение именно такого рода проблем. Это связано с тем, что процесс обнаружения достаточно трудоемок и с ним связаны такие понятия, как ложные срабатывания (ошибки первого рода), пропуски (ошибки второго рода) и агрегация данных. Традиционные решения в области безопасности (системы обнаружения и предотвращения вторжений) способны генерировать огромное количество журнальных записей, среди которых только несколько отображают реальные попытки исследования или проникновения в сеть. Определение факта атаки заложено в идею технологии honeypot – к средству honeypot не могут обращаться легитимные пользователи. Если к нему обратились, то это уже свидетельствует о подозрительной активности. Таким образом, минимизируется вероятность возникновения ошибок первого и второго рода.

Средства honeypot создают достаточно небольшое количество полезной информации, которая имеет высокое значение, что значительно упрощает агрегацию данных и последующий анализ полученной информации. Еще одним положительным моментом использования средств honeypot в процессе обнаружения является возможность идентифицировать неизвестные типы атак.

Системы-приманки используются и для ответной реакции на атаки. Если, например, злоумышленник проник в компьютерную сеть, и одна из атакованных рабочих станций оказалась средством honeypot, то полученная информация применяется для ответной реакции на атаку.

Описанные преимущества использования средств honeypot могут вызвать иллюзию того, что системы-приманки являются идеальным инструментом для обеспечения безопасности. Однако в силу ряда недостатков, присущих информационным системам такого рода, они могут служить дополнением к комплексу средств защиты компьютерных сетей, расширяя архитектуру безопасности. Среди характерных недостатков следует отметить следующие [1]:

- ограниченная область видения;
- возможность раскрытия;
- риск взлома.

Самым большим недостатком технологии honeypot является ее узкая область видения, т.е. средства honeypot осуществляет мониторинг только той активности, которая направлена против них. Если злоумышленнику удается проникнуть в компьютерную сеть, и его действия направлены на различные системы, то средства honeypot не обнаружат этой деятельности, если она не направлена непосредственно против них.

Другим недостатком систем-приманок является возможность раскрытия их злоумышленником. Причиной обнаружения систем-приманок является тот факт, что злоумышленник имеет определенные ожидаемые характеристики или особенности поведения сетевого ресурса. Злоумышленники, заподозрив обман, могут отказаться от дальнейших действий или попытаться скрыть следы присутствия, или, наоборот, начать активно взаимодействовать с системами-приманками, предоставляя ложную информацию.

Третьим значимым недостатком технологии honeypot являются риски взлома. Хотя средства honeypot специально подставляются злоумышленникам для взлома, однако в случае успешно проведенной против них атаки, они могут использоваться как плацдарм при нападении на другие сетевые ресурсы в компьютерной сети.

Помимо сугубо практического применения средств honeypot, описанного выше, не менее важен другой аспект – исследовательский. Среди основных направлений применения систем-приманок в исследовательских целях выделяются следующие [3]:

- отвлечение внимание злоумышленника от реальной сети, что позволяет снизить вероятность угрозы на основные информационные ресурсы;
- обнаружение новых типов вирусов и червей для дальнейшего изучения;
- построение профилей злоумышленников, с тем, чтобы выявлять характерные методы, технологии и инструменты нападения;
- выявление новых уязвимостей в различных операционных системах, сервисах и программах.

Для сбора полезной информации средства honeypot предоставляют возможность отследить весь процесс атаки, вплоть до фиксирования нажатия клавиш на клавиатуре. Это является важным моментом при администрировании компьютерной сети, так как позволяет, с одной стороны, принять меры предосторожности, с

другой – сохранить документальные следы воздействия, которые в дальнейшем могут быть использованы для проведения расследования.

Технология honeypot имеет свои преимущества и недостатки. Она являются полезным средством безопасности для отслеживания действий злоумышленников, сбора информации и генерирования предупреждений в случае, когда кто-либо взаимодействует с ними.

Значимость средств honeypot и проблемы, которые они помогают решать, зависят от того, как данные средства реализованы, развернуты и используются.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Lance Spitzner* Honeypots:Tracking Hackers, Addison-Wesley, 2002, 480 pages.
2. *Bruce Schneier* Secrets and Lies, John Wiley & Sons, 2000, 432 pages.
3. *Pouget, F., Holz, T.* A pointillist approach for comparing honeypots, Intrusion and malware detection and vulnerability assessment, Berlin / Heidelberg: Springer, 2005.

Филинова С. В.
Научный руководитель: Горшенин В.В.
г. Челябинск, ЧГУ

ГРАФИЧЕСКАЯ АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

В статье описывается практическое исследование удобства использования и запоминаемости графических паролей по сравнению с алфавитно-цифровыми. Для исследования была взята схема графических паролей PassPoints, разработанная Вайденбеком. Пароль в этой системе представляет последовательность кликов мышки по изображению. Участники исследования должны были создать пароль и потренироваться его вводить, а через неделю после этого ввести созданный ранее пароль еще раз. Результаты исследования показали, что графические пароли легче запоминаются и их проще создать, но ввод такого пароля занимает больше времени. Исходя из полученных результатов, был сделан вывод, что система графических паролей PassPoints является вполне жизнеспособной в плане удобства использования, безопасности и запоминаемости паролей.

Графические пароли, алфавитно-цифровые пароли, PassPoints, аутентификация, безопасность паролей, запоминаемость паролей.

Ранее проблема компьютерной и сетевой безопасности рассматривалась только с технической стороны. Но многие механизмы защиты предполагают взаимодействие с пользователем. Одной из важнейших областей, где взаимодействие человек-компьютер существенно, является процедура проверки подлинности. Наиболее распространенным методом аутентификации в компьютерной системе является механизм, основанный на том, что пользователь вводит имя пользователя и текстовый пароль. При этом пароль должен удовлетворять двум противоречивым требованиям:

1) пароли должны легко запоминаться, а процедура аутентификации должна быть удобной и не занимать много времени;

2) пароли должны быть безопасными, т.е. выглядеть случайно и быть сложными для угадывания, они должен периодически меняться и быть разными на разных системах для одного пользователя, они не должны быть записаны или сохранены в открытом виде.

Посколько людям трудно запомнить случайные последовательности, условия безопасности паролей часто нарушаются. Исследования показали, что пользователь предпочитает выбирать короткие пароли, либо пароли, которые легко запомнить. К сожалению, эти пароли легко отгадать или взломать. Согласно недавней статье журнала Computerworld, команда одной крупной компании, отвечающая за безопасность, запустила сетевую программу взлома паролей и за 30 секунд было раскрыто 80% паролей. В то же время, те пароли, которые трудно отгадать или взломать, запоминаются с трудом. Из исследований стало ясно, что пользователь может запомнить лишь ограниченное количество паролей, и поэтому он обычно записывает их на бумаге или использует одни и те же пароли для различных аккаунтов.

Для решения проблем, связанных с традиционной проверкой подлинности (имя пользователя – пароль), используются альтернативные методы проверки подлинности, такие как биометрика или аппаратные ключи и смарт-карты, но несмотря на это алфавитно-цифровые пароли остаются доминирующим методом аутентификации из-за проблем с надежностью, неприкосновенностью частной жизни, ценой и сложностью использования других технологий. В данной статье речь пойдет о другой альтернативе текстовым паролям: использовании в качестве паролей изображений.

Схемы графического паролирования были предложены в качестве альтернативы для схем паролирования на основе текста – это было мотивировано отчасти тем, что изображения человек запоминает лучше, чем текст; психологические исследования подтверждают это предположение [3]. Изображения обычно легче запоминать и узнавать, чем текст. Кроме того, если число возможных изображений достаточно велико, то возможное пространство паролей в схеме графического паролирования может превышать соответствующее пространство для схем на основе текста, и, вероятно, таким образом обеспечивать лучшую защиту против словарных атак.

Целью данной работы было исследование запоминаемости и простоты использования графических паролей по сравнению с алфавитно-цифровыми. Этой работой я попыталась ответить на два вопроса:

1) являются ли графические пароли жизнеспособной альтернативой алфавитно-цифровым в плане безопасности, а также простоты создания пароля, его запоминания и удобства использования

2) каково мнение пользователей о графических паролях по сравнению с текстовыми?

Обзор существующих схем графической аутентификации

Большинство схем графических паролей основываются или на узнавании или на последовательном вспоминании пароля. В системах, основанных на узнавании, пользователь должен узнать предварительно выбранные изображения из большей группы отвлекающих изображений. В системах, основанных на последовательном вспоминании пароля, пользователь должен кликнуть мышкой по некоторой заранее определенной последовательности областей парольного изображения.

Например, в парольной схеме PassFaces [4], чтобы создать пароль, пользователь выбирает 4 фотографии из большого перечня лиц. При входе в систему пользователь видит таблицу 3x3 с девятью лицами, состоящую из одного лица, выбранного пользователем, и восьми отвлекающих лиц. Пользователь должен узнать заранее выбранное лицо и щелкнуть по нему. Эта процедура повторяется в течение четырех раундов для каждого лица, составляющего пароль. Если пользователь выбирает все 4 лица правильно, то он будет допущен в систему. Эксперименты

показали [5], что пароли в системе PassFaces являются более запоминающимися, чем алфавитно-цифровые пароли. Однако недостатком данного метода является то, что в одном раунде может быть показано достаточно небольшое количество изображений, только одно из которых является правильным. Поэтому нападающий имеет шанс угадать изображение 1 к 9. Чтобы уменьшить этот шанс, можно увеличить количество раундов узнавания изображения, однако это повлечет за собой увеличение времени ввода пароля, что является нежелательным. Кроме того, использование лиц в качестве изображения привело к паролям с очень низкой энтропией, потому что люди выбирают лица достаточно предсказуемым образом.

Джеремин предложил схему под названием “Draw-a-Secret” [6], предназначенную для устройств PDA. В этой схеме пользователь рисует пароль на сетке. Чтобы воспроизвести пароль, каждая непрерывная линия рисуется на сетке, тем самым перечисляя ячейки, которые она проходит, в порядке, в котором она пересекает границу ячейки. Точное повторение позволяет сохранить пароль в виде хэша. Данная система получила большее распростаранение в мобильных устройствах, так как рисовать пароль намного удобней при помощи стайлуса, чем мыши, используемой в стационарных компьютерах.

Графические пароли, основанные на последовательном воспроизведении, впервые были предложены Блондером [7]. В этой схеме пользователь выбирает несколько местоположений на изображении, чтобы создать пароль. Чтобы загрузиться, пользователь должен кликнуть близко к выбранным точкам. Нет никаких многократных раундов, только единственное изображение.

