How to: Actually attack computers at cafes

Felix Ryan

YGHT

Who am I?

- I’m Felix
- I’m a pen tester

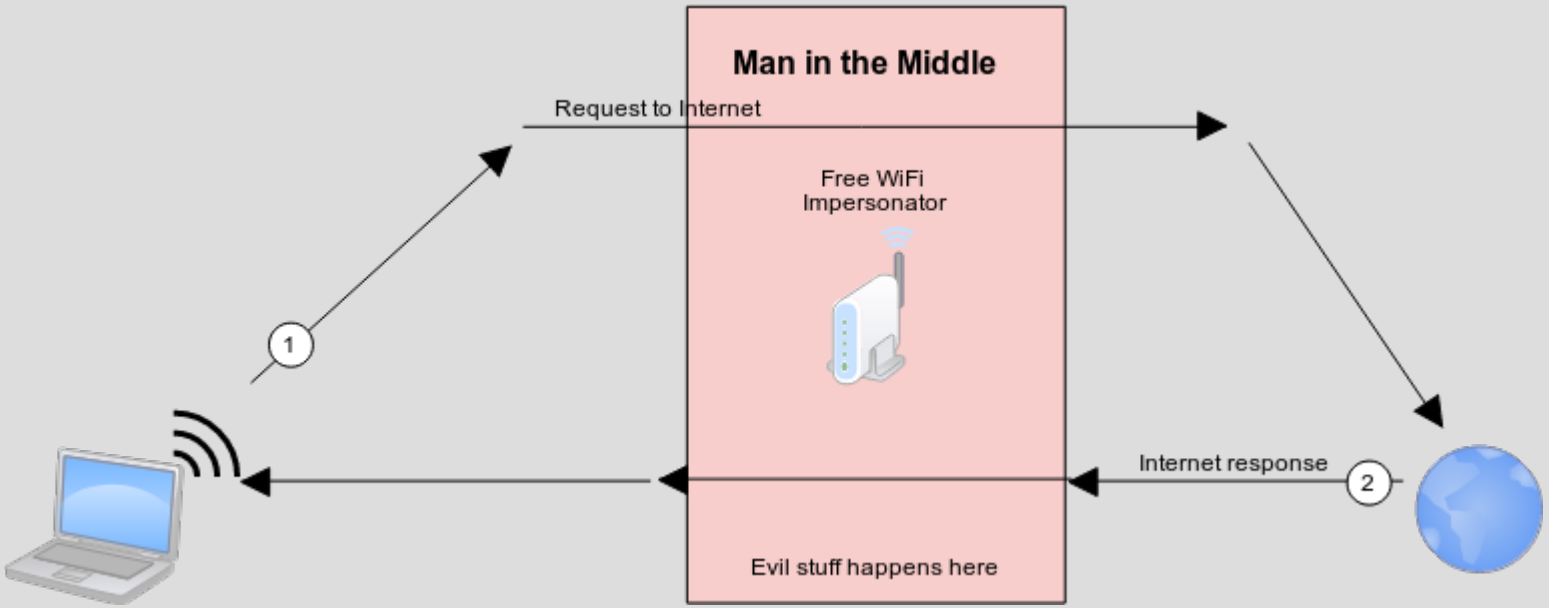
YGHT

Why this research?

- Masters degree dissertation
- Client didn’t just take my word for it
- Couldn’t find a tool

YGHT

Open WiFi MitM Condition



YGHT

What can you do with MitM conditions

- Listen to the communication
- Change the communication
- Stop the communication

YGHT

What I set out to do

- Create evil WiFi networks
- MitM some users
- Grab creds
- Politely inform my client that they were wrong (and “ner ner nee ner ner”)
- Convinced this would be easy...

