# OPSEC Fundamentals for Remote Red Teams

# Who am I?

- Michael Allen
  - Also go by Wh1t3Rh1n0 or just "Rhino"
  - Professional penetration tester/red teamer since 2014
  - Security Analyst at Black Hills Information Security since 2019
  - Certifications: OSCE, MLSE, CISSP, ...
- No formal training in OPSEC
- Still make lots of mistakes. Just sharing some lessons I've learned.

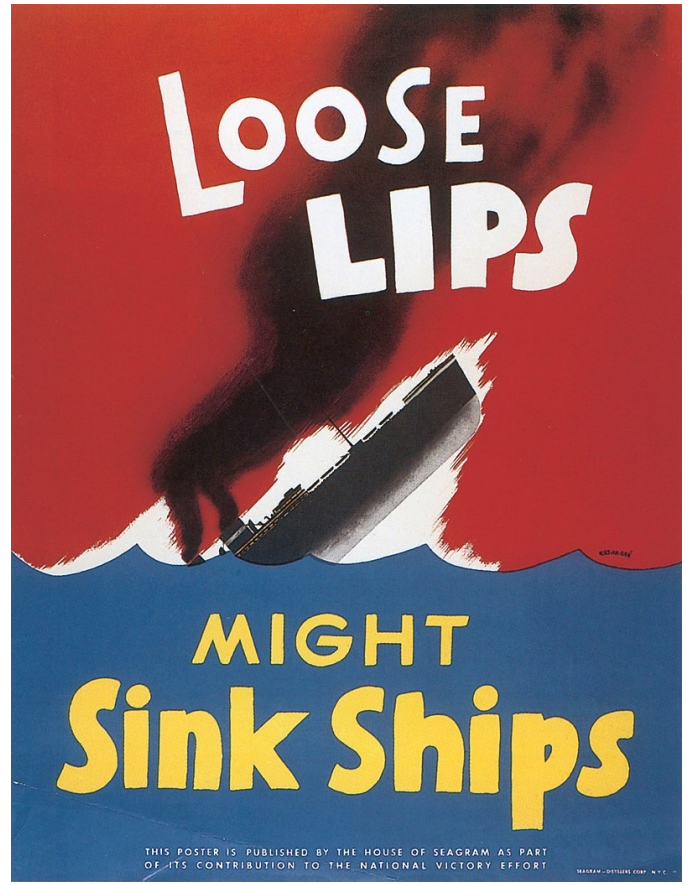
# What is this talk about?

- OPSEC Fundamentals for Remote Red Teams

# What is this talk about?

- **OPSEC** Fundamentals for Remote Red Teams
- "Operations security (OPSEC) is a process that identifies critical information to determine if friendly actions can be observed by enemy intelligence, determines if information obtained by adversaries could be interpreted to be useful to them, and then executes selected measures that eliminate or reduce adversary exploitation of friendly critical information."

  -- Wikipedia (highlighting added)



# What is this talk about?

- OPSEC Fundamentals for Remote **Red Teams**
- Red Team Exercises
  - Perform a cyberattack against the target organization.
  - Attack success demonstrates business impact. Examples:
    - Theft of intellectual property/confidential data
    - Unauthorized financial transactions
    - Unauthorized access to systems, accounts, or other assets
  - Unannounced to the organization's security team
  - The security team (Blue Team) of the target organization **actively** identifies and responds to suspicious activity - just like a real attack.

# What is this talk about?

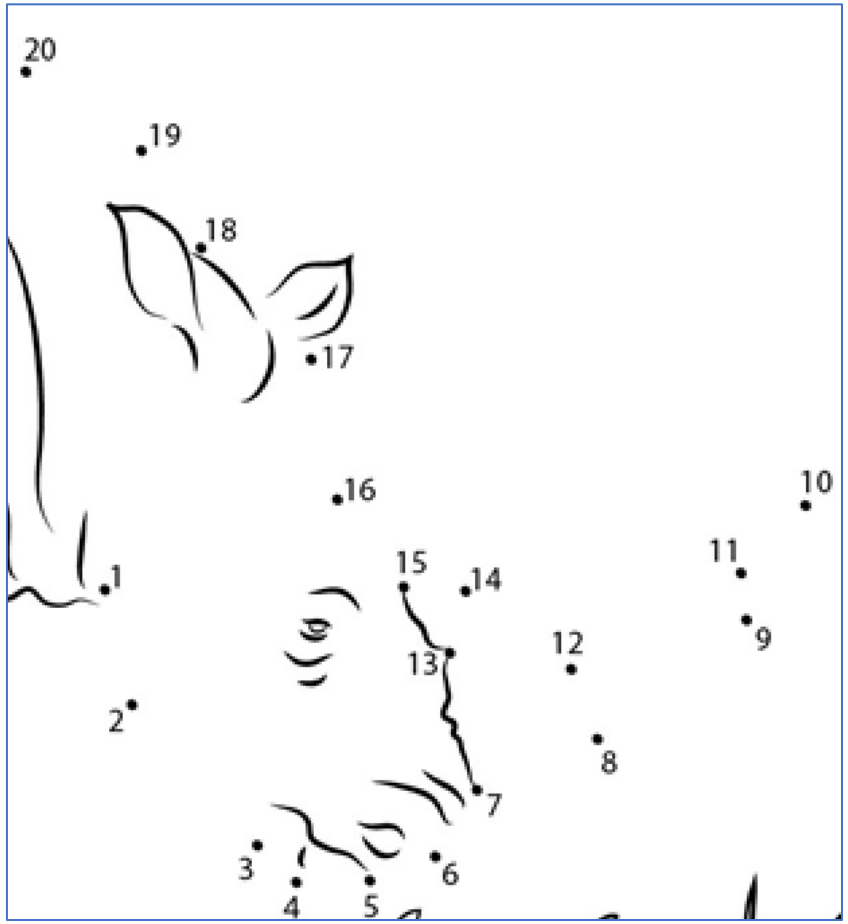
- OPSEC Fundamentals for **Remote** Red Teams
  - OPSEC for remote red teams will be different than for teams that are on-site.
  - No physical or wireless security components.
  - Not plugging any devices into the network.
- Other limitations:
  - This talk will focus on red team OPSEC up to the initial breach.
  - Additional OPSEC considerations are introduced once the red team starts post-exploitation and lateral movement in the target environment.