Для исследования была выбрана схема PassPoints, основанная на идее Блондера и впервые предложенная Вайденбеком [1]. Пароль в данной системе представляет собой последовательность кликов мышкой по областям изображения. Она является достаточно гибкой, так как для пароля можно использовать практически любое изображение, оно может как предоставляться системой, так и быть загруженным пользователем. Единственной рекомендацией к изображению является то, что оно должно содержать достаточно большое количество деталей, чтобы пользователю было проще создать и запомнить пароль. Так как изображение содержит множество точек, то соответственно пространство возможных паролей расширяется по сравнению с текстовыми паролями. Таким образом, мы можем создать больше графических паролей, состоящих из 5-6 кликов, чем текстовых длиной 8 символов, что увеличивает стойкость системы к словарным атакам.

В целях безопасности пароли не должны храниться и передаваться в открытом виде. Сначала может показаться, что к графическим паролям невозможно применять хеширование, так как при вводе пароля пользователь не может кликнуть мышкой по одним и тем же точкам изображения, он выбирает точки в некоторой окрестности. Таким образом, координаты точек пароля не будут точно совпадать при каждом вводе, и значения хэш-функций будут различаться. Для решения этой проблемы необходимо использовать дискретизацию, когда все изображение разбивается на квадраты достаточного размера, чтобы пользователь при каждом вводе выбирал один и тот же квадрат. В качестве метода дискретизации был выбран метод, описанный Бирджетом [2]. Особенностью данного метода является то, что на изображение накладывается ни одна сетка для дискретизации, а три, что помогает решить «проблему краев», когда пользователь выбирает точку пароля, близкую к краю квадрата и при дальнейшем вводе пароля с большой вероятностью может попасть в соседний квадрат, что повлечет к некорректному результату. Использование трех сеток помогает решить данную проблему, в такой системе любая точка будет достаточно удалена от краев квадратов как минимум в одной из сеток.

Исследование

В рамках исследования запоминаемости и удобства использования графических паролей было проведено тестирование пользователей.

Порядок тестирования

Процедура тестирования графических и алфавитно-цифровых паролей была одинаковой и состояла из двух этапов. На первом этапе участникам тестирования предлагалось создать пароль и потренироваться его вводить. При этом замерялись такие величины, как время, затраченное на создание пароля, время на тренировку, количество ошибок создания и ввода пароля. После первого этапа пользователи проходили тестирование, помогающее оценить их впечатление от работы с парольной системой.

Второй этап тестирования состоялся через неделю. Участники должны были вспомнить созданный ранее пароль и ввести его. При этом было измерено количество попыток, потребовавшееся для ввода правильного пароля и затраченное на это время.

Участники тестирования

Было протестировано 36 человек возрастом от 20 до 55 лет (средний возраст 25 лет), использующих компьютер каждый день. Из них 9 мужчин и 27 женщин. 40% участников имело гуманитарное образование, 60% – техническое.

Предмет тестирования

Для проведения тестирования было написано 4 программы: 2 – для тестирования графических паролей, 2 – для текстовых. Интерфейс программы тестирования схемы PassPoints включает в себя цветное изображение размером 800x600 пикселей и несколько управляющих кнопок (рис. 1), размер ячейки сетки дискретизации или окрестность точки составляла 20 пикселей, вверху экрана отображались инструкции для прохождения тестирования. Следует отметить, что окрестность в 20 пикселей выбрана не случайно, при таком параметре пространство возможных графических паролей из пяти кликов будет примерно совпадать с пространством алфавитно-цифровых паролей из 8 символов. Очевидно, что размер пространства графических паролей можно вычислить по формуле (A*B/R)^n, где A и B – размеры изображения, R – размер окрестности точки, n – длина пароля, в нашем случае это будет $(800*600/20)^5 = 2{,}4*10^{15}$, когда для алфавитно-цифровых паролей это значение составит $(94)^8 = 6*10^{15}$. Итак, данное условие позволяет нам сравнивать удобство использования и запоминаемость графических и текстовых паролей при примерно одинаковом их уровне безопасности.

Результаты

Создание пароля

Во время создания пароля участники должны были придумать и ввести пароль, удовлетворяющий условиям тестирования. Графический пароль должен был состоять из 5 символов, а текстовый – из 8, при этом содержать как минимум одну строчную букву, одну прописную букву и одну цифру. Количество попыток и время, затраченное на создание пароля, было замерено. Результаты показали, что придумать текстовый пароль оказалось сложнее, чем графический: 31 из 36 человек потребовалось 2 и более попыток, чтобы создать правильный пароль. Графический же пароль с первого раза правильно ввели 33 человека. Однако среднее время на создание графического пароля оказалось больше, чем алфавитно-цифрового. Результаты по количеству ошибок и времени создания пароля приведены в табл. 1 и на рис. 2.

Рис. 1. Интерфейс программы тестирования графических паролей

Таблица 1

Этап создания пароля

	Режим	Среднее значение
Общее число попыток создания пароля	Графические Текстовые	1,2 2,3
Общее время создания пароля (в секундах)	Графические Текстовые	64 52

Тренировка

На этапе тренировки участники практиковались во введении паролей с целью их запоминания. Для этого они последовательно вводили свой пароль, пока не введут его правильно 5 раз. Количество участников, не допустивших ни одной ошибки при работе графическими и текстовыми паролями, оказалось равным, но 2 человека допустили более 4 ошибок при введении графического пароля.

Таблица 2

Этап тренировки

	Режим	Среднее значение
Число ошибок при вводе пароля	Графические Текстовые	0,7 0,4
Общее время тренировки	Графические Текстовые	166 72

Рис. 2. Число ошибок, допущенных при создании пароля

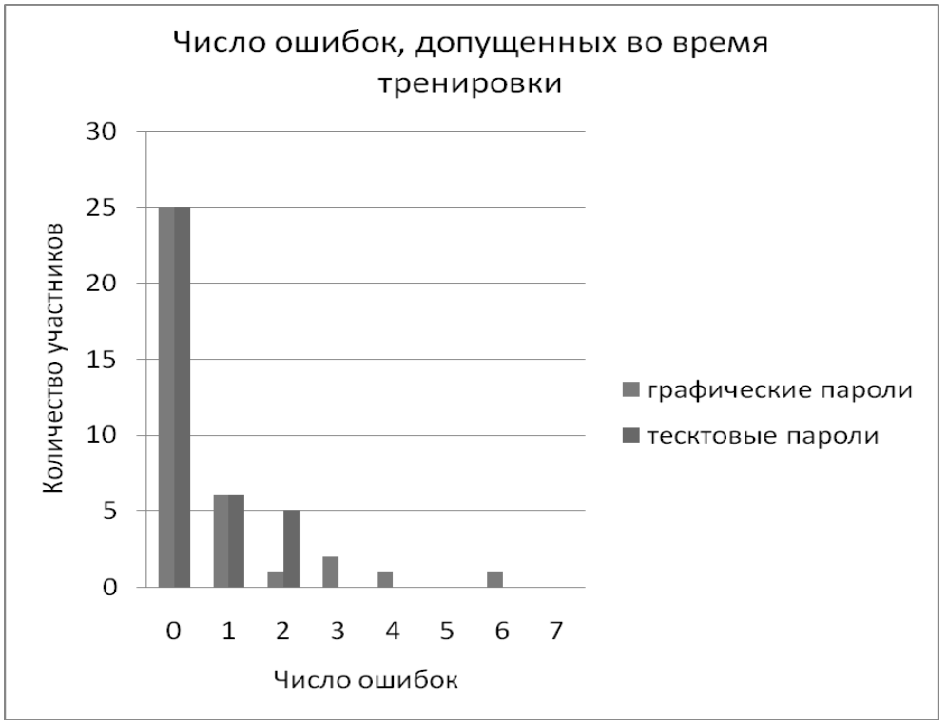


Рис. 3. Число ошибок, допущенных во время тренировки

Ввод пароля через неделю

Через неделю после создания участникам тестирования было предложено ввести пароль. Количество человек, у которых не получилось ввести графический пароль с первого раза – 2 (посмотрели пароль 3 человека), текстовый – 9 (посмотрели – 7). Среднее время введения графического пароля составило 79 с, текстового – 28.

Проведенное исследование показало, что графический пароль проще создать, чем алфавитно-цифровой. Однако это требует больше времени. Ввод текстового пароля также происходит значительно быстрее, чем графического, при этом участники допускали меньше ошибок, хотя разница в количестве ошибок ввода была незначительной. Здесь стоит учесть, что схема графической аутентификации была совершенно новой для пользователей и после некоторой практики они смогут быстрей вводить графический пароль и допускать при этом меньше ошибок. Также результаты тестирования показали, что запоминаемость графических паролей, как и предполагалось, оказалась лучше, чем текстовых. Судя по ответам на параметрический тест в конце тестирования, пользователи не ощущали значительных неудобств при вводе пароля, а также оценили время ввода как более чем приемлемое.

Исходя из всего вышеперечисленного, можно сделать вывод, что данная система является вполне жизнеспособной в плане удобства использования, безопасности и запоминаемости паролей. В дальнейшем планируется продолжить исследование пользователей, а также встроить данную систему в Windows/Linux как альтернативу стандартному методу аутентификации.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, Nasir Memon* “PassPoints: Design and longitudinal evaluation of a graphical password system”

2. *R. N. Shepard,* "Recognition memory for words, sentences, and pictures"

3. Real user “http://www.realusers.com”

4. *S. Brostoff and M. A. Sasse*, "Are Passfaces more usable than passwords: a field trial investigation"

5. *I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. "D. Rubin*, "The Design and Analysis of Graphical Passwords"

6. *Blonder, G.E.* Graphical passwords. United States Patent (1996).

7. *Xiaoyuan Suo, Ying Zhu, G. Scott.* Owen Graphical “Passwords: A Survey” Department of Computer Science Georgia State University

Майзаков К.Д., Эдель Д.А.

Научный руководитель: д.т.н., профессор Аграновский А.В.

г. Ростов-на-Дону, ФГНУ НИИ “Спецвузавтоматика”

Лозунг: Кто запутывает лучше?!

АНАЛИЗ МЕТОДОВ ЗАПУТЫВАЮЩИХ ПРЕОБРАЗОВАНИЙ ПРОГРАММ НА УРОВНЕ ИСХОДНЫХ ТЕКСТОВ

В статье определена формальная постановка задачи запутывающих преобразований на уровне исходного текста, предложен критерий оценки запутывающих преобразований и проведен анализ запутывающего программного обеспечения на основе введенного критерия.

Запутывание, обфускация, оценка качества обфускации, обфускация исходных текстов, языки на сценариях.