YGHT

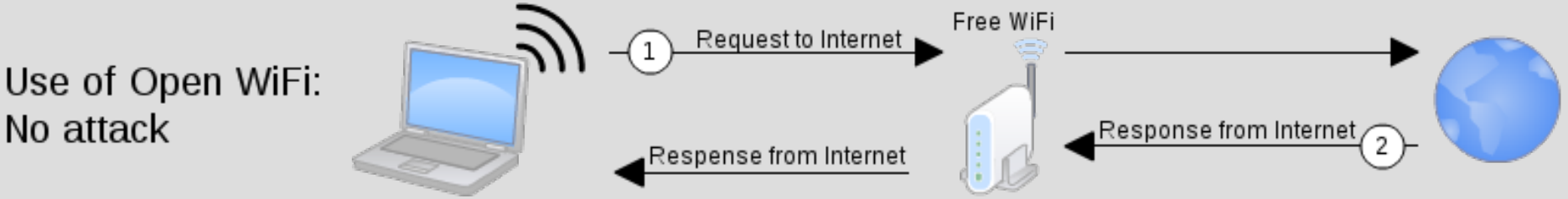
It turns out that encryption is a thing…

- can be done at all the layers
- not much plaintext auth these days
- confirmed I needed another way of getting creds

YGHT

The idea

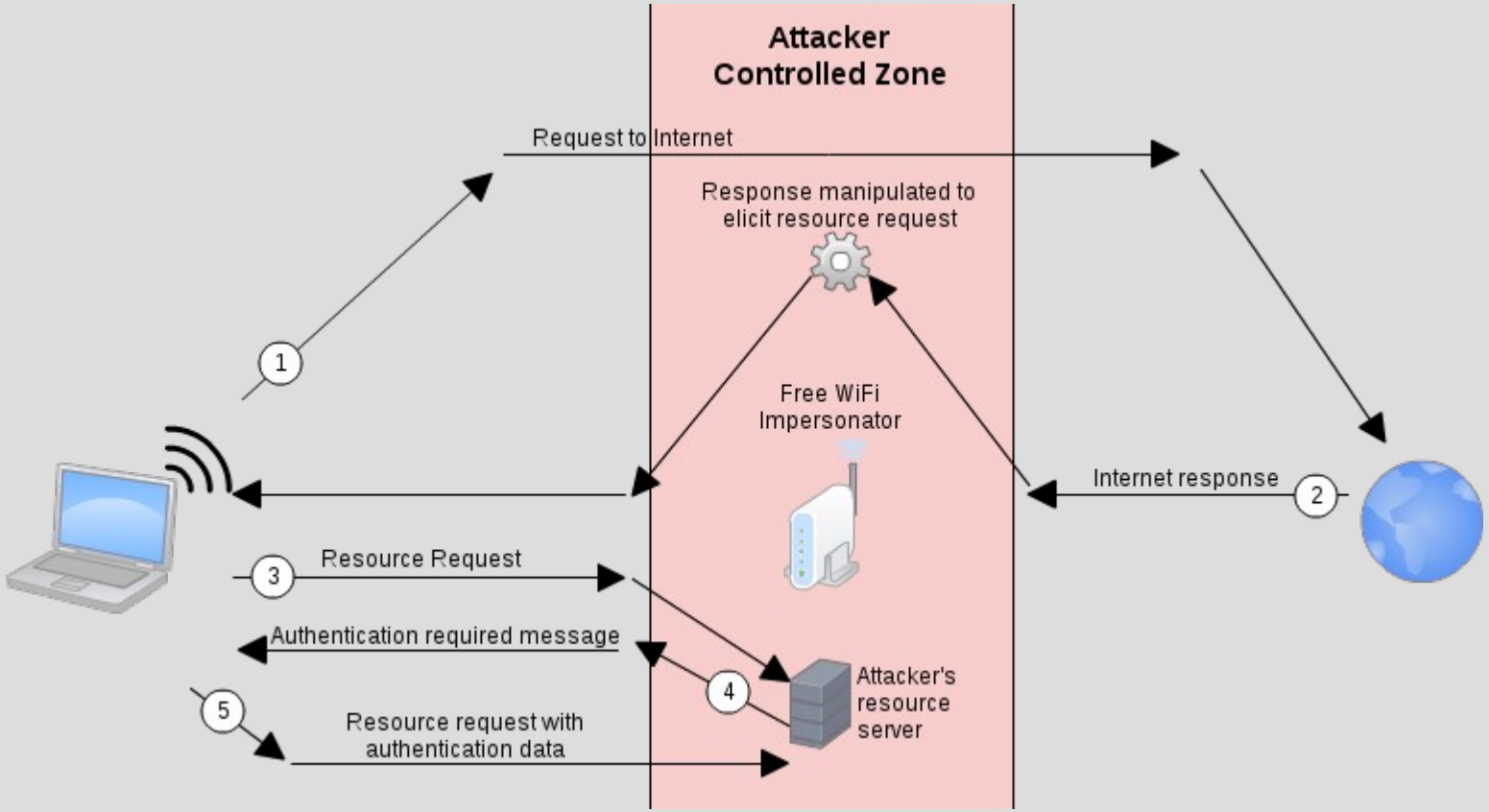
Go from this:



