Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем вычислительных комплексов (АСВК)

Козлов Никита Вячеславович

Извлечение данных о состоянии веб-приложения на основе анализа образа памяти веб-сервера

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
м.н.с. А.А. Петухов

Москва, 2017

Аннотация

Данная работа посвящена исследованию возможности извлечения конфиденциальных данных из образа памяти веб-сервера работающего под управление операционной системы из семества unix-подобных. Для исследований и экспериментов использовался веб-сервер Apache, а в качестве системы управления содержимым сайта использовался Wordpress. На полученной системе были протестированы различные методы извлечения приватного ключа и других конфиденциальных данных.

Содержание

1 Введение

1.1 Предметная область

Веб-приложения играют значительную роль в жизни современного общества. С их помощью люди могут оформлять документы, заказывать любые товары, покупать и продавать ценные бумаги и многое другое. Для работы любого такого веб-приложения необходим веб-сервер, который будет обслуживать всех клиентов.

Веб-сервер [31] — сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно станцию, на которой это программное обеспечение работает.

Клиент, которым обычно является веб-браузер, передаёт веб-серверу запросы на получение ресурсов, адресуемых идентификаторами в формате URL. Ресурсы — это HTML-страницы, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ веб-сервер передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP.

1.2 Дамп памяти

Дамп памяти [30] - это содержимое рабочей памяти процесса, его формирует операционная система в случае, когда процесс завершается из-за критической ошибки [6] [10]. Сигналы, которые посылает операционная система из семейства UNIX, приводящие к экстренному завершению работы процесса и созданию дампа памяти:

- SIGFPE - критичная ошибка в арифметической операции.
  - Деление на ноль.
- SIGILL - недопустимая инструкция.
- SIGSEGV - ошибка доступа к памяти.
  - Переполнение буфера [5].
  - Переполнение стека [27].
  - Переполнение кучи [11].
  - Обращение на чтение или запись к области памяти для которой запрещены чтение или запись [16].
- SIGBUS - разыменовывание недопустимого указателя.
- SIGABRT - вызов функции abort.

– Освобождение ранее не выделенной памяти.

- SIGTRAP - достигнута точка останова.
- SIGEMT - ошибка эмулятора.
- SIGSYS - невалидный системный вызов.
- SIGQUIT - критичная ситуация замеченная программой [26].

Максимальный размер дампа памяти зависит от настроек операционной системы - значения RLIMIT_CORE. Размеры дампов памяти, полученных в тестовых условиях, находятся в пределах нескольких сотен мегабайт, размер дампа памяти, полученного на реальном сервере составляет 30Мб.

Дамп памяти может включать в себя значения регистров процессора, содержимое стека, глобальные переменные, переменные окружения, локальные переменные и аргументы функций во всех фреймах на момент создания дампа памяти и т.д. С его помощью разработчик может посмотреть состояние программы на момент ошибки. Пример анализа дампа памяти на следующем листинге 1.