Во всех сферах деятельности человек использует различные программные средства. Любое программное средство является интеллектуальной собственностью. В связи с этим актуальна проблема обеспечения защиты программных средств и их исходного кода [1].

В языках сценариев таких, как PHP [2], Perl [3], Ruby [4], JavaScript [5], VBScript [6], данная проблема особенно важна, так как при работе программы, пользователю полностью доступен ее исходный текст.

Для защиты исходного кода программы, написанной на языке сценариев и, как следствие, для защиты интеллектуальной собственности программиста, применяются методы запутывания (затемнение, обфускация) на уровне исходных текстов.

Запутанной называется программа, которая на всех допустимых для исходной программы входных данных выдаёт тот же результат, что и оригинальная программа, но ее исходный код более труден для анализа, понимания и модификации.

Чтобы получить запутанную программу, необходимо к исходной программе применить запутывающие преобразования.

Существуют и другие методы запутывающих преобразований программ, как правило [7], они разделяются на три группы:

- методы запутывания исходного кода программ;
- методы запутывания байт-кода программ;
- методы запутывания исполняемого кода.

В данной статье вводится критерий оценки эффективности работы программ запутывания на уровне исходных текстов и производится сравнение существующих программ-обфускаторов на основе введенного критерия.

Существует большое количество программ как коммерческих, так и с открытым кодом, производящих запутывание программ на уровне исходных текстов. Запутывание на уровне исходных текстов особенно эффективно для интерпретируемых языков программирования, так как. современные компиляторы автоматически оптимизируют код, что уменьшает эффективность обфускации исходных текстов компилируемых языков. В табл. 1 представлены данные о существующих, наиболее популярных программах, выполняющих запутывание исходного кода, а также о технологиях [7], которые используются для лексической обфускации, обфускации потока данных и для обфускации потока управления.

В основе технологий, применяемых для запутывания программ на уровне исходного кода, лежит алгоритм, приведенный на рис. 1:

Шаг 1. Удаление комментариев, переносов строк, лишних пробелов, переименование переменных трудноразличимыми последовательностями символов (лексическая обфускация).

Шаг 2. Кодирование строковых и числовых констант, преобразования размерности массивов, изменения интерпретации типов данных (обфускация потока данных).

Шаг 3. Реструктуризация циклов, добавление недостижимого, избыточного или мусорного кода, реорганизация переходов [9,10] (обфускация потока управления).

Таблица 1

Программы обфускации на уровне исходных текстов

№	Название программы-обфускатора	Использование технологий лексической обфускации			Использование технологий обфускации потока данных			Использование технологий обфускации потока управления	Шифрование
		Удаление комментариев	Удаление пробелов	Переименование идентификаторов	Кодирование строк	Кодирование чисел	Слияние и разбиение строк		
1	PHP Replace	+	+	+	+	+	+	-	-
2	PHP Obfuscator	+	+	+	+	-	-	-	-
3	Free Javascript Obfuscator http://javascriptobfuscator.com	+	+	+	+	+	+	-	-
4	JavaScript Obfuscator & Encoder	+	+	+	+	+	-	-	+
5	The Perl Obfuscation Service http://liraz.org	+	+	+	+	-	-	-	-

Приведем формальную постановку задачи запутывания исходного кода программы.

Примем за оценку запутанности кода время, которое потребуется аналитику для восстановления исходного кода программы, исходного графа потока управления и графа потока данных модифицированной программы. Сложность анализа исходного текста оригинальной программы (которую обозначим за «x») примем за A(x), а сложность анализа запутанной – A(y), где «y» – исходный текст запутанной программы.

Коэффициент сложности запутанного исходного кода программы определяется как отношение оценки сложности анализа исходного кода программы к оценке сложности анализа запутанного кода программы:

$$K(x, y) = \frac{A(x)}{A(y)}. \quad (1)$$

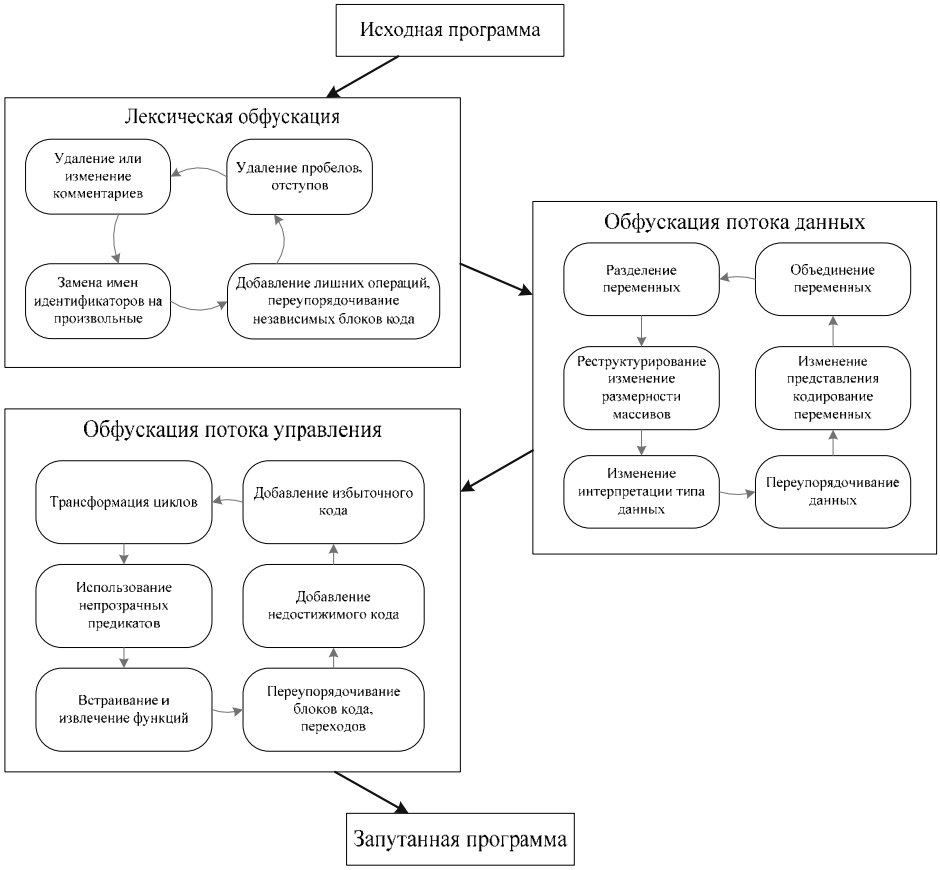


Рис. 2. Алгоритм запутывающих преобразований на уровне исходных текстов

Так как для анализа запутанного кода программы аналитику потребуется время, значительно большее, чем для анализа исходного кода, то A(y)>A(x). Отсюда $K(x,y)=\frac{A(x)}{A(y)}\in[0,1]$. Аналогично определим L(x,y) – коэффициент запутывания потока данных, M(x,y) – коэффициент запутывания потока управления.

Назовем коэффициентом запутывания программы величину

$$O(x,y)=K(x,y)*L(x,y)*M(x,y)\in[0,1]. \qquad (2)$$

Формально, O(x,y) представляет собой величину, равную отношению времени работы аналитика при восстановлении логики работы оригинальной программы ко времени работы при восстановлении логики запутанной программы. Чем больше эта величина, тем проще восстановить программу, тем слабее запутывающие преобразования.

Пусть дано S={I, F, D} – множество исходных текстов, где I – все множество имен переменных, F – множество операторов, D – множество данных и пусть $x\in S$ – исходный текст оригинальной программы, $y\in S$ – исходный текст запутанной программы. Обозначим через $o:S\to S$ запутывающее преобразова-

ние программы, если $\forall x \in S\ \exists! y \in S : o(x) = y, A(y) > A(x)$. Таким образом, получили математическую модель запутывающего преобразования исходного кода программы: преобразование $o : S \to S$ будем считать запутывающим, если для любого исходного кода *x* будет существовать единственное его отображение *y* во множестве запутанных текстов программ, причем A(y)>A(x). Тогда (2) примет вид

$$O(x,y) = K(x,y) * L(x,y) * M(x,y) = K(x,o(x)) * L(x,o(x)) * M(x,o(x))$$
$$x = i \bigcup f \bigcup d, i \in I, f \in F, d \in D, i \bigcap f \bigcap d = \varnothing$$
$$O(x,y) = K(x,o(i)+o(f)+o(d)) * L(x,o(i)+o(f)+o(d)) * M(x,o(i)+o(f)+o(d)) =$$
$$= (K(x,o(i)) + K(x,o(f)) + K(x,o(d))) * (L(x,o(i)) + L(x,o(f)) + L(x,o(d))) *$$
$$* (M(x,o(i)) + M(x,o(f)) + M(x,o(d))) =$$
$$K(x,o(i)) + L(x,o(d)) + M(x,o(f)). \qquad (3)$$

Проводя аналогичные рассуждения для x, получаем, что (2) эквивалентно выражению

$$O(x,y) = K(i,o(i)) * L(d,o(d)) * M(f,o(f)). \qquad (4)$$

То есть сложность восстановления имен идентификаторов не зависит от обфускации данных и операторов, одновременно сложность восстановления потока данных и потока управления зависит только от обфускации данных и операторов соответственно.

Таким образом, в соответствии с поставленной задачей обфускации, требуется построить функцию $o : S \to S$, при которой коэффициент запутывания программы

$$O(x,y) = K(i,o(i)) * L(d,o(d)) * M(f,o(f)) \to 0, \qquad (5)$$

при размере исходного текста оригинальной программы $|x| \to \infty$. Иначе говоря, сложность анализа возрастает с ростом размера запутанной программы.

Проведем анализ описанных в табл. 1 программ автоматического запутывания кода на основе введенного критерия. На основе серии экспериментов (анализа запутанных тестовых программ), были получены результаты, показавшие, что при использовании всех методик запутывания, указанных в табл. 1, сложность восстановления кода аналитиком возрастает приблизительно в 10 раз по каждому из направлений (восстановление идентификаторов, восстановление данных, восстановление потока управления). Таким образом при эффективной обфускации Kэфф.(x,y)=0.1, Lэфф.(x,y)=0.1, Mэфф.(x,y)=0.1.Поскольку рассмотренные программы не производят обфускации потока управления, то $M_i(f, o(f)) = 1, i = \overline{1..5}$. Положим коэффициент сложности запутанного исходного кода при использовании программой всех трех указанных в табл. 1 методик лексической обфускации K(x,y)=Kэфф.(x,y)=0.1. Если программа не использует одну из методик, коэффициент K(x,y) возрастает на 0,3.

