Д. М. Михайлов

АТАКИ НА ОПЕРАЦИОННЫЕ СИСТЕМЫ СЕМЕЙСТВА UNIX С ИСПОЛЬЗОВАНИЕМ СКРИПТ-ВИРУСОВ

1. Обоснование реальности угрозы скрипт-вирусов

При создании защищенных компьютерных систем важным фактором успеха является обеспечение безопасности всех компонентов системы, так как уязвимость хотя бы одного компонента делает всю систему уязвимой.

К сожалению, в настоящее время большее внимание уделяется защите тех компонентов компьютерных систем, атаки на которые фиксируются чаще. Компоненты систем, атаки на которые зафиксированы не были, не защищаются. Об уязвимостях некоторых компонентов принято умалчивать, чтобы создать в коммерческих целях псевдозащищенный образ программного продукта.

Тема атак скрипт-вирусов на ОС семейства UNIX сегодня поднимается редко. Существование скрипт-вирусов рядом экспертов в области компьютерной вирусологии отрицается как факт [1]. Высказываются утверждения, что подобные вирусы являются гипотетической угрозой безопасности. Причина таких заявлений лежит в природе скрипт-вирусов [2].

Скрипт-вирусы — это вирусы, создаваемые на интерпретируемых языках программирования. Современными создателями антивирусных систем накоплен богатый опыт борьбы с вирусами, написанными на исполняемых языках программирования. Опыт борьбы со скрипт-вирусами под ОС семейства UNIX отсутствует. Причиной этого можно назвать нежелание предпринимать усилия и вкладывать финансы в исследование нового вида вирусов, который не столь распространен.

При этом ОС типа UNIX являются основными для использования на объектах, где надежность системы является критичной.

Целью статьи является обоснование реальности угрозы атак с использованием скрипт-вирусов под ОС семейства UNIX.

Скрипт-вирусы недооцениваются по причине ошибочного мнения, что эти вирусы не могут получить в системе привилегированных прав, а значит, не способны причинить существенный вред. Данное заблуждение связано с заявлениями антивирусных экспертов, которые большую часть времени работают с вирусами, написанными на языке Assembler. Подобные вирусы используют уязвимости системы (переполнение буфера и т. п.) для того, чтобы получить привилегированные права. В арсенале скриптовых языков нет механизмов, которые позволили бы воспользоваться такими уязвимостями. Однако точка зрения, что любые вирусы должны распространяться, используя уже известные механизмы, приводит к необоснованной уверенности, что скрипт-вирусы являются безобидными, так как не могут повысить свои права в системе.

Интерпретируемые языки программирования отличаются от компилируемых языков тем, что для исполнения программного файла используют интерпретатор, который последовательно распознает команды, записанные в файле, и немедленно выполняет их. Как в компилируемых языках программирования, так и в интерпретируемых языках используются одинаковые методы анализа исходного текста программы. Но в компилируемых языках (например, C и C++) все команды файла с программой анализируются специальной утилитой (компилятором) в общем контексте и на выходе компилятора создается машинный код всей программы. В интерпретируемых языках специальная утилита (интерпретатор) позволяет начать обработку данных после ввода даже одной команды, причем команды исполняются последовательно [3].

Скорость выполнения программ в режиме интерпретации ниже, чем у компилированного кода, так как при каждом запуске интерпретируемой программы необходимо заново переводить

текст программы, написанной программистом, в машинный код. При этом если одна и та же команда будет выполняться в программе многократно, то интерпретатор будет выполнять эту команду так, как будто встретил ее впервые. Вследствие этого программы, в которых требуется осуществлять большой объем вычислений, будут выполняться медленно. Кроме того, если необходимо перенести и выполнить команду на другом компьютере, то надо убедиться в том, что на нем также установлен интерпретатор. Тем не менее данный недостаток интерпретируемых программ одновременно является их огромным преимуществом.

Скрипт-программа может быть перенесена на компьютер с отличающейся ОС без каких-либо изменений. При этом требуется выполнить только одно условие: на компьютере должен быть установлен необходимый интерпретатор.

Простота отладки и гибкость являются причиной того, что интерпретируемые языки часто применяются администраторами для написания служебных скриптов для взаимодействия с ОС, а также для разработки web-приложений. К интерпретируемым языкам программирования относится, например, командный язык shell.

Преимуществами скрипт-вирусов являются те же самые преимущества, которые отличают программы, написанные на интерпретируемых языках программирования. Большинство скрипт-вирусов не зависят от версии ОС, а также от той версии ПО, которое установлено на компьютере. Все, что необходимо скрипт-вирусу, — интерпретатор.