Листинг 1: Анализ дампа памяти в gdb

```
> gdb -q /usr/sbin/apache2 core.apache2.23383
Reading symbols from /usr/sbin/apache2...Reading symbols
    from /usr/lib/debug//usr/sbin/apache2...done.
done.
[New LWP 23383]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/
    libthread_db.so.1".
Core was generated by `/usr/sbin/apache2 -k start'.
Program terminated with signal SIGSEGV, Segmentation
    fault.
#0  0x00007fbbfa5b957f in do_fcntl (arg=0x7fbbfa9f88e0,
    cmd=7, fd=14)
    at ../sysdeps/unix/sysv/linux/fcntl.c:39
(gdb) bt
#0  0x00007fbbfa5b957f in do_fcntl (arg=0x7fbbfa9f88e0,
    cmd=7, fd=14)
    at ../sysdeps/unix/sysv/linux/fcntl.c:39
#1  __libc_fcntl (fd=14, cmd=7) at ../sysdeps/unix/sysv/
    linux/fcntl.c:88
#2  0x00007fbbfa7e1296 in ?? () from /usr/lib/x86_64-
    linux-gnu/libapr-1.so.0
#3  0x00007fbbf79165a2 in accept_mutex_on () at prefork.c
    :232
#4  child_main (child_num_arg=child_num_arg@entry=4) at
    prefork.c:611
```

```
#5  0x00007fbbf79169a6 in make_child (s=0x7fbbfb310de0,
    slot=slot@entry=4) at prefork.c:800
#6  0x00007fbbf7916a06 in startup_children (
    number_to_start=1) at prefork.c:818
#7  0x00007fbbf79176e0 in prefork_run (_pconf=<optimized
    out>, plog=0x7fbbfb30c028,
    s=0x7fbbfb310de0) at prefork.c:976
#8  0x00007fbbfb0f19ce in ap_run_mpm (pconf=0
    x7fbbfb33f028, plog=0x7fbbfb30c028,
    s=0x7fbbfb310de0) at mpm_common.c:94
#9  0x00007fbbfb0eb1a6 in main (argc=3, argv=0
    x7fffb086f128) at main.c:777
(gdb) select-frame 8
(gdb) info args
pconf = 0x7fbbfb33f028
plog = 0x7fbbfb30c028
s = 0x7fbbfb310de0
(gdb) info locals
pHook = 0x7fbbfb27f660
n = 0
rv = -4
(gdb) select-frame 9
(gdb) info args
argc = 3
argv = 0x7fffb086f128
(gdb) quit
```

Для создания дампа памяти должны быть соблюдены следующие требования [15]:

- Процесс должен иметь права на запись файла. По умолчанию дамп памяти называется core и создается в рабочей директории приложения.
- Если существует файл, с таким же именем с каким будет создан дамп памяти, то этот файл должен иметь не более одной жесткой ссылки (hard link).
- Директория в которой будет создан дамп памяти должна существовать.
- RLIMIT_CORE и RLIMIT_SIZE (максимальный размер файла) не должны быть равны 0.

1.3 Конфиденциальные данные

В памяти веб-сервера могут находиться следующие конфиденциальные данные, которые могут заинтересовать злоумышленника:

- Cookies, логины, пароли или хеши паролей пользователей. С их помощью можно получить неограниченный доступ к аккаунту пользователя.

- Конфигурационные файлы веб-сервера, исходные коды веб-сервиса и даже переменные окружения содержат информацию, которая может помочь атакующему найти и эксплуатировать различные уязвимости.

- Конфигурационные файлы веб-сервиса, которые могут содержать логины, пароли, токены для доступа к базам данных, FTP серверам и другим сервисам, попадание которых в руки злоумышленника может привести к раскрытию данных пользователей сервиса и компрометации самого сервиса.

- Для шифрования общения между клиентом и сервером используется криптографический протокол TLS [28] или его предшественник SSL. Для аутентификации сервера используются инфраструктура открытых ключей [24]. Одним из ключевых моментов инфраструктуры открытых ключей является то, что приватный ключ должен быть известен только владельцу сервера. Если злоумышленник сможет похитить приватный ключ, то он сможет осуществить атаку "Человек посередине"[17].

2 Цель работы и постановка задачи

Цель работы - исследовать возможность получения конфиденциальных данных из дампа памяти веб-сервера.

2.1 Постановка задачи

На примере веб-сервера Apache и системы управления содержимым сайта Wordpress исследовать возможность получения конфиденциальной информации из дампов памяти веб-сервера. Для достижения поставленной цели необходимо:

1. Ознакомиться с работами в предметной области.
2. Настроить окружение для проведения исследований.
3. Воспроизвести результаты интересующих работ изученных на первом шаге.
4. Вручную проанализировать содержимое нескольких дампов памяти.
5. Написать утилиту для извлечения конфиденциальных данных.
6. Настроить автоматическое создание окружения.
7. Провести автоматическое тестирование на наличие конфиденциальных данных в разных дистрибутивах.

2.2 Актуальность

Дамп памяти может содержать различные конфиденциальные данные, которые могут привести к утечке пользовательских данных или компометации всего веб-сервиса. Таким образом они представляют интерес для злоумышленников.

На январь 2016 года, 33.56% сайтов работали на Apache [18]. Если в частности рассматривать популярную систему управления содержимым Wordpress, то на 2 мая 2017 года 27.9% сайтов работают под её правлением. [19].