Аналогично конкретизируем значения коэффициента обфускации потока данных L(x,y) и оценим программы из табл. 1:

$$O_{PHP\ replace}(x, y) = 0.1 * 0.1 * 1 = 0.01$$
$$O_{PHP\ Obfuscator}(x, y) = 0.1 * 0.7 * 1 = 0.07$$
$$O_{Free\ Javascript\ Obfuscator}(x, y) = 0.1 * 0.1 * 1 = 0.01$$
$$O_{Javascript\ Obfuscator\ \&\ Encoder}(x, y) = 0.1 * 0.4 * 1 = 0.04$$
$$O_{The\ Perl\ Obfuscation\ Service}(x, y) = 0.1 * 0.7 * 1 = 0.07$$

Таким образом, на основе введенного критерия оценки, лучшими из рассмотренных программ по качеству обфускации программного кода являются PHP Replace и Free Javascript Obfuscator для языков программирования PHP и Javascript соответственно. Однако теоретически эффективный обфускатор должен иметь коэффициент сложности запутанного исходного кода, коэффициент запутывания потока данных и коэффициент запутывания потока управления не выше 0,1, и, следовательно, коэффициент запутывания программ:

$$O(x, y) \le 0.1 * 0.1 * 0.1 = 0.001 ,$$

что говорит о неэффективности проверенных программ.

Несмотря на то, что теоретически доказано [11], что универсального запутывающего преобразования не существует, активно продолжают разрабатываться новые методы запутывания кода.

В данной статье проведен анализ существующих методов запутывания исходного кода программы, которые применяются для защиты интеллектуальной собственности программиста, написавшего программу, также проведена оценка существующих программ, которые выполняют запутывание исходного кода, сделан вывод о теоретической неэффективности существующих программных решений. Таким образом, вопрос построения метода запутывающего преобразования, удовлетворяющего строгим требованиям качества обфускации, с достаточной теоретической стойкостью, все еще остается открыт.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гражданский кодекс Российской Федерации. Часть четвертая. Права на результаты интеллектуальной деятельности и средства индивидуализации. N 230-ФЗ от 18.12.2006
2. PHP: Hypertext Preprocessor // www.php.net
3. The Perl Directory - perl.org // www.perl.org
4. Ruby Programming Language // www.ruby-lang.org
5. JavaScript.com (TM) - The Definitive JavaScript Resource // www.javascript.com
6. VBScript // msdn.microsoft.com/en-us/library/t0aew7h6.aspx
7. *Гайсарян С.С., Иванников В.П., Аветисян А.И.* Анализ и трансформация программ // www.ict.edu.ru/ft/005642/62319e1-st06.pdf
8. *Munson J., Kohshgoftaar T.* Measurement of data structure complexity // J. of Systems and Software. – 1993. – N. 3. – P. 217-225.
9. *Collberg C., Thomborson C., Low D.* Manufacturing Cheap, Resilient and Stealthy Opaque Constructs // Proc. of 25th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, 1999. – P. 184 – 196.
10. *Иванов И. Ю.* О проблемах защиты интеллектуальной собственности в программных системах // Проблеми програмування, 2006. – № 2-3. – Стр. 580 – 584.
11. *Barak B., Goldreich O., Impagliazzo R.* On the (Im)possibility of Obfuscating Programs // Proc. of the 21st Annual Intern. Cryptology, Conf. on Advances in Cryptology, 2001. - Vol. 2139. - P. 1 – 18.

Благодаренко А.В.
Научный руководитель: д.т.н., профессор Макаревич О.Б.
г. Таганрог, ТТИ ЮФУ

Лозунг: Кто к нам с мечем прийдет - тот от меча и погибнет!

ПАССИВНЫЙ HONEYPOT НА АНАЛИЗЕ СПУТНИКОВОГО ТРАФИКА

В статье предлагается принципиально новый способ построения системы honeypot – системы сбора информации о злонамеренной активности в сети. Уникальной особенностью предлагаемого метода является получение «живого» трафика посредством перехвата данных из спутникового канала, а также способ изучения воздействия перехваченного трафика на клиентское программное обеспечение. Система построенная по такой схеме может быть использована также для изучения безопасности отдельного приложения.

Honeypot, уязвимости ПО, верификация, спутниковый интернет, fuzzing, process stalking.

В условиях, когда информация выходит на одно из первых мест в жизни пользователям компьютерных систем приходится участвовать в гонках вооружений, которая происходит между создателями злонамеренных систем и специалистами по компьютерной безопасности. Иметь установленный набор специальных средств защиты, таких как брэндмауэр и антивирус, на каждой рабочей станции, подключенной в сеть – это теперь жизненная необходимость. Вирусописатели придумывают новую технологию по распространению своего продукта и антивирусная компания тут же включает соответствующее противодействие в свой, и с этого момента вирусописатель начинает работать над ответным шагом.

В такой борьбе безопасность конечного пользователя компьютерных систем зависит от скорости реакции специалистов по компьютерной безопасности на новую угрозу. Чтобы свести к минимуму время, когда угроза уже есть, а средства противоборства еще нет, можно выполнять следующие мероприятия:

- проводить постоянный анализ безопасности программных продуктов, своевременно исправлять найденные ошибки, разрабатывать технологии, которые усложнят в будущем злонамеренное использование продукта;
- получать обратную связь от пользователей продукта и своевременно реагировать на сообщения о найденных уязвимостях ПО;
- производить мониторинг злонамеренной активности в сети, быть в курсе самых новых технологий, вводить изменения в программное обеспечение до получения сообщений об уязвимостях от пользователей.

Для того чтобы было возможно идти последним из предложенных выше путей существуют системы honeypot. *Honeypot* (от англ. «горшок меда») – это ловушка для злоумышленника. Как правило это сервер, доступный через сеть, единственная цель которого привлечь к себе внимание, зафиксировать факт попытки взлома и получить как можно больше информации о технологиях, которые применялись во время атаки. На рис. 1 представлено место honeypot в некоторой корпоративной сети.

Даже если рабочие сервера (отмеченные на рисунке как prodaction-сервера) оснащены всеми возможными актуальными средствами защиты, добавление в сеть сервера honeypot позволяет быть в курсе об атаках, противостоять которым не способны даже они.

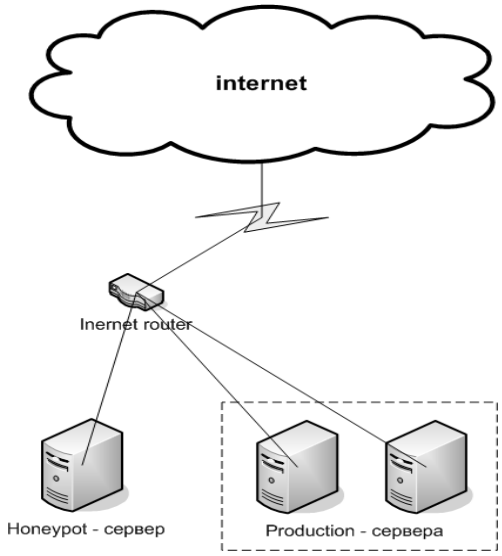


Рис. 1. Пример организации honeypot

Сервера honeypot бывают двух типов[1]: *low interactive* и *hi interactive.* Системы первого типа предоставляют атакующему в распоряжение полную операционную систему. Такие honeypot обычно реализуются на основе виртуальных машин, например, Vmware. Существуют также разработки использующие usermode linux.

Системы второго типа серверами в полном смысле не являются. Для атакующего они создают лишь видимость работы. Реализуются столько сетевых сервисов, сколько нужно. При этом реально процессы этих сервисов могут физически находиться даже не на одном сервере. Honeyd – один из самых популярных серверов honeypot, реализованных по такой схеме.

В названии данной статьи фигурирует термин «пассивный honeypot». Автор не встречал такого определения ранее, но изложенная ниже схема не попадает ни под один из выше перечисленных вид ловушек. По степени взаимодействия он значительно менее активен low interactive honeypot. Взаимодействие с атакующим происходит через получение подготовленного им контента и «запуска» его в тестовом приложении. На рис. 2 показана схема работы так называемого *пассивного honeypot*.

Важным отличием данной схемы является ориентация на клиентское программное обеспечение. В случае, если тестируется серверное ПО, то необходимо заставить злоумышленника соединиться с подставным сервером и выполнить некоторые злонамеренные действия. Клиентские рабочие станции как правило не запускают серверных приложений, не принимают входящие соединения и т.д. В свою очередь они сами соединяются с сервером для того, чтобы получить некоторый контент (html-страница, файл с ftp-сервера и т.д.). В такой ситуации одной из самых эффективных атак является «атака через содержимое» [2].

Пользователь рабочей станции скачивает некий файл и передает его приложению, которое его обрабатывает (проигрывает, отображает и т.д.). Атакующий подготавливает файл так, что попадая в приложение он вызывает непредвиденную ошибку. Таким образом, используя уязвимость ПО, атакующий имеет возможность удаленно запустить произвольный код на рабочей станции.

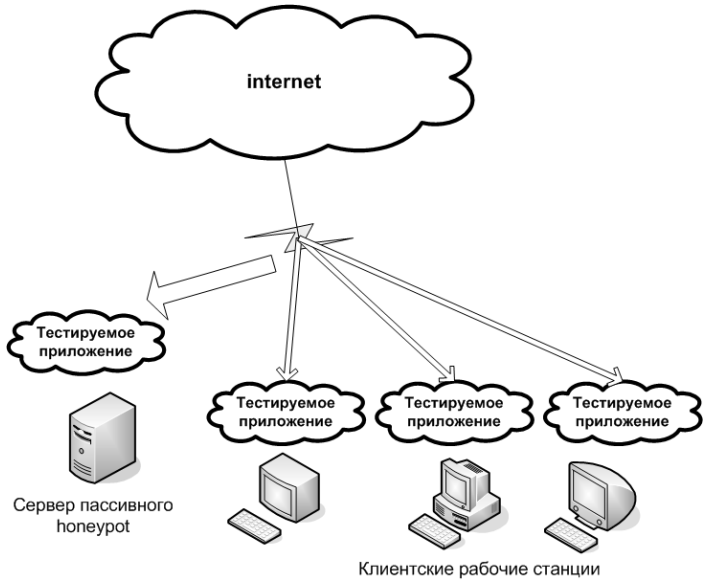


Рис. 2. Схема работы пассивного honeypot

Очевидно, чтобы иметь возможность выявлять атаки подобного вида honeypot должен сам получать файлы, схожие с файлами остальных пользователей (этот факт отмечен на рис. 2 толстой стрелочкой). Еще лучше, если он будет получать те же самые файлы. Это станет возможным, если действовать по одной из предложенных ниже схем:

- поставить на множество рабочих станций специальное ПО – датчики, которые будут собирать образцы;
- собирать образцы из сети интернет с помощью программ-пауков (подобных тем, что используют поисковые машины);
- перехватывать трафик пользователей, анализировать его и выделять интересующие образцы.