# Why is OPSEC important for red teams?

- Indicators of suspicious activity are well known
  - Open sharing of information is how InfoSec works
- Modern blue teams have many sources to inform them of suspicious activity:
  - Logs on Internet-facing systems/services
  - Logging on systems and connections that make egress from the internal network (e.g. workstations)
  - Various of third-party "threat intelligence" services
  - Analysis and correllation of information collected from those sources

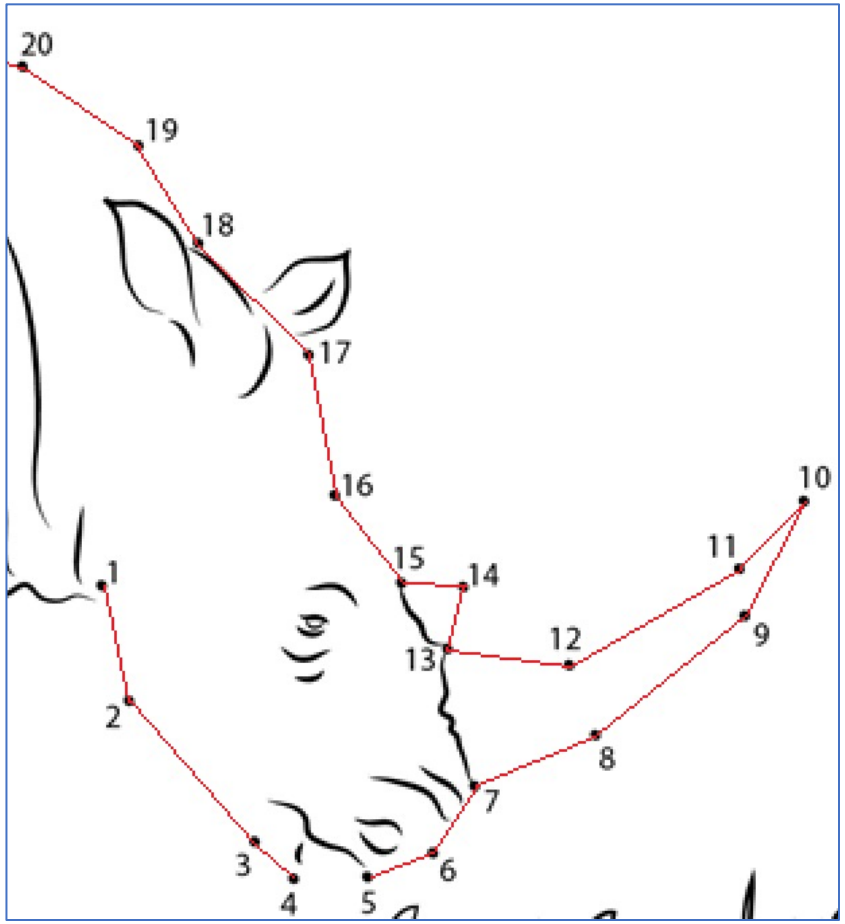
# Red doesn't want blue to connect the dots

- Each time the red team interacts with the target organization, it leaves a "dot".



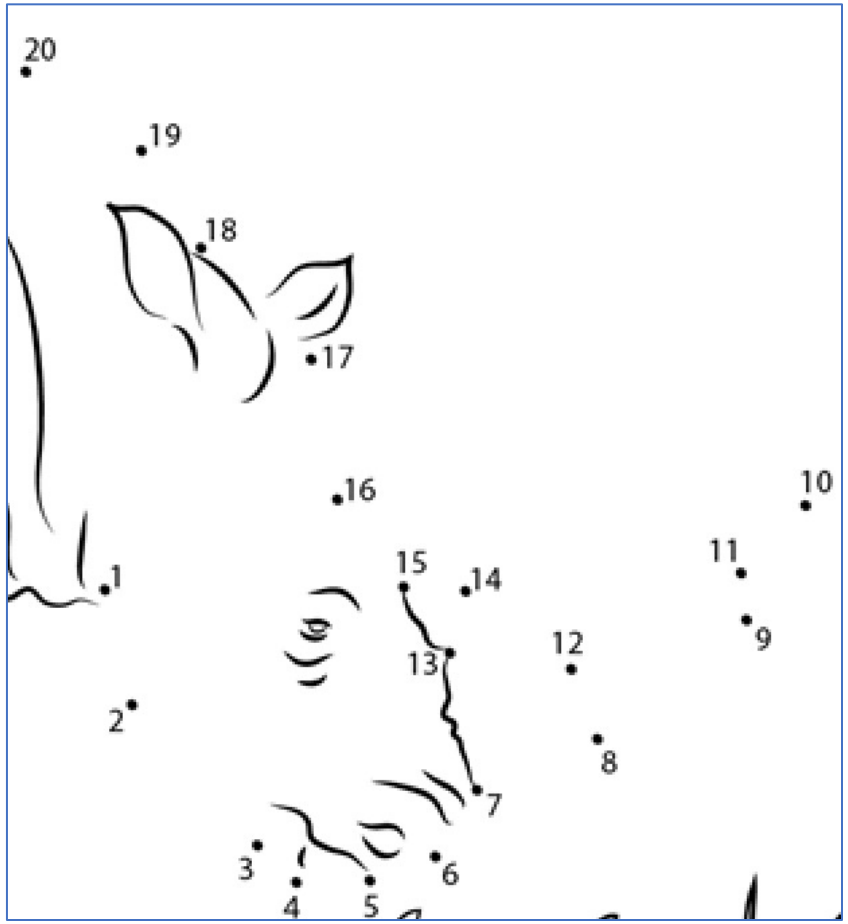
# Red doesn't want blue to connect the dots

- Each time the red team interacts with the target organization, it leaves a "dot".
- With enough dots, the blue team can identify red team tools and infrastructure, and prevent attacks before they occur.
- This is especially frustrating if the blue team connects the dots before the red team knows it.



# Possible countermeasures

How can the red team keep the blue team from connecting the dots?



# Possible countermeasures

How can the red team keep the blue team from connecting the dots?

1. Don't leave dots.

# Possible countermeasures

How can the red team keep the blue team from connecting the dots?