В Apache было найдено как минимум 777 [3] уязвимостей, а в интерпретаторе PHP 538 [23].

В экосистеме Wordpress (ядро, плагины, темы) было найдено как минимум 6892 уязвимостей [33], из них 254 уязвимости [32] в ядре.

Некоторые из этих уязвимостей позволяют вызывать создание дампа памяти, а некоторые дают возможность его прочитать. При этом существует неправильно сконфигурированные веб-сервера, которые позволяют читать файлы из корневой директории и соотвествующих поддиректорий.

3 Методы вызова создания дампа памяти

Атакующий может создать дамп эксплуатируя ошибки, приводящие к экстренному завершение работы веб-сервера. Вот примеры таких ошибок [22]:

- Бесконечная рекурсия.
  - Ошибка в библиотеке GD, проверка условия выхода из рекурсии опредленным образом зависела от структуры данного изображения. Специально сконструированное изображение вызывало переполнение буфера и экстренное завершение работы интерпретатора PHP.
- Ошибки в библиотеках.
  - Ошибки обработки заголовков. Существуют расширения PHP,такие как exif, GD, imagemagic и другие, которые позволяют работать с изображениями. Для работы с изображениями необходимо выделить память, а необходимый объем определяется с помощью заголовков изображения. Таким образом выставляя специальные значения заголовков можно добиться экстренного завершнения работы интерпретатора PHP.
- Ошибки в различных встроенных функциях.
  - CVE-2010-3710 [7]. Если был включен режим FILTER_VALIDATE_EMAIL, то злоумышленник мог вызвать отказ работы сервиса, послав очень длинный почтовый адрес, его проверка вызывала использование большого объема памяти.
  - Ошибка в функции pack(). Будучи вызванной со специальными аргументами эта функция заставляет интерпретатор PHP выделить памяти больше чем есть на сервере, что вызовет экстренное завершение работы интерпретатора PHP.
- Ошибки в коде дополнительных модулей.
- Другие ошибки.

4 Способы получения дампа памяти

Атакующий может получить дамп памяти разными способами:

- При неправильной конфигурации веб-сервера становится возможным чтение файлов из поддиректорий корневой директории веб-сервиса [8], куда может быть сохранен дамп памяти. Директория с дампами памяти задается диррективой CoreDumpDirectory в конфигурационных файлах веб-сервера Apache. По умолчанию дамп памяти называется core и создается в рабочей папке приложения.
- Открытый FTP сервер или FTP сервер с пользователем и паролем, которые легко перебрать.
- Directory traversal [9]. Если на сайте присутствует форма просмотра или скачивания некоторого файла, в которой не фильтруется название этого файла, задаваемого пользователем, то злоумышленник может задать название вида "../../../../../etc/passwd"и просмотреть или скачать содержимое этого файла.
- XML external entity [34] + php-wrapper [21] Сайт может быть уязвим, если обрабатывает XML-документы, на содержание которых влияет пользователь, а пользовательские данные не фильтруются должным образом. Атакующий может добавить в документ внешнюю сущность '<!ENTITY xxe SYSTEM "file://etc/passwd»]>', с помощью которой может скачать любой текстовый файл. Для того чтобы скачать произвольный файл, нужно использовать php-wrappers, с их помощью можно конвертировать исходный файл в несколько форматов, например base64: "php://filter/convert.base64-encode/resource=/etc/passwd".
- Другие атаки.

5 Извлечение конфиденциальных данных

5.1 Исследования в предметной области

Heartbleed [12] - критическая уязвимость в популярной криптографической библиотеке OpenSSL. Эта уязвимость позволяла атакующему, пославшему специально сформированный пакет, читать части оперативной памяти размером до 64КБ. Таким образом атакующий мог подслушивать части взаимодействия пользователей с сервисом. Скомпрометированными могли оказаться приватные ключи, используемые для шифрования, логины и пароли пользователей и другие конфиденциальные данные.