Пойти первым путем могут позволить себе разве что крупные антивирусные компании, у которых существует большая сеть клиентов. Но даже в этом случае будет не просто убедить большое количество людей участвовать в программе и высылать файлы данных, которые даже не вызывают подозрения.

Второй способ выглядит более реализуемым, и даже интересным, однако требует для своей реализации затрат большого количества трафика.

Третий способ на первый взгляд нереализуем, ведь при современной организации сетей перехват данных, адресуемых другим пользователям, в общем случае невозможен. Повторители повсеместно заменены маршрутизаторами, но даже если это было бы не так, то анализ трафика локальной сети не так интересен, как файлы адресуемые среднестатистическому пользователю Интернет.

Но даже в современных условиях такую возможность нам предоставляют провайдеры спутникового Интернет. Для организации асинхронного Интернет-канала (исходящий трафик передается по наземному каналу, входящий – через спутник) используется спецификация DVB для построения систем широковещательной передачи данных. Эта же схема используется и для вещания спутникового телевидения. Неограниченное количество пользователей принимают на свои тарелки сигнал со спутника, который несет в себе mpg-2 поток [3].

Интернет-трафик, адресуемый клиентам, также передается в одном потоке. Поток делиться на части (pids). Каждый клиент анализирует поток с нужным ему pid (оговаривается провайдером) и забирает данные, адресуемые карте с соответствующим MAC-адресом.

Так же как и у сетевых ethernet-карт у сетевых DVB-карт есть возможность работать в режиме полного захвата пакетов (promiscuous mode). В этом случае карта будет перехватывать трафик (в зависимости от символьной скорости передачи со скоростью от ~25 Мб/с до 50 Мб/с), адресуемый всем пользователям спутникового Интернет, использующих тот же pid. На этом принципе основывается так называемая *спутниковая «рыбалка»* – распространенный способ бесплатного получения медиаданных у пользователей спутникового Интернет. Файлы, которые скачивают пользователи спутникового интернет (в зависимости от выбора провайдера, можно ограничить аудиторию, например, территориально) могут быть использованы как образцы для пассивного honeypot. При этом пользователи, чьи файлы перехватываются, не против участия в подобной программе (ведь им известно, что файлы передаваемые через спутник не шифруются) и нет расхода трафика Интернет совсем.

Идея исследования влияния данных на исполнение ПО не новая. При тестировании приложений используются такие методы как инжектирование ошибок (fault injection) и fuzzing [4]. Смысл их одинаков: передаем программе данные и изучаем ее реакцию. Различия заключается в том, какие данные передаются. При инжектировании ошибок передаются данные призванные ввести программу в аварийное состояние (и тестировать таким образом ветки обработки аварийных состояний). При использовании fuzzing данные формируются случайно либо в соответствии с форматом входных данных.

Кардинальное отличие пассивного honeypot от вышеописанных методов в том, что приложению передаются «живые» данные из сети. Таким образом, если приложение после того как получило входные данные начало вести себя аномально, то либо просто найдена новая ошибка, либо найден ресурс в сети, который намеренно распространяет специальным образом подготовленный контент, который способен атаковать рабочую станцию. Программное обеспечение, используемое для исследования влияния входных данных на ПО должно выполнять следующий функционал:

1) статически строить деревья исполнения кода;
2) динамически дополнять деревья исполнения кода;
3) инжектировать данные в ПО;
4) выявлять аномальные состояния исполняемого ПО.

Вышеперечисленные действия в некоторых источниках принято называть process stalking [5]. В схеме, описанной в данной статье, использовалась система описанная в [6]. Генератор контента в данном случае заменен на ПО, получающее данные из спутникового потока. Также в схему необходимо добавить блок выявления аномальных состояний, который, основываясь на собранных остальными подсистемами данных, должен принять решение была ли предпринята атака.

В результате объединения вышеописанных методов получаем схему изображенную на рис. 3. Система из [6] обозначена на рисунке как process stalker.

Описанная в статье схема может быть использована при исследовании безопасности ПО, а также для выявления в сети ресурсов, распространяющих злонамеренный контент. Прототип функции был реализован автором. Исследования продолжаются.

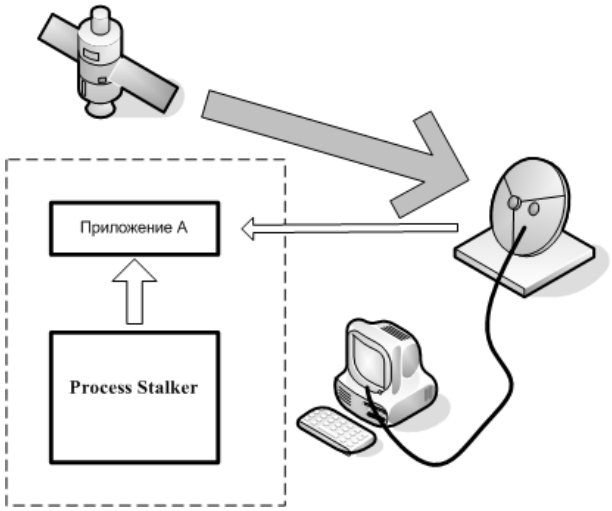


Рис. 3. Схема пассивного honeypot на анализе спутникового трафика

Работа выполнена при поддержке гранта РФФИ №09-07-00245-а.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Niels Provos, Thorsten Holz*. Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison Wesley Professional, 2007. – 480 p.
2. *Greg Hoglund, Gary McGraw.* Exploiting Software: How to Break Code. Addison Wesley. 2004. – 512 p.
3. Digital Video Broadcasting (DVB) Interaction channel for satellite distribution systems. European Standart (Telecommunications series). ETSI EN 301 709 v 1.5.1 (2009-5).
4. *Michael Sutton, Adam Greene, Pedram Amini*. Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley Professional; 1 edition (July 9, 2007).
5. *Pedram Amini*. Process Stalking. Run-time Visual Reverse Engineering. 1. // [Интернет] - режим доступа http://dvlabs.tippingpoint.com/pub/pamini/ProcessStalking_PedramAmini.pdf, свободный
6. *Благодаренко, А.В*. Инструментальное средство для проведения сертификационных испытаний программного обеспечения без исходных кодов. Известия ЮФУ. Технические науки. Тематический выпуск. 2007. – С. 212-215.

Тумоян Е.П.[*], Голубев А.М.[**]
Консультант: д.т.н., профессор Макаревич О.Б.
[*]г. Таганрог, ТТИ ЮФУ; [**]г. Москва, НТЦ "Орион"

Лозунг: Si vis pacem, para bellum

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО МЕТОДА МОДЕЛИРОВАНИЯ СЕТЕВЫХ АТАК С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ METASPLOIT FRAMEWORK

В статье предлагается новый метод формального моделирования сетевых атак на основе похода exploit-dependency. В рамках предложенного метода моделирование атаки реализуется поэтапно с использованием автоматов с функция-

ми переходов и выходов на основе искусственных нейронных сетей. Метод обеспечивает сопряжение с системами тестирования безопасности компьютерных сетей, такими как Metasploit Framework. С использованием данных средств появляется возможность симуляции атак на реальные компьютерные системы, что является существенным достоинством метода.

Моделирование компьютерных атак, вероятностные автоматы, нейронные сети, Metasploit Framewrok.

Одним из важных направлений информационной безопасности является разработка методов и средств, которые позволяют обнаружить факт наличия в программном обеспечении ошибок или недокументированных возможностей, использование которых может привести к нарушению безопасности системы. Однако для обнаружения уязвимостей на этапе эксплуатации системы не существует общепринятых формальных или автоматизированных методов оценки безопасности системы, кроме наиболее простых случаев. В большинстве случаев используются методы тестирования на проникновение (penetration testing), которые предполагают исследование системы квалифицированными специалистами. Такие исследования обычно занимают достаточно больше время и чрезвычайно дороги, поскольку предполагают привлечение значительного количества экспертов. Для упрощения подобных исследований разработаны и активно используются инструментальные средства, которые обеспечивают автоматизированное обнаружение уязвимостей, а также оценку их опасности. Одним из наиболее распространенных средств обнаружения уязвимостей является система Metasploit Framework. По результатам опроса сайта Sectools.org Metasploit Framework занимает пятое место среди ста наиболее популярных средств сетевой безопасности. Однако данные системы имеют некоторые недостатки. Одним из наиболее серьезных недостатков является то, что система не обеспечивает возможностей автоматизации поиска неизвестных уязвимостей и реализацию многостадийных атак. В настоящее время существуют работы, которые позволяют устранить данные недостатки путем моделирования процесса тестирования на проникновения, по сути, моделирования атакующего воздействия на компьютерную систему. Например, в работах [1 - 6] предложены различные модели атак на основе деревьев и графов.

Разрабатываемая модель предназначена для исследования атаки как последовательности этапов, которая приводит целевую систему в состояние, необходимо атакующему. Кроме того, исходя из условий, которые являются типичными для компьютерных атак, модель предполагает следующие ограничения:

1. Атака представляет собой последовательность этапов атаки (так называемый поход exploit-dependency). Этап атаки представляет собой законченное воздействие на атакуемую систему, результаты которого могут быть каким-либо образом оценены атакующим. В отличие от существующих методов на основе exploit-dependency предполагается, что воздействие может быть как эксплуатацией некоторой уязвимости целевой системы, так и нейтральным воздействием.
2. В отличие от существующих моделей, например [2,4], атакуемая система представляется в виде black-box, т.е. атакующий не имеет возможности отслеживать внутреннее состояние атакуемой системы и не имеет ее спецификации, а может воздействовать на систему и получать ответы системы на воздействия через каналы коммуникаций данной системы.

Приведенные ограничения позволяют моделировать сетевые атаки с возможностью сопряжения моделирующей части с средствами тестирования сетевой безопасности.

Для описания предлагаемой модели используем теорию автоматов. Учитывая приведенные выше свойства системы, опишем модель системы как детерминированный конечный автомат, т.е. $M = \{Q, \Sigma, Y, \delta, \lambda, q_0, F\}$, где Q – конечное *множество состояний* автомата, $Q = \{q_i\}$, $\|Q\| = N$, N – количество состояний автомата, Σ – допустимый *входной алфавит,* $\Sigma = \{a_i\}$, $\|\Sigma\| = L$, L – мощность алфавита, Y – допустимый *выходной алфавит,* $Y = \{y_i\}$, $\|Y\| = W$, W – мощность алфавита, δ – *функция переходов автомата,* т.е. отображение множества $Q \times \Sigma$ во множество подмножеств *P(Q)*, при этом $\delta : Q \times \Sigma \to Q$, λ – *функция выходов автомата,* т.е. отображение множества $Q \times \Sigma$ во множество подмножеств *P(Y)*, при этом $\lambda : Q \times \Sigma \to Y$, q_0 – *начальное состояние* автомата $q_0 \in Q$, F – множество *заключительных* состояний, таких что $F \subset Q$, при достижении одного из которых работа автомата прекращается.