YGHT

The idea

To this:



YGHT

What I actually did

- Used a WiFi Pineapple
- Set up a wireless network to simulate a cafe
- Called it “DANGER ZONE – DO NOT USE” (and still got random people connecting)

YGHT

It looked a bit like this:

YGHT

Developed a tool

- Butchered someone else's tool into submission (Responder.py in particular)
- Added my own code
- Sulked in the corner when it didn't work
- Repeat
- Eventually have some success

YGHT

My tool

A transparent proxy that injects HTML tags into HTTP responses such as:

Couldn't get plaintext creds
Got NetNTLM hashes instead

ETAC = Evil Twin Authentication Capture

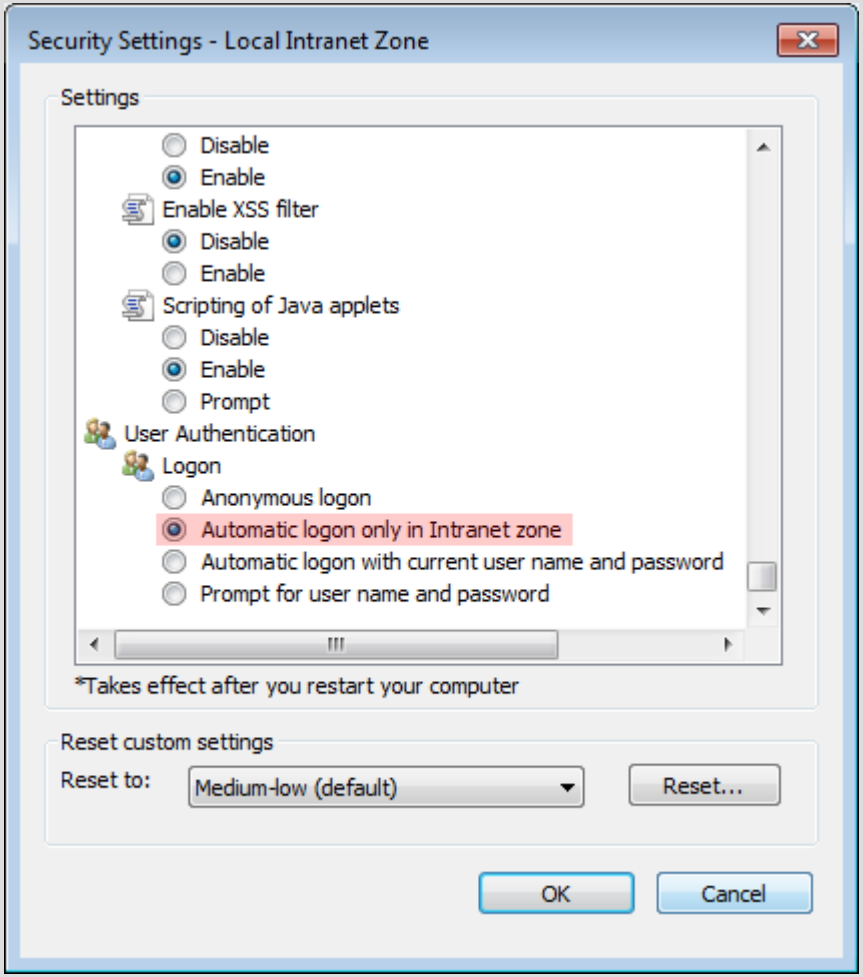
YGHT

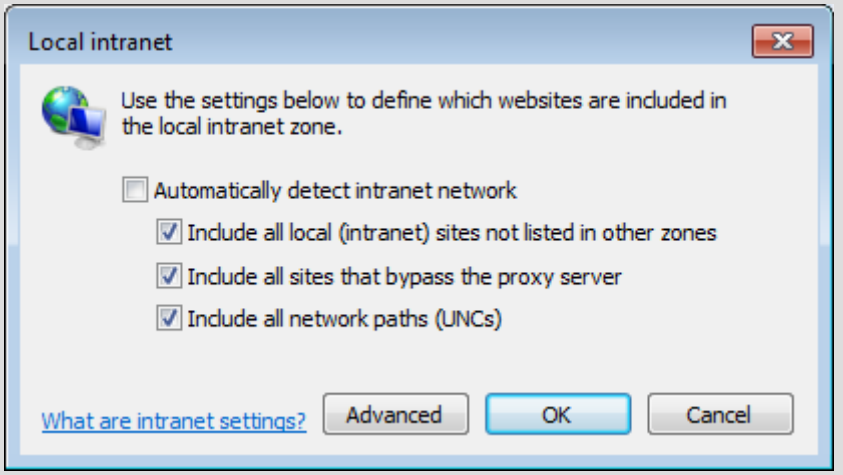
Windows auth

Windows 7 test machine
Kept with default config
This means IE…
Remember: NetNTLM auth is the goal