После того как была опубликована информация об уязвимости, компания CloudFlare провела собственное расследование [14], целью которого было выяснить, возможно ли извлечь приватный ключ из памяти веб-сервера с уязвимой версией OpenSSL. Во время расследования у инженеров компании

не получилось извлечь приватный ключ. Для изучения проблемы было организовано публичное соревнование [14] с той же самой целью. Каждый желающий мог попробовать извлечь приватный ключ из данного веб-сервера с уязвимой версией OpenSSL.

Победителем данного соревнования стали Fedor Indutny [13] и Ilkka Mattila через 9 часов с момента старта соревнования. Они успешно смогли извлечь приватный ключ. Fedor смог получить ключ отправив около 2.5 миллионов запросов, а у Ilkka получилось это сделать за 100 тысяч запросов.

По итогам соревнования coudflare рекомендовала всем перевыпустить все сертификаты и ускорила перевыпуск сертификатов своих клиентов.

5.2 Методы извлечения приватного ключа RSA [35]

5.2.1 Введение

В основу криптографической системы с открытым ключом RSA положена сложность задачи факторизации произведения двух больших простых чисел. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования за разумное время необходимо уметь вычислять функцию Эйлера от данного большого числа, для чего необходимо знать разложение числа на простые множители.

Важные числа в криптографической системе RSA:

- простые числа p и q
- их произведение $n = p \cdot q$, которое называется модулем
- значение функции Эйлера от числа n: $\varphi(n) = (p-1) \cdot (q-1)$.
- целое число e, которое называется открытой экспонентой ($1<e<\varphi(n)$), взаимно простое со значением функции $\varphi(n)$.
- число d, мультипликативно обратное к числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее сравнению: $d \cdot e \equiv 1 \pmod{\varphi(n)}$.

Пара e, n публикуется в качестве открытого ключа RSA . Пара d, n играет роль закрытого ключа RSA и держится в секрете.

5.2.2 Исходные данные

Известны n и e используемые в схеме RSA, их можно узнать из TLS/SSL сертификата веб-сервиса. Строка длинны u - содержимое дампа памяти процесса веб-сервера, для которой известно, что она содержит приватный ключ d как последовательную запись длинны v бит. Обычно v имеет порядок 10^3.

5.2.3 Алгоритм: последовательный перебор

Простейший способ поиска приватного ключа, который применим ко всем криптосистемам, заключается в том, чтобы получить пару открытый текст/-шифротекст, а затем сканировать данную последовательность и проводить попытки дешифровки шифротекста, используя как ключ каждую последовательность длинны v.

Пару открытый текст/шифротекст можем получить зашифровав некоторый открытый текст публичным ключом.

Редкие ложноположительные результаты могут быть устранены проверкой дополнительных пар текст/шифротекст.

5.2.4 Алгоритм: поиск делителей

Данный подход заключается в поиске в данной строке делители модуля n - p и q. Несмотря на то, что серверу нет необходимости хранить делители в памяти, так как нужно считать лишь $m^d (\mathrm{mod}\ \mathrm{n})$, почти все реализации RSA их хранят в целях оптимизации вычислений.

Структура приватного ключа в библиотеке OpenSSL приведена в листинге 2.

Листинг 2: Структура приватного ключа RSA

```
struct rsa_st
  {
  /* The first parameter is used to pickup errors where
   * this is passed instead of aEVP_PKEY, it is set to 0
       */
  int pad;
  long version;
  const RSA_METHOD *meth;
  /* functional reference if 'meth' is ENGINE-provided */
  ENGINE *engine;
  BIGNUM *n;
  BIGNUM *e;
  BIGNUM *d;
  BIGNUM *p;
  BIGNUM *q;
  BIGNUM *dmp1;
  BIGNUM *dmq1;
  BIGNUM *iqmp;
  /* be careful using this if the RSA structure is shared
       */
  CRYPTO_EX_DATA ex_data;
  int references;
  int flags;
```

```
/* Used to cache montgomery values */
BN_MONT_CTX *_method_mod_n;
BN_MONT_CTX *_method_mod_p;
BN_MONT_CTX *_method_mod_q;

/* all BIGNUM values are actually in the following data
    , if it is not
 * NULL */
char *bignum_data;
BN_BLINDING *blinding;
BN_BLINDING *mt_blinding;
};
```

Структура BIGNUM показана в листинге 3:

Листинг 3: Структура BIGNUM

```
struct bignum_st
{
BN_ULONG *d;    /* Pointer to an array of 'BN_BITS2'
    bit chunks. */
int top;    /* Index of last used d +1. */
/* The next are internal book keeping for bn_expand. */
int dmax;    /* Size of the d array. */
int neg;    /* one if the number is negative */
int flags;
};
```

Делители n хранятся в памяти, а поле d структуры BIGNUM указывает на значение числа.