1. Don't leave any dots.
2. When leaving a dot is unavoidable, don't leave clues that associate it with other dots.

# Possible countermeasures

How can the red team keep the blue team from connecting the dots?

1. Don't leave any dots.
2. When leaving a dot is unavoidable, don't leave clues that associate it with other dots.
3. Create dots that associate other dots with unrelated dots.

# Possible countermeasures

How can the red team keep the blue team from connecting the dots?

1. Don't leave any dots.
2. When leaving a dot is unavoidable, don't leave clues that associate it with other dots.
3. ~~Create dots that associate other dots with unrelated dots.~~

# Other threats to the red team

- Vendors of infrastructure services (e.g. AWS, Azure)
  - Possibility of being blocked for violating Terms of Service
  - May interfere with future projects
- Data leaks
  - Potentially damaging to the company reputation
  - May reveal the red team's parent organization
  - May reveal the identity of the red team's customers
- Real-world threat actors
  - Network and application security vulnerabilities

# Steps for assessing red team actions

1. Plan likely actions
2. Brainstorm information disclosed by each action
3. Assess whether the disclosed information can be used to identify the red team
   - What information is disclosed?
   - Who can observe the disclosed information?
   - It is likely for the information be used against the red team?
4. Adjust plans to mitigate the risks of information disclosure

# Steps for assessing red team actions

- Apply the steps to each phase of the exercise
- Document and apply in advance to common actions (standard operating procedures)
  - Setup (Local VMs, C2 servers, domains, ...)
  - Reconnaissance (Search queries, port scans, browsing target's web site, ...)
  - Attacks (Password guessing, phishing, ...)
- Apply as necessary before using new tools/techniques
  - Setup a test environment to mimic the target
  - Observe the action from the target's point of view

# My Standard Operating Procedures for Red Team OPSEC

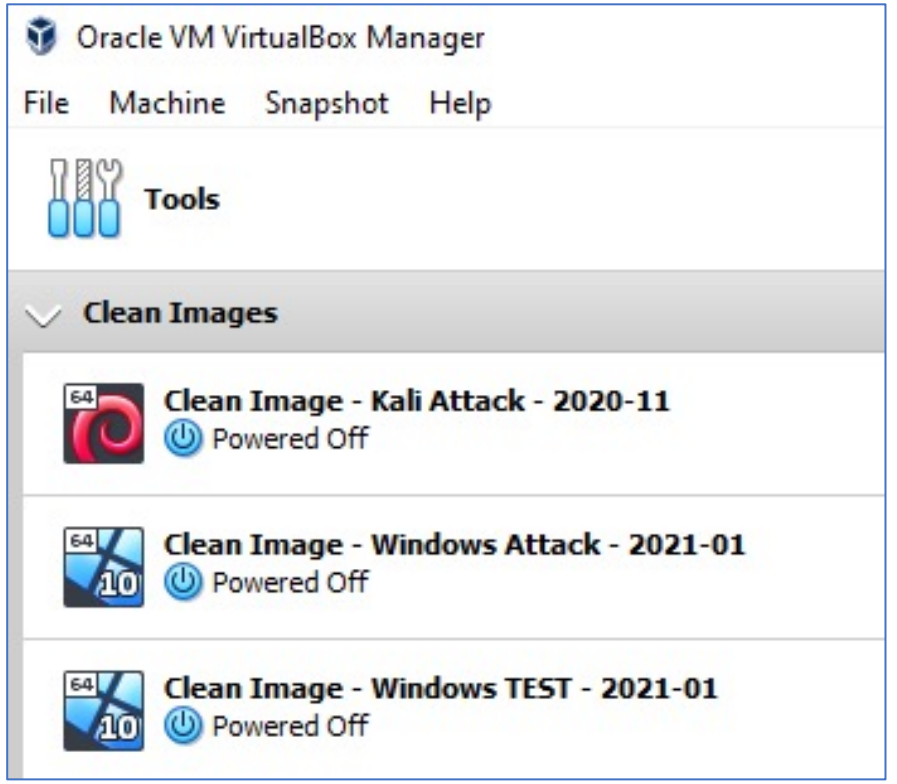
1. Local workstation setup
2. Source IP addresses
3. Other third-party services
4. Payloads and network services
5. Testing new tools

# 1. Local workstation setup

# Use Virtual Machines

- Quick deployment:
  - System configured for OPSEC best practices
  - Tools - installed, customized, and configured
  - Clean environment - no artifacts left over from other customers
  - Additional VMs when necessary
- Images can be updated/modified without a full rebuild
- Keep a checklist for your build and configuration process for each OS
  - Update it as you learn more
- Protect against accidental compromise - downloaded hacking tools

# Use Virtual Machines




- Kali Linux Attack VM
- Windows Attack VM
  - All antivirus, firewall, and other defenses disabled.
  - Browser configurations adjusted for OPSEC.
  - Payload development and execution testing.
- Windows Test VM
  - Stock - Latest Windows build. No extra libraries.
  - All defenses left intact. Additional defenses installed to mimic the target.
  - Payload portability and AV testing. (Revert snapshot after.)
  - Used for authenticated connections to the target network.

# Operating System modifications

- All local VMs:
  - Change the hostname
    - Remote: localhost, DESKTOP, PC
    - On-site: PRINTER, deskjet, LEXMARK
  - Change the local user name
  - Set a strong password (no kali:kali)
- Windows Attack VM:
  - Completely disable Windows Defender
- Windows Test VM:
  - Set the username, hostname, and domain name to match real/similar values used by the target