YGHT

Windows Auth - The dot rule

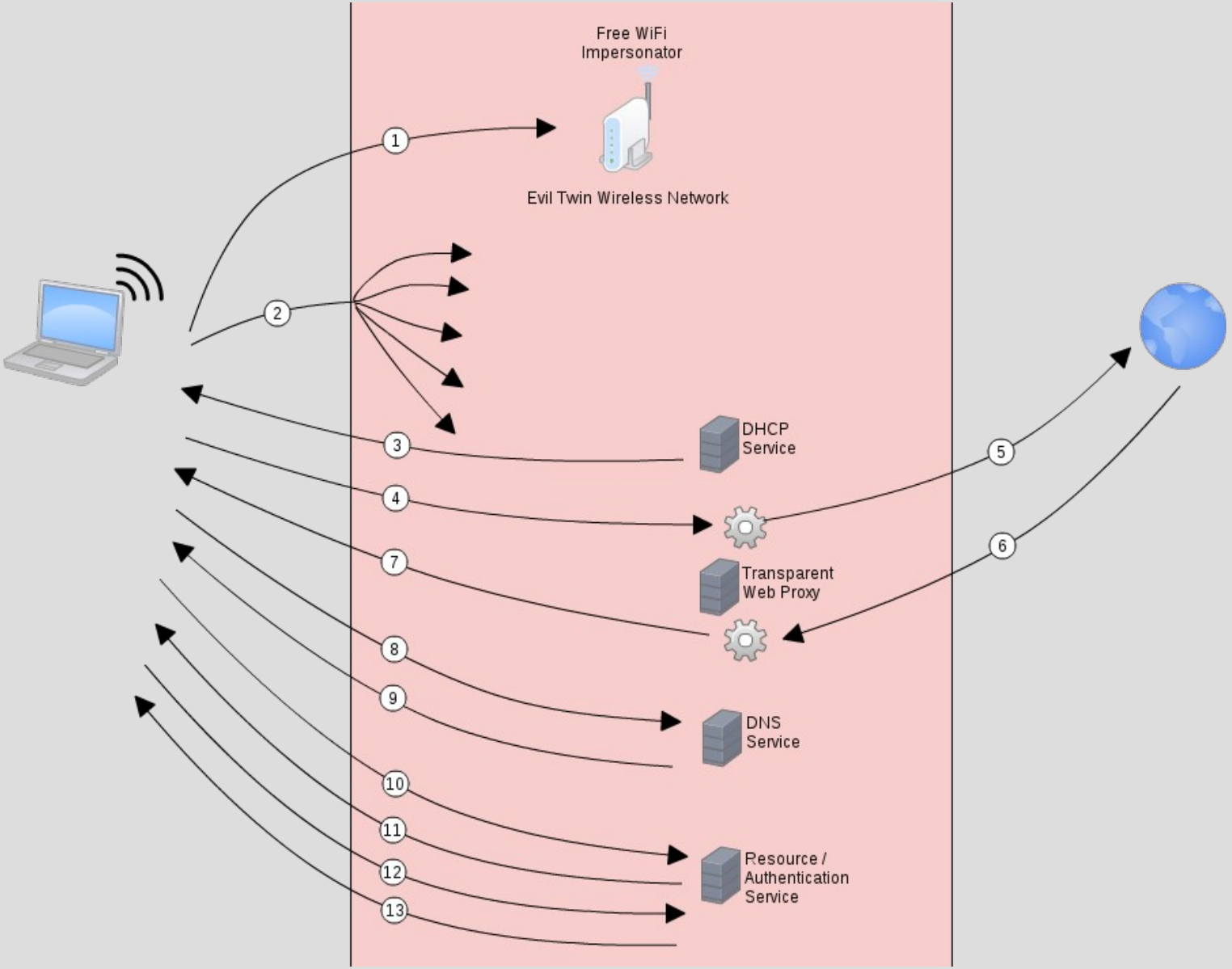




I needed a DNS server

YGHT

The final attack flow



YGHT

The Challenges

So far so good?
Ehhh… not quite

YGHT

HTTP is a pain

- Coding this without many libraries
- Random HTTP status codes
- Different HTTP versions
- HTTP request headers

(Compression / Encoding / Caching / Ranges / Connection status / content types)

- “Normal” error handling
- Differences in declared and transparent proxies
- Response size and browser behaviour
- Chunking

YGHT

Transaction size and chunking

Declared size of response:

```
Content-Length: 244271
```

Chunked Transfer Encoding (CTE):

```
Transfer-Encoding: chunked

1cfe (chunk size markers)

Response ends with
'\r\n0\r\n\r\n'
```

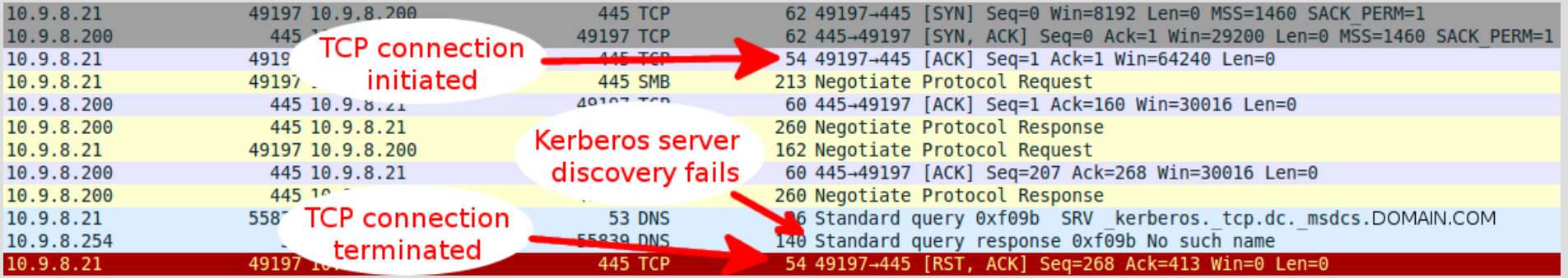
```
Transfer-Encoding: chunked

(but no chunk markers)

Response ends with
'\r\n0\r\n\r\n'
```

YGHT

Successes and failures

- Active Directory joined vs unjoined

```
[*] HTML poisoning performed (SrcAdd=172.16.42.117 DstHost=stackoverflow.com RxConn=0)
...SNIP...
[SMB] NTLMv2-SSP Client : 172.16.42.117
[SMB] NTLMv2-SSP Username : IE11WIN7\IEUser
[SMB] NTLMv2-SSP Hash :
IEUser::IE11WIN7:1122334455667788:B6EC84566FB6ADFDA7FFD18DB6FD5DF3:010100000000002F5B9F1C37CDD101F
F3B86...SNIP...F31BC8BC3F48005DB1E6B426C66DEBE9EA92316671CC10A00100000000000000000000000000000000
0900260063006900660073002F00680074006D006C0069006E006A006500630074002E006C0061006E0000000000000000
[SMB] Requested Share: \\EVIMACHINE\SHARE
```



(damn you Kerberos!)

Summary

- Tool is on GitHub
- Could develop it further

- AD joined workstation = boo
- Non-AD = yay

YGHT

Questions?

x@yg.ht

https://github.com/yg-ht/ETAC
(moving to gitlab… brb)

Thanks to all those who's tools I abused

YGHT