Используя реализацию этого алгорима, получилось извлечь приватный ключ TLS\SSL из дампа памяти на тестовом стенде.

5.2.5 Алгоритм: анализ структур кода приложения

Альтернативным способом нахождения приватного ключа является поиск структур данных, специфичных для конкретного веб-сервера. Поиск опирается на нахождение структуры данных в которой хранится приватный ключ [листинг 2], в дампе памяти находится эта структура данных, и производится попытка расшифровать некоторое сообщение или найти делитель модуля.

Таким образом работает утилита passe-partout [20], которая может извлекать приватный ключ из памяти работающего веб-приложения. В данный момент использовать эту утилиту для извлечения приватного ключа RSA из дампа памяти нельзя, так как утилита работает только с запущенными процессами.

5.2.6 Оптимизации: поиск мест с высокой энтропией

Процесс поиска можно оптимизировать, если понять какими отличительными характеристиками обладает объект поиска, и искать места с этими характеристиками. Большая часть кода и данных формируется неслучайным образом, а криптографические ключи редко выбирается неслучайно. Таким образом можно утверждать, что криптографические ключи находятся в области с высокой энтропией.

5.3 Извлечение других конфиденциальных данных

5.3.1 Извлечение логинов и паролей, конфигурационных файлов

Изучив тестовые дампы памяти, можно составить список сигнатур, рядом с которыми находятся интересующие нас данные. Например, в тестовых дампах были найдены следующие данные:

- Секреты wordpress.
  NONCE_SALT uk[* 4J5M)/wi #25XFu(uD3...
  AUTH_SALT kP8_d9 f}Ie|T#*hCCKHCH1...
- Пользователь базы данных.
  DB_USER mighty_Manwe
- Имя базы данных.
  DB_NAME sacred_Arda
- Пароль базы данных.
  DB_PASSWORD children_of_Iluvatar
- Глобальная конфигурация Apache.
  IncludeOptional conf-enabled/*.conf
  # Include the virtual host configurations:
  IncludeOptional sites-enabled/*.conf
  # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
  CoreDumpDirectory /var/www/wordpress
- Конфигурация Apache для виртуального хоста.
  CustomLog $APACHE_LOG_DIR/access.log combined
  SSLEngine on
  SSLCertificateFile /etc/apache2/ssl/apache.crt
  SSLCertificateKeyFile /etc/apache2/ssl/apache.key
  </VirtualHost>

5.3.2 Извлечение файлов с известными сигнатурами

Для извлечения файлов с заданными сигнатурами можно использовать утилиту binwalk [4]. Данная утилита сканирует данный файл, ищет известные магические числа [25], например для jpg это \xFF \xD8 \xFF, а для pdf

это \x25 \x50 \x44 \x46. После нахождения известной сигнатуры, binwalk пытается извлечь весь файл по ней 4.