# Why change the hostname and username?

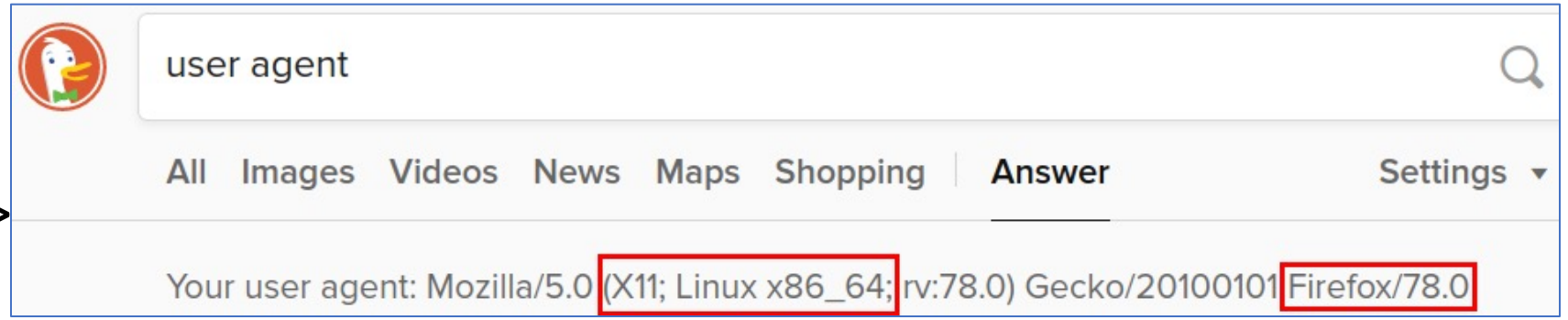
- Many opportunities to leak your local host or user name to the target
- NTLM authentication requests from the target's website
- Interaction with other network services
- Metadata in PDF/Microsoft Office document payloads
- "Honeypot" documents that phone home
- **Prevention against others that may be unknown**

# Why change the domain name?

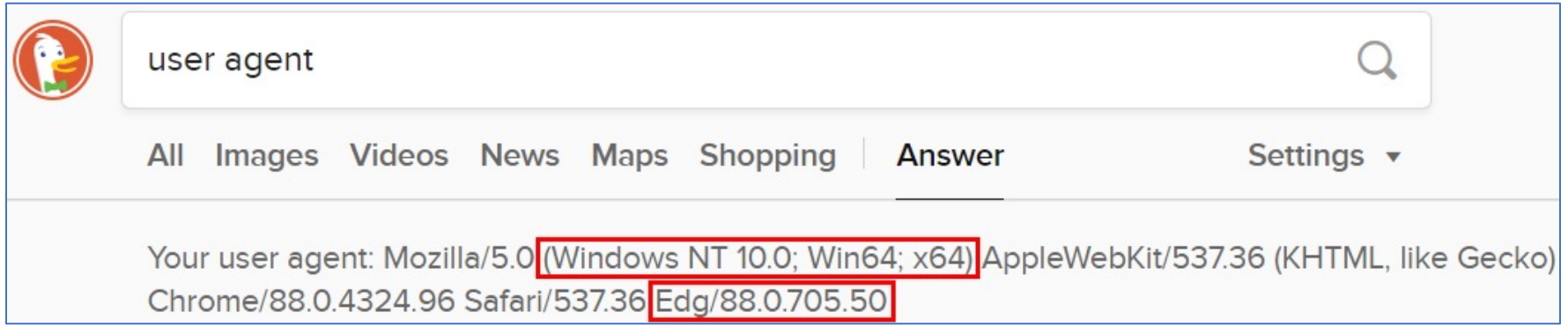
- VPN clients commonly report criteria about the system to the VPN server that causes the connection to be allowed or denied:
  - Domain name
  - Host name
  - **Domain** user name
  - Operating system version and patch level
  - Whether expected EDR/AV products are installed
- It's trivial for blue teams to alert on VPN connections from systems not joined to the expected domain name.
- Also useful when testing payloads keyed to the target's domain name.

# Tool configurations

- "User-Agent" header - Tells web servers:
  - Browser/client software and version number
  - OS version
- Kali Linux ----->




- Edge ------->



# User-Agent header

- Example: Nmap default User-Agent

```
┌──(kali㉿kali)-[~]
└─$ nmap -p80 --script=default 127.0.0.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-01-23 13:49 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0023s latency).

PORT   STATE SERVICE
80/tcp open  http
|_http-title: Directory listing for /
```

```
GET / HTTP/1.1
Host: localhost
Connection: close
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
```

# User-Agent header

- Example: WPScan default User-Agent

```
┌──(kali㉿kali)-[~]
└─$ wpscan --url 127.0.0.1
```

```
GET / HTTP/1.1
Host: 127.0.0.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: WPScan v3.8.12 (https://wpscan.com/wordpress-security-scanner)
Referer: http://127.0.0.1/
```

# User-Agent header

- Example: EyeWitness default User-Agent

```
┌──(kali㉿kali)-[~]
└─$ /opt/EyeWitness/Python/EyeWitness.py --web --single http://127.0.0.1
```

```
GET / HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

# Changing the default User-Agent

- Pick a user agent that isn't the default value for a hacking tool
  - Consider matching with the type and/or amount of traffic the site receives
  - You may not want to use the same user agent string for every tool
- Examples:
  - Google Chrome on Windows 10

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/87.0.4280.88 Safari/537.36
```

  - Googlebot - Web crawler of the Google search engine

```
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
```

# Changing the default User-Agent

- Command-line tools:
  - Add to ~/.bashrc or ~/.zshrc:

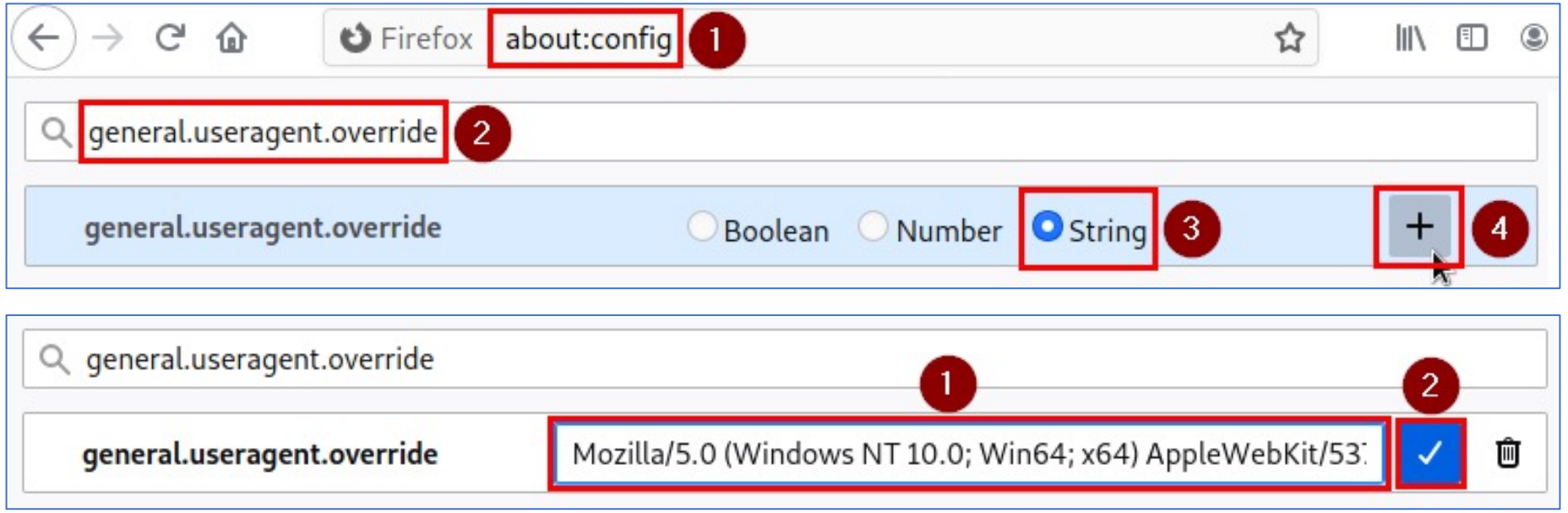
```
export AGENT="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36"

alias curl="curl -A '$AGENT'"

alias wget="wget -U '$AGENT'"

alias nmap="nmap --script-args=\"http.useragent='$AGENT'\""
```

- Use with additional commands like:

```
wpscan --ua "$AGENT" --url 127.0.0.1
```

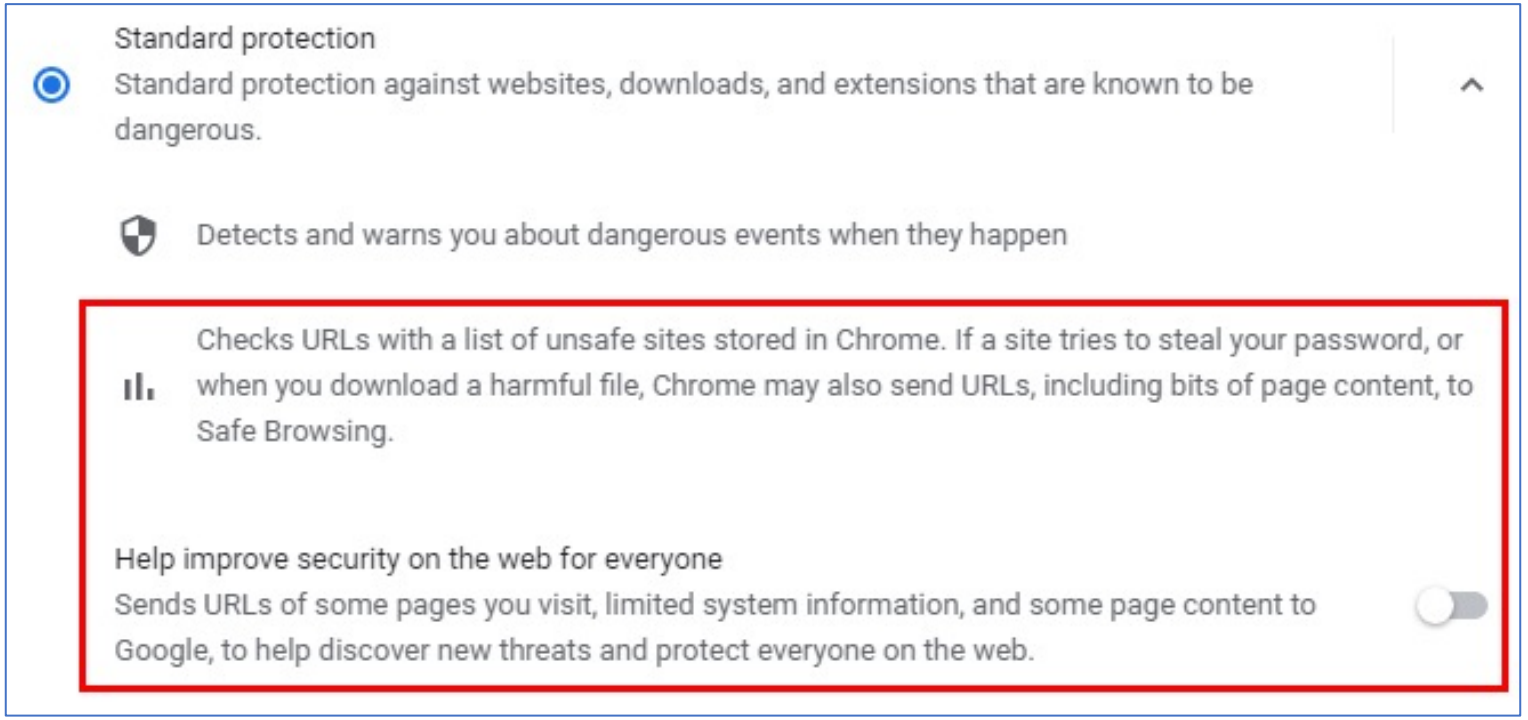
# Changing the default User-Agent

- In Firefox:
  - Use an extension like "User-Agent Switcher and Manager"
  - Or set the "general.useragent.override" string in "about:config"



# Watch out for browser leaks

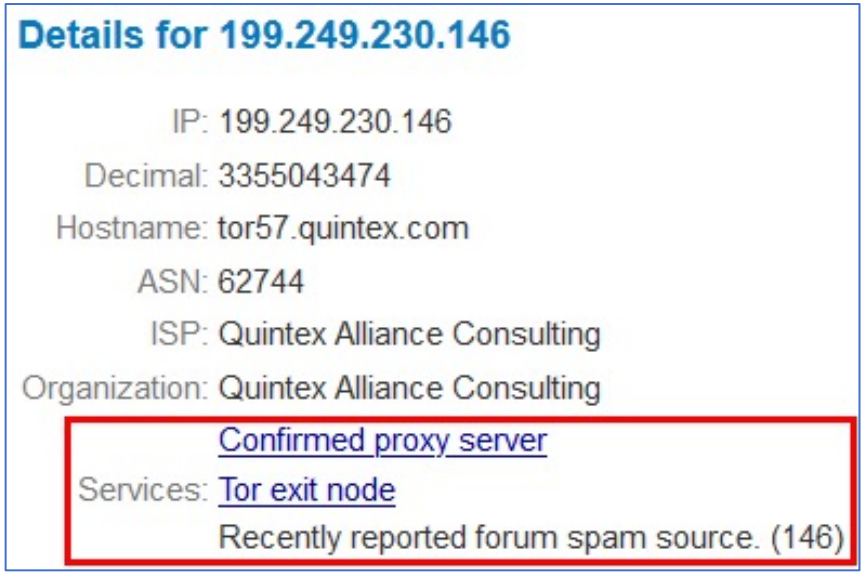
- Use unique tokens, User-Agent detection, etc. to block or redirect bots visiting payload URLs and landing pages
- Examine browser settings that send data to third-parties