Переходы межу состояниями автомата зависят от состояния автомата и входа автомата в каждом состоянии, т.е. переход между состояниями q_i и q_j представляется как

$$q_{i+1} = \delta(q_i, a_i) \; \forall i \,, \tag{1}$$

где a_i – входной символ автомата. Выход автомата вычисляется после перехода автомата в некоторое состояние и остается в нем до следующего перехода:

$$y_i = \lambda(q_i) \; \forall i \,. \tag{2}$$

Необходимо учесть следующий фактор неполноты информации о системе. Пусть множество доступных для контроля входных параметров – a_i', а множество неизвестных параметров – a_i''. Тогда $a'' = a \setminus a'$. Если $a'' = \varnothing$, то мы имеем всю информацию о входных данных некоторого состояния, однако в большинстве реальных случаев это не так. Следовательно, существует подмножество множество $a' \subseteq a$ такое, при котором сложно однозначно определить следующее состояние автомата, т.е. функция переходов $\delta(q_i, a'_i)$ становится недетерминированной, т.е.

$$Q^{i+1} = \bigcup \delta'(Q^i, a'_i) \,, \tag{3}$$

где $Q^i = \{q_k\}$ – множество состояний автомата в момент времени *i*, $Q^{i+1} = \{q_l\}$ – множество состояний автомата в момент времени *i+1*. Необходимо отметить, что данный случай является более общим по отношению к приведенному в (1), поскольку если $a_i' = a_i$, то $q_{i+1} = Q^{i+1}$.

Кроме того, необходимо учитывать, что аналитическое или табличное представление функций перехода в подходе exploit-dependency при отсутствии спецификации невозможно, поскольку взаимосвязи между входами и состояниями автомата неизвестны. Для решения данной задачи предлагается использовать аппроксимацию функции переходов, т.е. на основе конечного набора известных пар

$\{a_j, q_{i+1}\}$ для каждого данного состояния q_i построить функцию $Q^{i+1} = \delta'(a_j)\ j = 1..M, M << L$, где $Q^{i+1} = \{q_k\}$ – поскольку автомат является недетерминированным.

Другой проблемой является сложность аналитического представления функции выходов автомата $\lambda(.)$. Для рассматриваемого случая количество отображений входов автомата в выходы является чрезвычайно большим, а связи между входами и выходами зачастую неявными. Как и в предыдущем случае перейдем к приближению функции переходов для каждого данного состояния.

Предложенная модель позволяет решить проблемы моделирования атаки, однако обладает одним недостатком – модель является чрезвычайно сложной, поскольку каждое состояние характеризуется собственной функцией переходов и выходов, при этом изменение входных данных порождает свой набор состояний.

Для решения данной проблемы предлагается использовать модель общей памяти. Как показано в ряде работ, например [8], состояние автомата можно представить моделью с общей памятью (shared memory) при определенных ограничениях. Модель общей памяти позволяет отражать и накапливать данные о текущем состоянии системы, на основе которых будут рассчитываться функции выходов автомата. В общей памяти содержится подмножество данных, которое влияет только на выход данного состояния $P_i \subseteq a_i$. Такое предположение не нарушает принципов автоматной модели, поскольку не затрагивает функций перехода между состояниями.

Для решения задачи разработки функций переходов введем понятие кластеров состояний. Кластер состояний Q_n – это множество состояний, такое, что $\bigcup_{\forall n} Q_n = Q$. Все элемента кластера состояний должны удовлетворять следующим условиям:

1. Все элементы кластера отличаются только содержимым общей памяти и не отличающихся функциями переходов, т.е. $\forall q_i, q_j \in Q_n : \lambda(P_i) = \lambda(P_j)$.

2. Пусть для всех состояний кластера можно найти такие функции $F(a'_i)$ и $G(a'_i)$, что их суперпозиция будет приближать функции перехода с необходимой точностью, т.е. $\delta'(a_i') \approx F(a'_i) \circ G(a'_i)$. Тогда потребуем, чтобы $\forall q_i, q_j \in Q_n : F(a'_i) = F(a'_j)$.

Таким образом функция переходов автомата представляется в виде суперпозиции функций $F(a'_i)$ и $G(a'_i)$, причем функция $F(a'_i)$ одинакова для всех состояний кластера.

Чтобы гарантировать корректное вычисление общей функции $\delta'(.)$ для расчета используются сети функций радиального базиса (RBFN). Доказано, что [9] данный тип нейронной сети позволят приблизить любую гладкую функцию от входных аргументов. Кроме того, для данного вида нейронной сети результат может быть интерпретирован как вероятность возникновения некоторого выхода, что как будет показано далее, может использоваться при различных операциях с полученной моделью, в том числе при оптимизации.

Учитывая вышеприведенное для расчета $G(a'_i)$ можно использовать простые правила на основе табличных расчетов. Таким образом, функция перехода автомата представляется в виде функции, реализуемой RBFN, которая одинакова

для всех состояний данного кластера и простой табличной функции, которая может быть различна для каждого состояния.

Построение функции выходов на основе различных видов правил может быть приемлемо только для частных случаев, поэтому необходимо использовать аппроксимацию функции выходов.

С учетом приведенных выше соображений в данной работе предлагается вычисление $\lambda(.)$ на основе многослойных нейронных сетей прямого распространения с гладкими активационными функциями (Multilayer Perceptrons, MLP). Доказано, что [10] данный тип нейронной сети позволяет приблизить любую гладкую функцию от входных аргументов. Недостатком данной сети является большое время обучения.

Одним из достоинств данной модели является возможность сопряжения реализации с системой Metasploit Framework.

Модель атакующей системы реализована в среде Mathworks Simulink 6 с использованием библиотеки Stateflow. Необходимые нейронные сети взяты из системы Mathworks Matlab 2006. Функции переходов кластеров состояний (F – часть) реализованы с использованием newrb, функции выходов кластеров состояний реализованы с использованием newff. Атакующая система может взаимодействовать с модельной уязвимой или реальной уязвимой системой.

Для взаимодействия с реальными уязвимыми системами используются скрипты Metasploit Framework 3.2. Такой выбор позволяет не отвлекаться на реализацию конкретного эксплойта или воздействия на систему и сконцентрироваться на моделировании атаки. Скрипты Metasploit выполненные в виде модулей системы Metasploit на языке Ruby. Данные для скриптов Metasploit подготоваливаются в системе Siumulink. Запуск скриптов также осуществляется из системы Simulink. После выполнения скриптов результаты их работы передаются в Simulink и анализируются для определения переходов автомата. В дальнейшем планируется реализация модели на языке Ruby и более тесную интеграцию с Metsaploit Framework 3.

Предложенная в работе модель позволяет выполнять поэтапное моделирование атаки для различных компьютерных систем. Одним из достоинств данной модели является то, что использование общей памяти не нарушает концепции автоматного моделирования, поскольку как уже отмечалось общая память является аргументом только для формирования функции выходов. Приведенные соображения позволяют предположить, что для данного типа автоматов может быть разработана алгебра, позволяющая производить над автоматами наборы действий.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Schneier B.* Attack Trees [Электронный ресурс] /Brus Schneier // Dr. Dobb's Journal, 1999. Режим доступа: http://www.schneier.com/paper-attacktrees-ddj-ft.html.

2. *Camtepe S.A.* A Formal Method for Attack Modeling and Detection [Электронный ресурс] /Seyit Ahmet Camtepe, Bulent Yener // TR-06-01, Rensselaer Polytechnic Institute, Computer Science Department. 2006. Режим доступа: http://citeseer.ist.psu.edu/751069.html.

3. *Sheyner O.* Automated Generation and Analysis of Attack Graphs /Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, Jeannette M. Wing // Proceedings of the IEEE Symposium on Security and Privacy. Oakland, CA, USA, 2002. – P. 273 – 284.

4. *Jha S.* Two Formal Analyses of Attack Graphs /S. Jha , O. Sheyner, J. Wing// Proceedings of the 15th IEEE Computer Security Foundations Workshop. Nova Scotia, Canada, June 2002. – P. 49-63.

5. *Sheyner O.* AttackGraph Tool 0.5 [Электронный ресурс]. Режим доступа: http://www.cs.cmu.edu/~odobzins/scenariograph/as_files/AttackGraph-0.5.tar.gz.

6. *Von Ohiemb D.* Formal security analysis with Interacting state machines /David Von Ohiemb, Volkmar Lotz, Dieter Gollmann, Karjoth Günter, Michael Waidner// Lectute Neotes in Computer Science, 2002, № 2502. ─ P. 212-228.

7. Хопкрофт Д. Введение в теорию автоматов, языков и вычислений / Д. Хопкрофт, Р. Мотивани, Д. Ульман //2-е изд. ─ М.: Издательский дом «Вильямс», 2002. – 528 с.

8 *Rieke R.* Tool based formal Modelling, Analysis and Visualisation of Enterprise Network Vulnerabilities utilising Attack Graph Exploration [Электронный ресурс]/ *Roland Rieke//* In U.E. Gattiker (Ed.), EICAR 2004 Conference CD-rom: Best Paper Proceedings (ISBN:87-987271-6-8) 31 pages. Copenhagen: EICAR e.V.

9. *Poggio T.* A theory of networks for approximation and learning /Poggio, T. Girosi, F.// Technical Report A.I. Memo 1140, Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center for Biological Information Processing, Whitaker College.

10. *M. H. Stone* (1937). Applications of the Theory of Boolean Rings to General Topology. Transactions of the American Mathematical Society 41 (3). ─ 375–481 p.

Хромов А.В.
г. Юбилейный Московской области, ЗАО «ЭКА»

Лозунг: Моделирование и компьютерные игры

ТЕХНОЛОГИЯ ПОДГОТОВКИ И ПРОВЕДЕНИЯ КОМПЬЮТЕРНЫХ ИГР ДЛЯ ОЦЕНКИ УСТОЙЧИВОСТИ ФУНКЦИОНИРОВАНИЯ КРИТИЧЕСКИ ВАЖНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ С ПРИМЕНЕНИЕМ СПЕЦИАЛИЗИРОВАННЫХ ПРОГРАММНО-АППАРАТНЫХ КОМПЛЕКСОВ

В статье представлено описание перспективной технологии подготовки и проведения компьютерных игр для оценки устойчивости функционирования критически важных информационных систем с применением специализированных программно-аппаратных комплексов.