Листинг 4: Пример работы binwalk на дампе памяти

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------

0             0x0             ELF, 64-bit LSB core file
   AMD x86-64, version 1 (SYSV)
31968         0x7CE0          Unix path: /usr/sbin/
   apache2 -k start
45100         0xB02C          Unix path: /usr/lib/x86_64-
   linux-gnu/libxslt.so.1.1.28
19535744      0x12A1780       Ubiquiti firmware header,
   third party, ~CRC32: 0x44415441, version: "
   SSL_CIPHER_AES_256_CBC"
19536088      0x12A18D8       Ubiquiti firmware header,
   third party, ~CRC32: 0x54555245, version: "
   SSL_TLSEXT_SERVER_NAME"
19612800      0x12B4480       Unix path: /var/www/
   wordpress/wp-blog-header.php
19614760      0x12B4C28       Unix path: /var/www/
   wordpress/
19633322      0x12B94AA       HTML document header
19633832      0x12B96A8       HTML document footer
19642449      0x12BB851       Unix path: /var/www/
   wordpress/wp-includes/compat.php0x7f46f4fec312
...
19767912      0x12DA268       XML document, version:
   "1.0"
...
37433704      0x23B3168       Unix path: /var/www/
   wordpress/wp-includes/formatting.php
37648994      0x23E7A62       Copyright string: "
   copyright November 4, 2001"
37859599      0x241B10F       HTML document footer
37859707      0x241B17B       HTML document header
37888488      0x24221E8       Unix path: /var/www/
   wordpress/wp-includes/capabilities.php
37940648      0x242EDA8       Unix path: /var/www/
   wordpress/wp-includes/class-wp-roles.php
207986779     0xC65A05B       Unix path: /usr/lib/apache2
   /modules/mod_authz_host.so
207990872     0xC65B058       Unix path: /etc/apache2/
   mods-enabled/authz_host.load
```