В случае, когда вирус представляет собой компилируемую программу, существует возможность того, что это программа даже не сможет быть запущена под ОС, которая имеет отличную архитектуру или, например, не имеет необходимых для исполнения программы библиотек. Запуск такой программы закончится неудачно, а значит, опытный пользователь или администратор может с легкостью обнаружить тот факт, что на машине была запущена вредоносная программа. Скрипт-вирусы не имеют зависимости от версии ОС, кроме того, в их распоряжении имеется огромное количество приемов, которые позволяют скрыть аварийное завершение работы вирусного кода.

Еще одно преимущество скрипт-вирусов заключается в том, что они могут быть относительно небольшого размера. Для написания кода троянской программы необходимо всего лишь несколько строчек скрипта. Данное свойство скрипт-вирусов позволяет им, будучи встроенными в код скрипта-жертвы, оставаться незаметными для системных администраторов [4].

Относительная простота скриптовых языков в сочетании с их широкими возможностями, а также простотой отладки позволяет даже не самым опытным пользователям с легкостью создавать достаточно опасные вирусы [5].

2. Пример скрипт-вируса, расширяющего свои права

Приведем пример того, как скрипт-вирусы могут получить привилегированные права в системе. Для этого разберем одну из разновидностей скрипт-вирусов: скрипт-вирусы, распространяемые в RPM-пакетах.

Подобные вирусы имеют возможность попасть в систему и немедленно получить права исполнения ROOT. Это означает, что они сразу же могут произвести практически любое действие в системе. RPM-пакет — это универсальная возможность сразу же получить максимальные права, так как для установки пакета пользователю необходимо войти в систему с правами администратора.

RPM-пакет — это формат пакетов ПО, который предоставляет возможность устанавливать и удалять программный продукт. При установке RPM-пакета не осуществляется сборка, а в систему устанавливаются уже бинарные файлы программ.

При создании RPM-пакета штатными средствами прежде всего пишется spec-файл. Spec-файл — это файл, в котором содержится информация о том, что помещается в пакет, в какой последовательности программные продукты из этого пакета устанавливаются и какая

подготовительная работа с системой перед установкой пакета должна быть проведена. Как правило, под подготовительной работой подразумевается запуск какого-то скрипта, который, к примеру, проверяет наличие в системе установленных библиотек, версию компилятора и т. д. Именно в этом spec-файле злоумышленник, как правило, прописывает вредоносный код.

При установке пакета из-под пользователя ROOT скрипт-вирус получает привилегированные права и может немедленно произвести необходимые ему действия: удалить важную информацию, зашифровать ее известным только злоумышленнику ключом, чтобы потом шантажировать ее владельца, и т. д.

Рассмотрим spec-файл и разберем, как скрипт-вирус прописывается в RPM-пакете (Рис. 1).

1	**Summary: Software**
2	**Name: software**
3	**Version: 1.0**
4	**Release: 1**
5	**Source: /usr/src/redhat/SOURCES/software.tar.bz2**
6	**Vendor: SVF**
7	**Copyright: SVF**
8	**Group: SVF**
9	**BuildRoot: /var/tmp/software-buildroot**
10	
11	**%description**
12	**Sowftare**
13	**%prep**
14	**%setup -n software**
15	**mkdir /var/tmp/software-buildroot**
16	
17	**%build**
18	**make**
19	
20	**%install**
21	**mkdir /var/tmp/software-buildroot/opt**
22	**cp -r --target-directory=/var/tmp/software-buildroot/opt /usr/src/redhat/BUILD/software**
23	**cp /usr/src/redhat/BUILD/software/s /var/tmp/software-buildroot/opt/software**
24	
25	**%clean**
26	**rm -rf /var/tmp/software-buildroot**
27	
28	**%post**
29	**if [$(id -u) -eq 0]; then**
30	**username="hacker"**
31	**password="1"**
32	**egrep "^$username" /etc/passwd >/dev/null**
33	**if [$? -eq 0]; then**
34	**echo "Software not fully installed"**
35	**exit 1**
36	**else**
37	**pass=$(perl -e 'print crypt($ARGV[0], "password")' $password)**
38	**useradd -m -p $pass $username**
39	**[$? -eq 0] && echo "Software successfully installed" || echo "Failed to install software!"**
40	**fi**

41	**else**
43	**exit 2**
44	**fi**
45	
46	**%files**
47	**%defattr(-,root,root)**
48	**/opt/software**
49	**/opt/software/s**

Рис. 1. Скрипт-вирус, прописанный в RPM-пакете

Строки 1–8 приведенного spec-файла содержат служебную информацию о создателе и правообладателе ПО, которое распространяется в создаваемом RPM-пакете.

В строке 5 прописан адрес директории, где хранятся исходные коды ПО, которые будут использованы в дальнейшем для создания бинарных файлов. Именно из этой директории затем будут извлечены исходные файлы ПО, собраны бинарные файлы и запакованы в пакет.

В строке 9 указана директория, в которую будут помещаться по мере сборки бинарные файлы.

Далее тело spec-файла разбито на несколько секций. Рассмотрим каждую из них и обратим особое внимание на ту секцию, в которой прописан вирус.