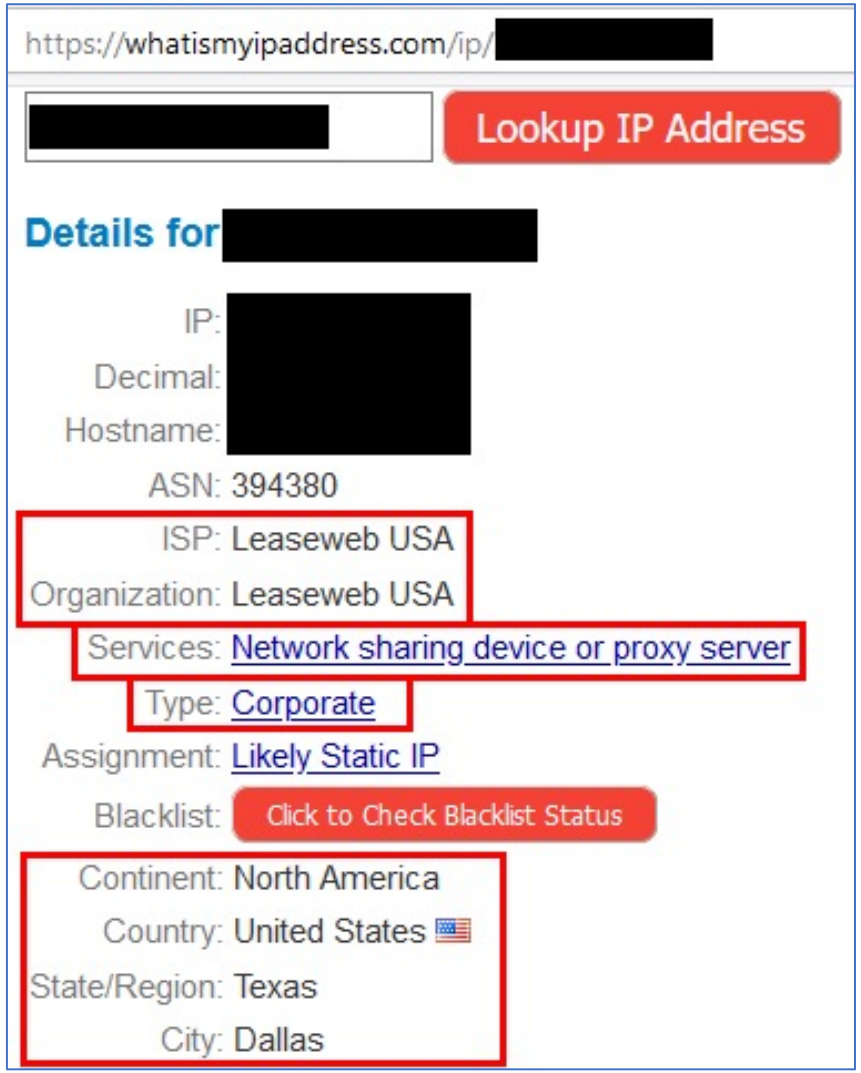


# 2. Source IP addresses

# What might make an IP address suspicious?

1. Association with other suspicious traffic
2. Physical location
3. Service provider and Type of connection
   - Residential/Commercial
   - Mobile
   - Cloud
   - VPN
   - Proxies
   - TOR



# Association with other actions

- Example scenario:
  1. Red team executes a port scan of target infrastructure from Source IP Address A.
  2. Blue team detects the port scan.
  3. A few days later, the red team sends a phishing email from the same Source IP Address.
- What might the blue team do?
  - Block all traffic and emails from the suspicious IP - phishing email fails.
  - Scrutinize further actions associated with the IP
    - Phishing email gets analyzed by the blue team
    - Blue team gathers additional intelligence on the red team - C2 infrastructure, landing pages, domains, etc.

# Physical location

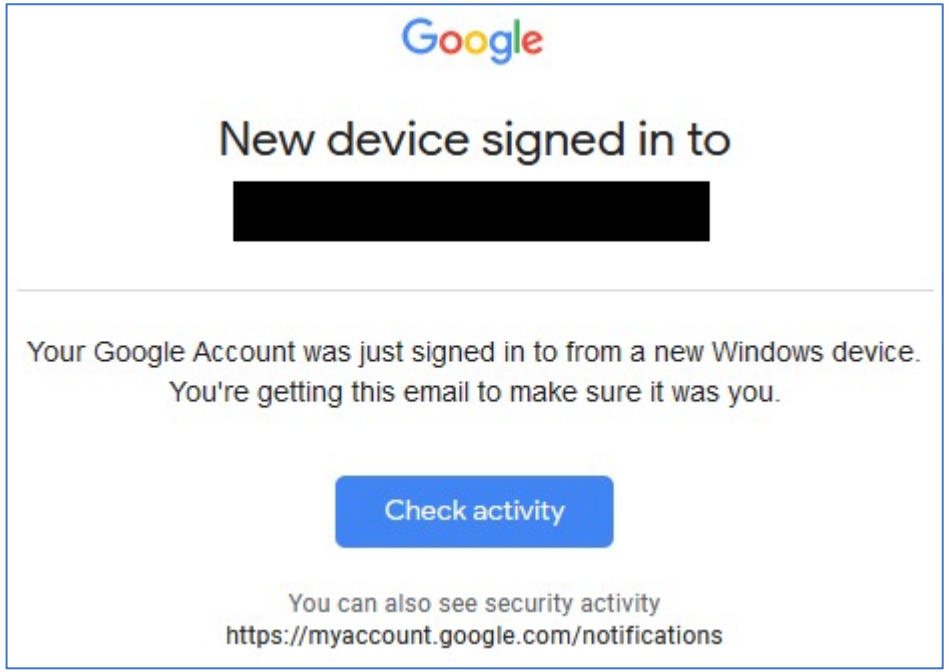
- Example scenario:
  - The target organization has offices in New York and Los Angeles.
  - The red team discovers login credentials and accesses the corporate VPN from a source IP address in Atlanta.
  - The blue team receives an alert on a remote login from an unfamiliar location.
- Possible outcome:
  - The blue team contacts the user whose account was compromised and confirms that the user is not in Atlanta. (May be automated.)
  - The user's password is changed and all login sessions are closed.
  - The red team no longer has access to the corporate network.