Технология, компьютерная игра, устойчивость функционирования, критически важная информационная система.

Современный этап развития систем управления характеризуется интенсивным внедрением и применением автоматизированных систем (АС), комплексов средств автоматизации, средств информационного и программного обеспечения, которые используются, в настоящее время, практически во всех сферах деятельности.

Широкое использование автоматизированных систем в деятельности органов управления (ОУ) приводит к тому, что эффективность их функционирования всё больше и больше зависит от качества и устойчивости функционирования применяемых АС. То есть выход из строя применяемых АС может с высокой вероятностью привести к значительному снижению эффективности и в некоторых случаях невозможности выполнения задач ОУ, что в свою очередь может привести к крайне нежелательным последствиям.

Одним из перспективных направлений решения данной группы проблемных вопросов является создание средств моделирования поведения критически важных информационных систем (КВИС) в различных условиях обстановки с целью оценки устойчивости функционирования, выработки обоснованных рекомендаций, направленных на повышение устойчивости функционирования КВИС, на основе

применения компьютерных игр (КИ) и программно-аппаратных средств их реализации.

Разработанная для реализации данной идеи новая информационная технология включает две группы процессов:

1. Технологические процессы *подготовки* компьютерных игр.

2. Технологические процессы *проведения* компьютерных игр.

Описание технологических процессов подготовки и проведения компьютерных игр, выполнено с применением современного инструментального программного средства моделирования технологических процессов BizAgi Process Modeler BETA в перспективной нотации BPMN. При этом роли участников КИ формализованы следующим образом:

- руководитель КИ;
- атакующая сторона;
- защищающаяся сторона.

В область ответственности Руководителя КИ входит определение условий проведения (общая характеристика моделируемой обстановки, уровень сложности КИ, диспозиция, состав доступных для использования средств и ограничения на их применение, а также на время проведения КИ) и целевой установки для операторов КИ (что должен достичь каждый из участников в процессе КИ), а также контроль, анализ и оценка действий, выполненных операторами КИ.

Операторы КИ (Атакующая и Защищающаяся стороны КИ) в соответствии с установленными условиями КИ в течение отведённого времени выполняют действия, направленные на достижение целевой установки.

Разработка новой информационной технологии подготовки и проведения компьютерных игр проведена с использованием методов, методик и математического аппарата теории игр, теории исследования операций, теории нечётких множеств, теории матриц и построения графов.

Описание технологических процессов подготовки компьютерных игр с применением специализированных аппаратно-программных комплексов.

Формализованное описание (общая схема) технологического процесса подготовки компьютерных игр с применением специализированных аппаратно-программных комплексов (АПК) представлено на рис. 1.

Шаг 1. Компьютерная игра начинается с определения Руководителем КИ целевой установки и ключевых условий (исходных данных) её проведения для операторов КИ.

В рамках выполнения данного шага должно быть установлено:

а) общие условия обстановки и диспозиция (легенда игры), а также цель КИ для каждой из противоборствующих сторон, т.е., что должно быть достигнуто операторами КИ в процессе её проведения.

Например, для атакующей стороны типовой целевой установкой может быть вывод из строя КВИС защищающейся стороны или хищение (подмена) электронных документов или сведений, циркулирующих в автоматизированном контуре управления моделируемого объекта защиты, а для защищающейся стороны – сохранение работоспособности и требуемого уровня производительности КВИС, достаточных для реализации моделируемого технологического цикла управления (ТЦУ);

б) уровень сложности КИ, который может принимать значения «высокий», «низкий» или средний. Данный параметр игры является довольно значимым, так как от установленного уровня сложности зависит состав средств (средств защиты и воздействия), доступных для применения операторами КИ, а также устанавливаются ограничения на возможности их применения в процессе КИ.

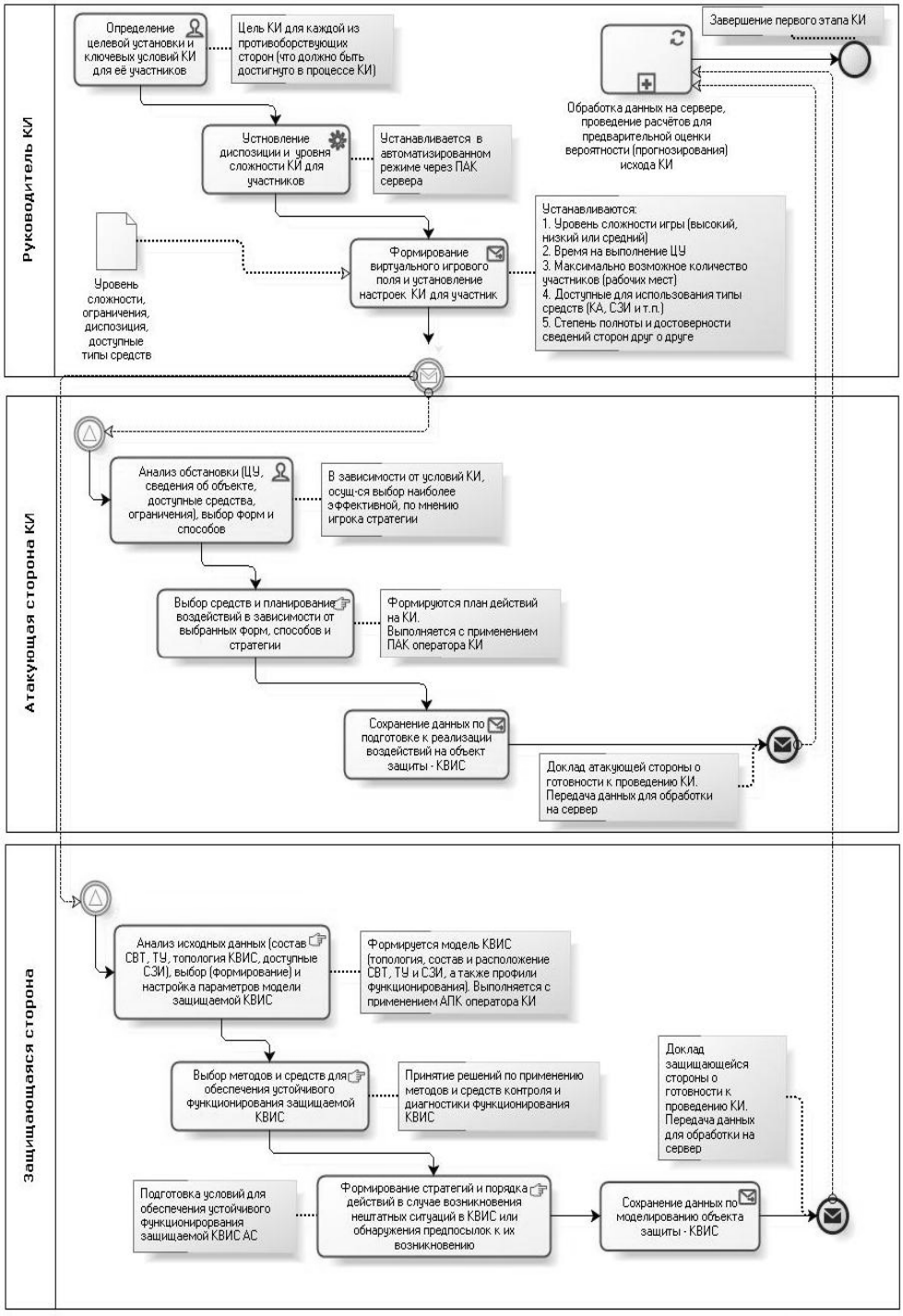


Рис. 1. Схема технологического процесса подготовки компьютерных игр с применением специализированных АПК

Например, может быть ограничено количество применений атакующей стороной компьютерной атаки (КА) определённого вида (не более 2-3 раз за игру) или количество межсетевых экранов или средств защиты информации от несанкционированного доступа, которые могут быть использованы защищающейся стороной при формировании модели КВИС. Ещё одним фактором, который зависит от установленного уровня сложности игры, является объём достоверной информации, т.е. вся информация о противоборствующей стороне известна только в том случае, когда уровень сложности для игрока установлен и имеет значение «низкий». Если уровень сложности для игрока установлен «высокий», то ему в начале КИ предоставляются минимальные сведения о противоборствующей стороне.

Следует отметить, что в случае установления уровня сложности КИ «высокий» для одного из игроков, для второго уровень сложности автоматически устанавливается в значение «низкий» и наоборот. Другими словами, если, например, для атакующей стороны установлен «низкий» уровень сложности, то оператору КИ доступно максимально возможное количество средств для моделирования реализации КА и становится известна практически вся информация о модели объекта защиты и применяемых в ней средствах. В то же время защищающаяся сторона не обладает практически никакими сведениями об атакующей стороне;

в) дополнительными ключевыми параметрами, которые могут быть установлены Руководителем КИ, являются:

– время на проведения КИ;

– доступные для использования типы (виды) моделируемых средств;

– степень полноты и достоверности сведений сторон друг о друге и т.п.

Шаг 2. В соответствии с установленными по результатам выполнения шага 1 условиями в автоматизированном режиме через АПК сервера КИ производится в интерактивном режиме ввод или загрузка из заранее подготовленного файла коммуникативного формата исходных данных, по завершению которых:

– формируется виртуальное игровое поле компьютерной игры;

– формируются данные для установки настроек на АПК операторов КИ;

– выдаётся сигнал оповещения операторам КИ «приступить к подготовке КИ», после чего открывается доступ на АПК операторов КИ к использованию настроенных в соответствии с установленными условиями игры и ограничениями информационно-программных средств для: моделирования КВИС Защищающейся стороной и планирования воздействий на объект защиты Атакующей стороной.

Шаг 3. Операторы КИ выполняют с применением специализированных АПК все необходимые действия по подготовке к проведению КИ, в рамках которых должно быть реализованы следующие мероприятия

а) оператор КИ Атакующей стороны:

– Проводит анализ обстановки (ЦУ, сведения об объекте защиты – КВИС, доступные средства и ограничения на их применение и т.п.) и, в зависимости от заданных Руководителем условий КИ, осуществляет выбор наиболее эффективной, по его мнению, формы (форм) и/или способа (способов) ведения КИ.

– В соответствии с выбранными формами и способами производится выбор средств моделирования реализации КА на КВИС и планирование действий для достижения целевой установки. По результатам выполнения данной операции с помощью АПК оператора КИ определяется первоначальная стратегия действий.