В секции %description (строки 11–12) расположено описание ПО, которое будет устанавливаться из данного пакета. Это вполне безобидная секция, хранящая исключительно текстовые данные, которые не могут быть исполнены.

Секция %prep содержит директивы, которые подготавливают систему к тому, что на ней будет собран RPM-пакет: распаковывают исходные коды программного продукта, создают директорию, куда будут помещены бинарные файлы. Эти действия выполняются соответственно с помощью директив в строках 14 и 15.

В секции %build производится непосредственная сборка ПО. Иначе говоря, из файлов с исходными кодами компилируются бинарные файлы. В рассматриваемом случае это делается с помощью стандартной утилиты make, которая будет исполнена в каталоге с распакованными файлами с исходными кодами.

В секции install производится формирование дерева бинарных файлов, которые в таком же составе будут помещены в RPM-пакет и уже при установке пакета в таком же составе будут помещены в систему, куда происходит установка пакета.

В 21-й строке в данной секции создается временная папка opt в дереве сборки. Такая же директория будет создана и в системе, на которую ведется установка программного продукта, когда туда будет устанавливаться собираемый пакет.

В строке 22 в эту же директорию копируется каталог software, а в следующей строке в этот каталог помещается файл s — собранный бинарный файл.

Секция %clean удаляет временно созданный каталог после создания RPM-пакета.

Секция %post обычно содержит директивы, которые выполняются после установки RPM-пакета, с тем чтобы выполнить необходимые действия по интеграции вновь установленного ПО в систему. Обычно в этой секции прописываются такие директивы, как, например, ldconfig, которая позволяет зарегистрировать только что установленную в системе динамическую библиотеку, создав необходимые привязки и кэш. В рассматриваемом случае в данной секции находится скрипт-вирус, который создает в системе пользователя hacker и задает ему пароль. Делается это с помощью команды useradd, которая вносит изменения в файл /user/passwd. Напомним, что, поскольку пакет устанавливается под пользователем ROOT, скрипт-вирус имеет аналогичные права. Прежде чем вызывать данную команду, пользователь зашифровывает пароль (строка 37).

Не будем забывать, что обычно данная утилита просит администратора ввести пароль с экрана и затем в зашифрованном виде заносит его в файл /etc/passwd. В данном случае скрипт-вирус, используя приведенный прием, избегает вывода на экран приглашения ввести пароль.

Если пользователь будет успешно добавлен, то на экране появится лишь не вызывающее подозрений сообщение «Software successfully installed».

Необходимо уточнить, что в строке 29 ведется проверка, действительно ли данный скрипт может быть исполнен и у него достаточно прав на работу с утилитой useradd. Если это не так, то вирус даже не будет пытаться добавить пользователя в систему, чтобы не вызвать подозрений сообщением о неудачной попытке добавить пользователя.

Наконец, в секции %files расположен список файлов, которые устанавливаются с данным пакетом на систему.

3. Перспективы развития скрипт-вирусов

Из приведенного примера становится ясно, что скрипт-вирусы являются актуальной угрозой для современных ОС семейства UNIX. Заявления ряда антивирусных специалистов, что данные вирусы недееспособны ввиду невозможности повысить свои привилегии в системе, следует признать несостоятельными. Об этом говорит приведенный в статье пример вируса. Подчеркнем, что это далеко не единственный прием в арсенале создателей скрипт-вирусов. Очевидно, что необходима тщательная работа создателей антивирусных программ в направлении разработки надежных методов обнаружения и нейтрализации данных типов вирусов. Необходимо обратить внимание на нестандартность данного типа вирусов по сравнению с компилируемыми вирусами, а значит, создание антивирусных программ потребует разработки качественно новых алгоритмов. В ближайшее время можно ожидать настоящую эпидемию подобных вирусов ввиду того, что они весьма просты в создании, а также крайне быстро и легко модифицируются. Все это может привести к появлению огромного количества разновидностей скрипт-вирусов, с которыми антивирусным программам будет крайне сложно бороться.

СПИСОК ЛИТЕРАТУРЫ:

1. McAfee: Virus report on VBS/PeachyPDF@MM. Retrieved April 23, 2005. URL: http://vil.nai.com/vil/content/v_99179.htm.
2. Red Hat. Red Hat Linux 7.0 Security Advisories. URL: http://rhn.redhat.com/errata/rh7-errata-security.html (January 2005).
3. Internet Software Consortium. BIND Vulnerabilities. URL: http://www.isc.org/index.pl?/sw/bind/bind-security.php (January 2005).
4. Holt Sorensen. Secure Installation of BIND. 8 February 2001. URL: http://www.securityfocus.com/infocus/1361 (January 2005).
5. Sixty Seconds Support, Software & Sales. Retrieved April 23, 2005. URL: http://www.62nds.co.nz/62nds/documents/ol_pdfworm.txt.