# Service provider and Type of connection

- May already be blocked/flagged:
  - Block all logins from known VPNs and TOR exit nodes
  - Block logins from cloud service provider IP addresses
  - Block logins from locations perceived as unusual/high-risk
- Correllate activities to identify the red team's service provider
- Example:
  1. Port scanning observed from a Linode VPS
  2. Typosquatted domain observed hosted on a different Linode IP address
  3. Suspicious emails to non-existent mailboxes sent from a third Linode IP
  4. The blue team might alert on future traffic to/from Linode or block it

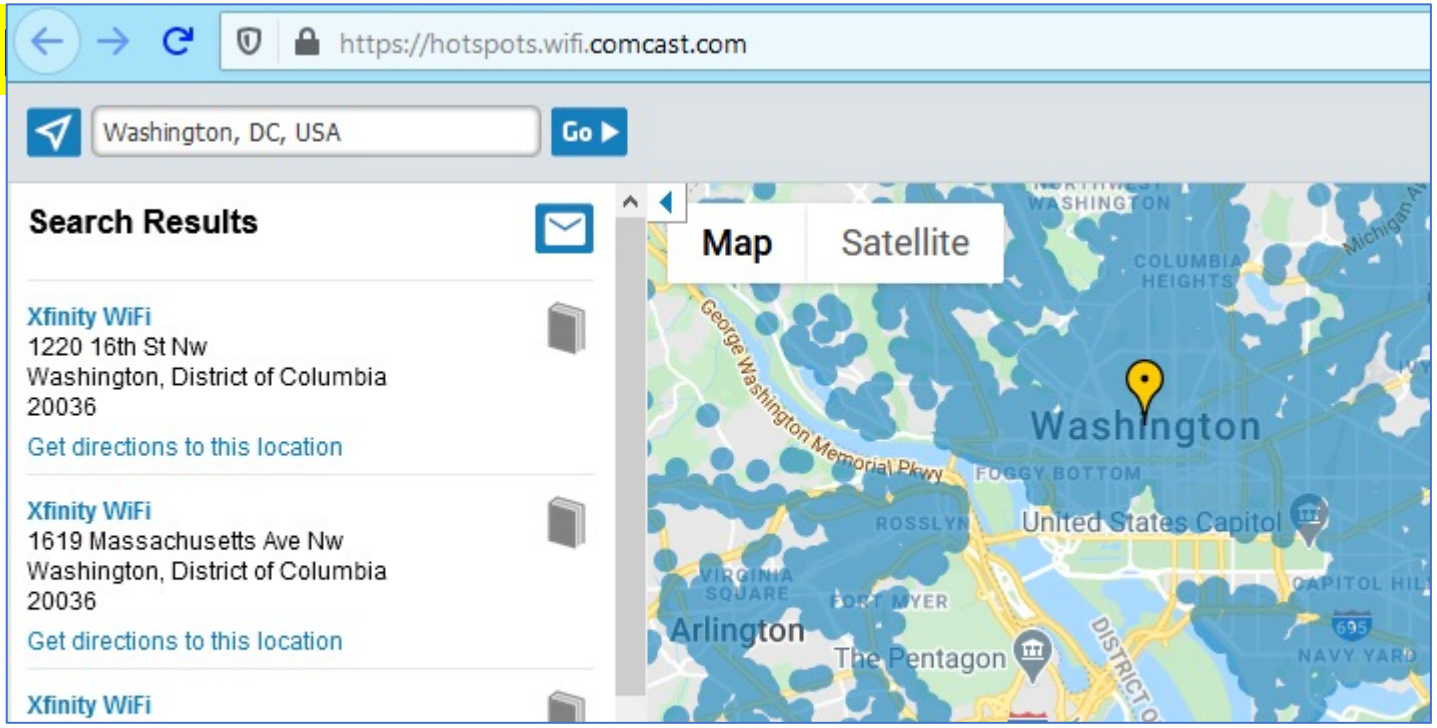
# Countermeasures for source IP addresses

- Never use the same IP address for two activities that you don't want associated with each other
- IP addresses should make sense relative to your actions from the IP
- Logging into user accounts:
  - One IP address per user account - and keep using the same IP for all subsequent logins
  - Login from an IP address in the same region as the user (common web mail alert)
  - Login from a service provider that makes sense for the target user. Probably not a VPS
  - Avoid known-suspicious IPs: TOR, proxies



# Where to get new source IP addresses?

- Paid services -- INSERT LUM
- Mobile hotspot
- Wifi hotspots (free or paid)
- VPNs (free or paid)
- Cloud service providers
- TOR




- Inspect each source IP for "tells" and build your own vetted list

# Make your VPN connection fail safe

- If your VPN connection drops unexpectedly, subsequent connections may come from your real IP address.
- Not ideal in the middle of suspicious activity (e.g. password spraying)

```
openvpn --script-security 2 --down vpn-down.sh --config <OPEN VPN FILE>
```

```
#!/bin/bash
echo 'Disabling network interfaces...'
systemctl stop network-manager
killall -9 dhclient
for i in $(ifconfig | grep -iEo '^[a-z0-9]+:' | grep -v '^lo:$' | cut -d ':' -f 1)
do
    ifconfig $i 0.0.0.0 down
done
```