```
207992656       0xC65B750       Unix path: /etc/apache2/
    mods-enabled/authz_user.load
207994971       0xC65C05B       Unix path: /usr/lib/apache2
    /modules/mod_authz_user.so
```

Раскрывается структура файловой системы хоста и присутствуют некоторые части HTML документов.

6 Экспериментальное исследование

6.1 Цель исследования

Целью экспериментального исследования является проверка возможности извлечения конфиденциальных данных из дампа памяти веб-сервера. В частности, проверка работы утилиты по извлечению приватного ключа RSA, проверка наличия таких конфиденциальных данных как логины, пароли, хеши паролей, части конфигурационных файлов и т.д. в дампе памяти веб-сервера.

6.2 Методика исследования

- Создаем чистую виртуальную машину с нужной версией операционной системы.
- На полученной машине устанавливаем все необходимые пакеты для работы веб-сервера и системы управления содержимым сайта (LAMP Stack, Linux, Apache, Mysql, PHP).
- Добавляем пользователей сервиса: рядового посетителя и администратора.
- Делаем несколько типовых для пользователя и администратора действий.
  - Аутентификация.
  - Чтение постов.
  - Редактирование постов.
  - Манипуляция сервисом (администратор).
- Вызываем создание дампа памяти.
- Получаем один или несколько дампов памяти, в зависимости от настроек именования файла дампа памяти.
- Запускаем утилиту для анализа дампов памяти, которая ищет в дампе заранее заданные конфиденциальные данные.
- Сохраняем полученные результаты.

6.3 Инструментарий

Для воспроизводимости экспериментов нужна виртуальная машина или контейнер. Манипуляции с ними можно проводить с помощью соответствующих утилит, но удобный унифицированный интерфейс для этого предлагает инструмент Vagrant [29]. Он позволяет создавать виртуальные машины и контейнеры с произвольной операционной системой, а их начальное состояние задавать с помощью конфигурационного файла.

Таблица 1: Результаты тестирования

Эксперимент	Ubuntu trusty 64
database name	+
database user	+
database password (plaintext)	+
FTP user	+
FTP password (plaintext)	+
wordpress user name	+ (присутствует не во всех дампах)
wordpress user password	-
wordpress user passwords hash	+ (присутствует не во всех дампах)
wordpress user email	+ (присутствует не во всех дампах)
ssl private key	+
Client requests	+ (присутствует не во всех дампах)
web pages (html/js/css)	+- некоторые части html/js/css
Source code	-
process envrironment	+
Apache (global) configs	+
Apache (vhost) configs	+
Wordpress secrets (AUTH_KEY, SECURE_AUTH_KEY, ...)	+

Для установки необходимых программных пакетов и задания конфигурационных файлов используется система управления конфигурациями Ansible [1]. Она позволяет детально задавать процесс установки программного обеспечения с помощью конфигурационных файлов. При этом существует порядка 11 тысяч таких файлов для разнообразных нужд, написанных пользователями Ansible [2], что значительно упрощает процесс упрощает установки и настройки необходимого прграммного обеспечения.

6.4 Результаты

Результаты полученные на операционной системе Ubuntu trusty приведены в таблице 1. Результаты экспериментов на других операционных системах доступны по ссылке

7 Заключение

Получение злоумышленником дампа памяти может привести к полной компрометации веб-сервиса, так как дамп памяти может содержать различную конфиденциальную информацию, такую как логины и пароли к FTP серверам или базам данных, а также из дампа памяти может быть извлечен приватный ключ используемый веб-сервером. Но при этом получение дампа памяти сопряжено с трудностями, так как веб-сервис должен быть неправильно сконфигурирован или содержать уязвимости.

Список литературы

[1] Ansible. https://www.ansible.com/, .

[2] Ansible-galaxy. https://galaxy.ansible.com/list#/roles, .

[3] Apache security vulnerabilities. https://www.cvedetails.com/vulnerability-list/vendor_id-45/Apache.html.

[4] Binwalk. https://github.com/devttys0/binwalk.

[5] Buffer overflow. https://en.wikipedia.org/wiki/Buffer_overflow.

[6] Program error signals. http://www.gnu.org/software/libc/manual/html_node/Program-Error-Signals.html#Program-Error-Signals.

[7] Cve-2010-3710. http://www.cvedetails.com/cve/cve-2010-3710.

[8] Directory listing configuration. https://wiki.apache.org/httpd/DirectoryListings, .

[9] Directory traversal attack. https://en.wikipedia.org/wiki/Directory_traversal_attack, .

[10] Безопасность доступа к памяти. https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0_%D0%BA_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8#.D0.A2.D0.B8.D0.BF.D1.8B_.D0.BE.D1.88.D0.B8.D0.B1.D0.BE.D0.BA_.D0.BF.D0.B0.D0.BC.D1.8F.D1.82.D0.B8.

[11] Heap overflow. https://en.wikipedia.org/wiki/Heap_overflow, .

[12] Heartbleed. http://heartbleed.com/, .

[13] Cracking cloudflare's heartbleed challenge. http://darksi.de/9.heartbleed/, .

[14] Heartbleed cloudflare challenge. https://blog.cloudflare.com/answering-the-critical-question-can-you-get-private-ssl-keys-using-heartbleed/, .

[15] Core dump man. https://linux.die.net/man/5/core.

[16] Memory corruption. https://en.wikipedia.org/wiki/Memory_corruption.

[17] Man-in-the-middle attack. https://en.wikipedia.org/wiki/Man-in-the-middle_attack.

[18] Apache http server. https://ru.wikipedia.org/wiki/Apache_HTTP_Server, .

[19] Usage of content management systems for websites. https://w3techs.com/technologies/overview/content_management/all/, .

[20] In-memory extraction of ssl private keys. https://c0decstuff.blogspot.nl/2011/01/in-memory-extraction-of-ssl-private.html.

[21] Php wrappers. https://secure.php.net/manual/en/wrappers.php.php, .

[22] https://ilia.ws/archives/5_Top_10_ways_to_crash_PHP.html, .

[23] https://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/PHP-PHP.html, .

[24] Public key infrastructure. https://en.wikipedia.org/wiki/Public_key_infrastructure.

[25] List of file signatures. https://en.wikipedia.org/wiki/List_of_file_signatures, .

[26] Termination signals. http://www.gnu.org/software/libc/manual/html_node/Termination-Signals.html, .

[27] Stack overflow. https://en.wikipedia.org/wiki/Stack_overflow.

[28] Transport layer security. https://en.wikipedia.org/wiki/Transport_Layer_Security.

[29] Vagrant. https://www.vagrantup.com/.

[30] Core dump wiki. https://en.wikipedia.org/wiki/Core_dump, .

[31] Web-server. https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80, .

[32] Wordpress core security vulnerabilities. https://www.cvedetails.com/vulnerability-list/vendor_id-2337/product_id-4096/, .

[33] Wordpress security vulnerabilities. https://wpvulndb.com/, .

[34] Xxe. https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing.

[35] Adi Shamir and Nicko van Someren. Playing hide and seek with stored keys. http://hawaii.ms11.net/keyhide2.pdf.