– Сохраняет введённые через интерфейс АПК оператора КИ данные по подготовке к реализации КА на объект защиты – КВИС и докладывает Руководителю КИ о завершении подготовки и готовности к проведению КИ. Данные сохраняются автоматически в базе данных сервера КИ.

б) оператор КИ Защищающейся стороны:

– Проводит анализ исходных данных (ЦУ, сведения об Атакующей стороне, состав СВТ, технических устройств и средств телекоммуникации и сетеобразования, возможные варианты топологии построения вычислительной сети КВИС, средства и ограничения на их применение и т.п.) и формирует модель объекта защиты. Формирование модели объекта защиты производится с применением информационно-программных средств АПК оператора КИ посредством выбора элементов технической инфраструктуры КВИС из конечных списков (построенных на основе классификаторов и справочников НСИ базы данных сервера), задания физических и/или логических связей между ними, а также установления значений для основных параметров и режимов функционирования.

– Принимает решения по применению достаточных, по его мнению, методов и средств контроля и диагностики функционирования для обеспечения устойчивого функционирования защищаемой КВИС, которые доступны для использования по условиям КИ, посредством выбора и настройки через АПК оператора КИ соответствующих элементов моделирования.

– Формирует перечень стратегий и порядка действий в случае возникновения нештатных ситуаций в КВИС или обнаружения предпосылок к их возникновению, наиболее, по его мнению, вероятных, исходя из условий проведения КИ, известных или прогнозируемых сведений об Атакующей стороне и особенностями сформированной модели КВИС.

– Сохраняет введённые через интерфейс АПК оператора КИ данные по результатам моделирования объекта защиты – КВИС (модели инфраструктуры КВИС, ТЦУ и функционирования) и докладывает Руководителю КИ о завершении подготовки и готовности к проведению КИ. Данные сохраняются автоматически в базе данных сервера КИ.

Шаг 4. По окончанию выполнения операторами КИ всех подготовительных действий для проведения игры (выполнению мероприятий в соответствии с установленными на шаге 3) сохранённые в базе данных сведения обрабатываются средствами сервера КИ и проводятся расчёты для предварительной оценки вероятности (прогнозирования) исхода КИ.

На этом технологический процесс подготовки компьютерных игр с применением специализированных АПК завершается.

Описание технологических процессов проведения компьютерных игр с применением специализированных аппаратно-программных комплексов.

Формализованное описание (общая схема) технологического процесса проведения компьютерных игр с применением специализированных аппаратно-программных комплексов (АПК) представлено на рис. 2.

Шаг 5. По результатам обработки данных о выполнении подготовительных мероприятий Операторами КИ на сервере производится расчёт вероятности исхода КИ, на основании которого Руководитель КИ оценивает степень готовности к её проведению и принимает решение либо о начале КИ, либо о нецелесообразности её проведения. Решение о нецелесообразности проведения КИ может быть принято руководителем в том случае, если прогнозируемая вероятность выигрыша одной из сторон (исход игры) составляет не менее 0,9, после чего осуществляется переход к выполнению шага 8.

Шаг 6. Сущность выполнения данного шага Операторами КИ состоит в выполнении следующей типовой последовательности действий.

а) оператор КИ Атакующей стороны:

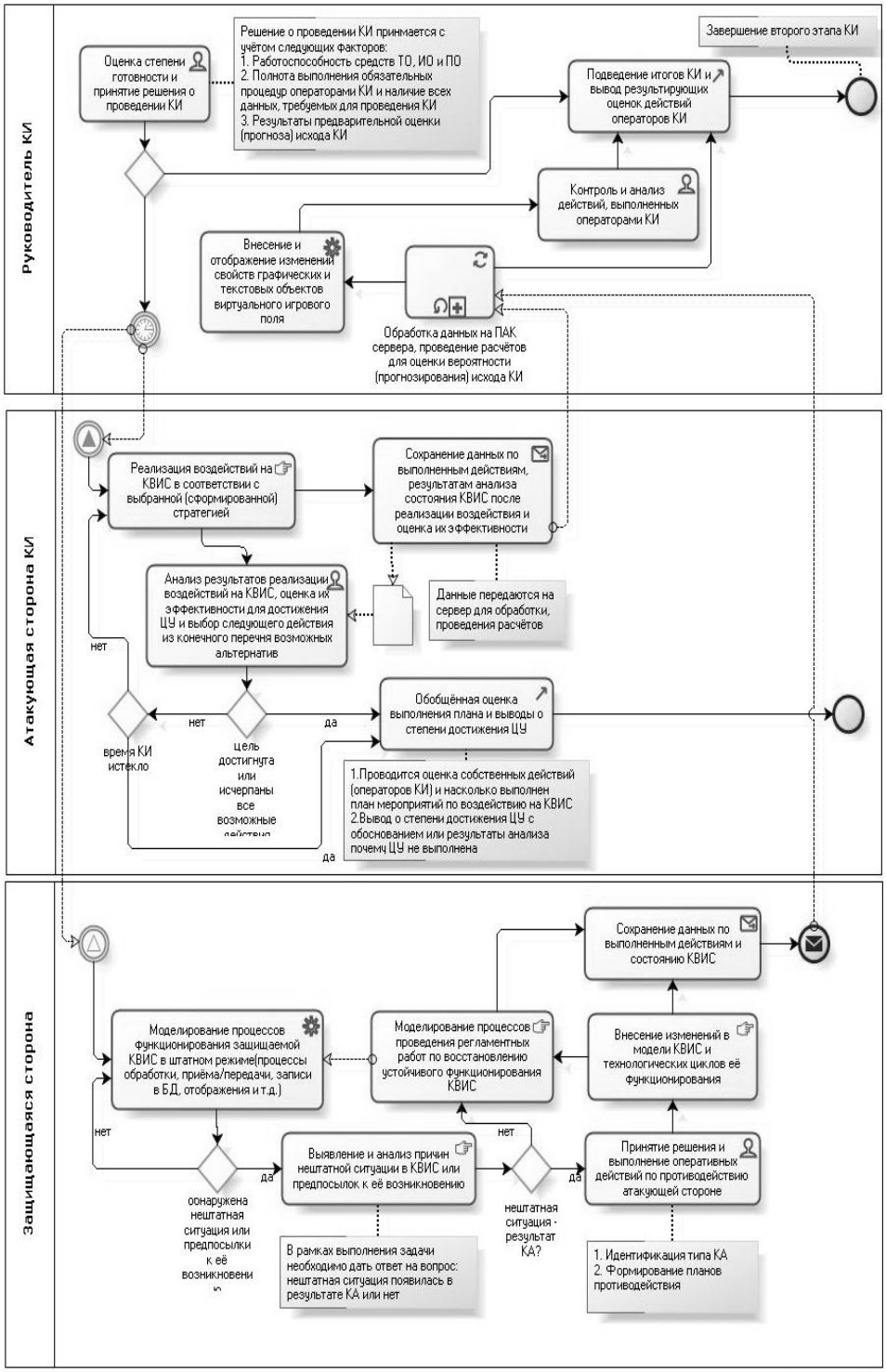


Рис. 2. Схема технологического процесса проведения КИ с применением специализированных АПК

– Производит инициализацию (моделирование) воздействий на КВИС с помощью АПК Оператора КИ в соответствии с выбранной (сформированной) стратегией и разработанным планом действий.

– Результаты выполненных Оператором КИ действий по моделированию КА на КВИС, а также экспертные оценки эффективности их проведения с точки зрения степени достижения поставленной цели сохраняются и передаются на обработку на сервер КИ.

– Если целевая установка, определённая для Атакующей стороны Руководителем КИ, по мнению Оператора КИ, достигнута или истекло установленное на КИ время, проводится обобщённая оценка и выводы о степени достижения ЦУ с обоснованием или результатами анализа причин, по которым ЦУ не была достигнута.

б) оператор КИ Защищающейся стороны:

– Производит инициализацию (моделирование) технологических циклов управления и/или процессов штатного функционирования КВИС (например, по заблаговременно сформированному для КВИС профилю нормального поведения), а также мониторинг (контроль) их выполнения с помощью АПК Оператора КИ в соответствии с параметрами сформированной модели.

– В случае обнаружения нештатной ситуации или предпосылок к её возникновению (например, обнаружены отклонения от профиля нормального поведения КВИС или заданный ТЦУ выполняется не в той последовательности, в которой предполагалось) Оператором КИ по различным признакам проводится анализ возникшей ситуации и выявляются причины, по которым должен быть с той или иной степенью уверенности должен быть сделан однозначный вывод о том, что нештатная ситуация возникла или в результате компьютерной атаки или без внешнего вмешательства.

– Если делается вывод о выявленной нештатной ситуации или предпосылок к её возникновению в защищаемой КВИС как результата внутренних проблем, то производится моделирование процессов проведения регламентных работ по восстановлению нормального (штатного) функционирования КВИС. Сущность моделирования состоит в выборе Оператором КИ той или иной последовательности действий (выбираются из списков через интерфейс АПК оператора) до тех пор, пока моделируемая КВИС не возвращается в устойчивое состояние.

– Если делается вывод о выявленной нештатной ситуации или предпосылок к её возникновению в защищаемой КВИС как результата компьютерной атаки, то принимается решение и должны быть выполнены действия по противодействию Атакующей стороне КИ, в рамках которых должны быть произведены, в первую очередь, идентификация КА и формирование планов его (её) нейтрализации. После этого Оператор защищающейся стороны получает возможности по внесению изменений в модель КВИС через АПК Оператора КИ.

– Информация по всем действиям, выполненным Оператором КИ по устранению нештатных ситуаций или предпосылок к их возникновению в защищаемой КВИС, сохраняются в БД и передаются для обработки на сервер.

Шаг 7. В процессе проведения игры Руководителем КИ через сервер осуществляется контроль и анализ действий Операторов КИ противоборствующих сторон. Сущность контроля со стороны Руководителя КИ заключается в анализе действий Операторов КИ посредством:

– визуального контроля изменений обстановки, отображаемых на средства отображения коллективного пользования или средства отображения сервера КИ после выполнения действий Операторами КИ;

– прогнозирования исхода КИ по результатам расчётов вероятностей исхода КИ, выполняемых средствами сервера в автоматическом режиме после поступления данных о выполненных действиях Операторами КИ (после выполнения каждого хода Атакующей и Защищающейся сторонами и

сохранения результатов выполненных действий в БД).

Шаг 8. Руководителем КИ выдаётся команда на завершение игры, проводится подведение итогов, формируются окончательные выводы и результирующие оценки действий Операторов КИ.

Таким образом, в данной статье представлено описание разработанной перспективной технологии подготовки и проведения компьютерных игр для оценки устойчивости функционирования КВИС с применением специализированных АПК.