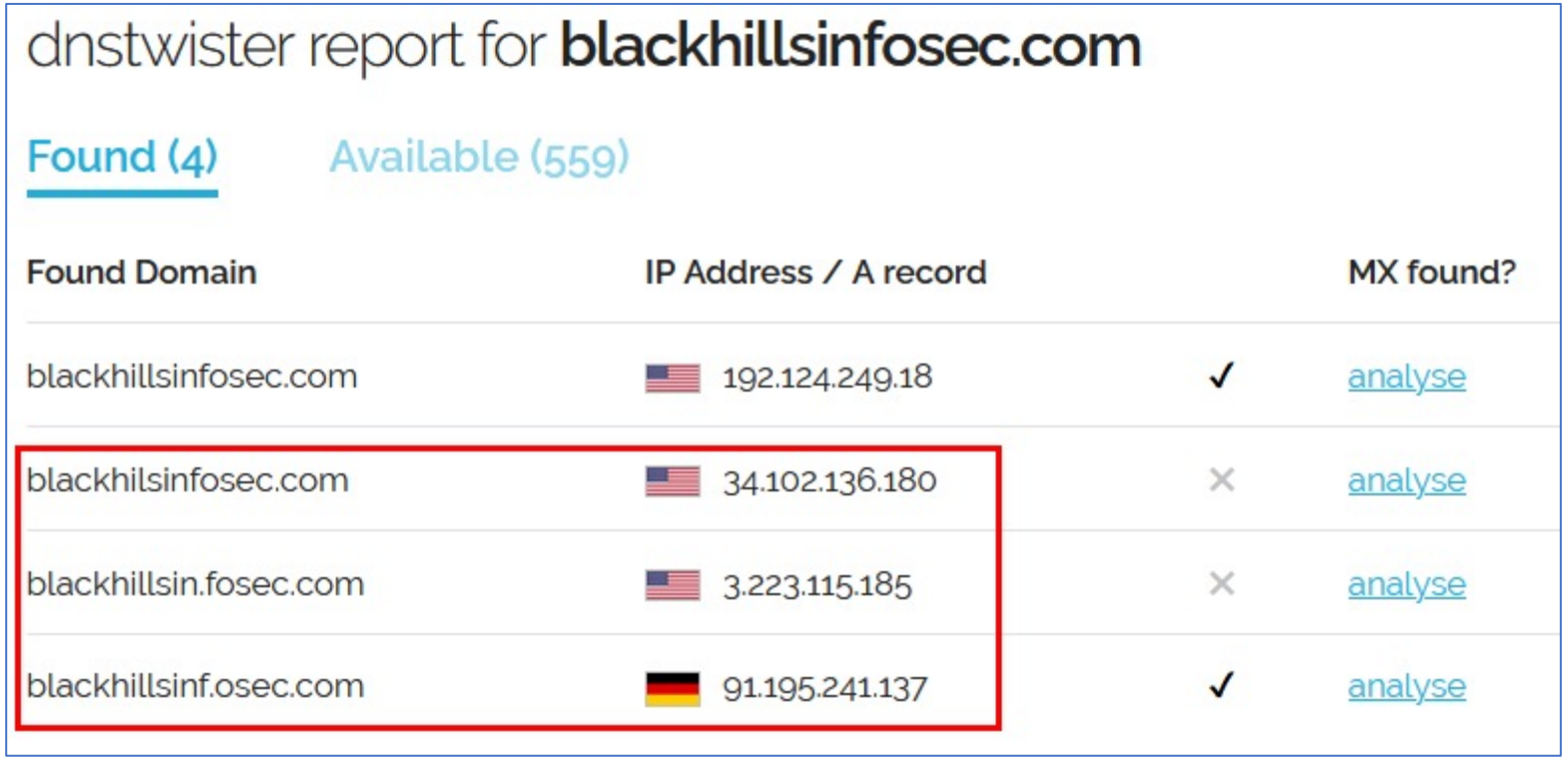
# 3. Other third-party services

# Third-party services

- Assess whether red team actions are likely to violate TOS
  - Modify identity, credit card number, and source IP for repeat sign-ups
  - "Identity": Name, email address, billing address, phone number
- Assess whether your registration information is likely to be exposed
  - Example: Domain registration/WHOIS info
- Assess whether use of the same account across multiple projects is likely to leak information about the red team or its customers
  - Can the resource owner be identified by outsiders?
  - Can relationships between multiple resources be associated with each other?
  - Is there a reasonable level of trust with the resource provider?

# Domain names

- Avoid typosquatting the target domain or company name
- Subdomains may be detected too



dnstwister report for **blackhillsinfosec.com**

Found (4) Available (559)

| Found Domain | IP Address / A record | | MX found? |
|---|---|---|---|
| blackhillsinfosec.com | 192.124.249.18 | ✓ | analyse |
| blackhilsinfosec.com | 34.102.136.180 | ✗ | analyse |
| blackhillsin.fosec.com | 3.223.115.185 | ✗ | analyse |
| blackhillsinf.osec.com | 91.195.241.137 | ✓ | analyse |

# Domain names

- Use private registration/WHOIS privacy
  - Make sure it is enabled at the time of registration
  - Consider fake registration information?
- One red team action per domain name - same as source IPs
  - Domain A sends email
  - Domain B serves payload files
  - Domain C receives C2 callbacks
  - …

# SSL/TLS certificates and CAs

- Watch out for data leaks in certificate transparency logs
  - Customer names (e.g. in subdomain names)
  - Email addresses

```
certbot --apache --register-unsafely-without-email
```

- No default or self-signed certificates - Both easily flagged/blocked
  - Censys.io search for Cobalt Strike: 87f2085c32b6a2cc709b365f55873e207a9caa10bffecf2fd16d3cf9d94d390c
- Let's Encrypt certificates *might* be suspicious
  - Default CA in lots of hacking tutorials
  - Free
  - Used by legitimate websites too

# 4. Payloads and network services

# Network services

- Don't expose services to the Internet that aren't required
  - Use SSH port forwarding for access from the red team
- Change all the default settings on hacking tools that listen on an Internet-facing port
- Use redirectors liberally and spread them across multiple CSPs
  - On the server: Whitelist redirector IP addresses and block all others
- Use common web services (Apache, Nginx) to redirect HTTP/S traffic to servers run by hacking tools

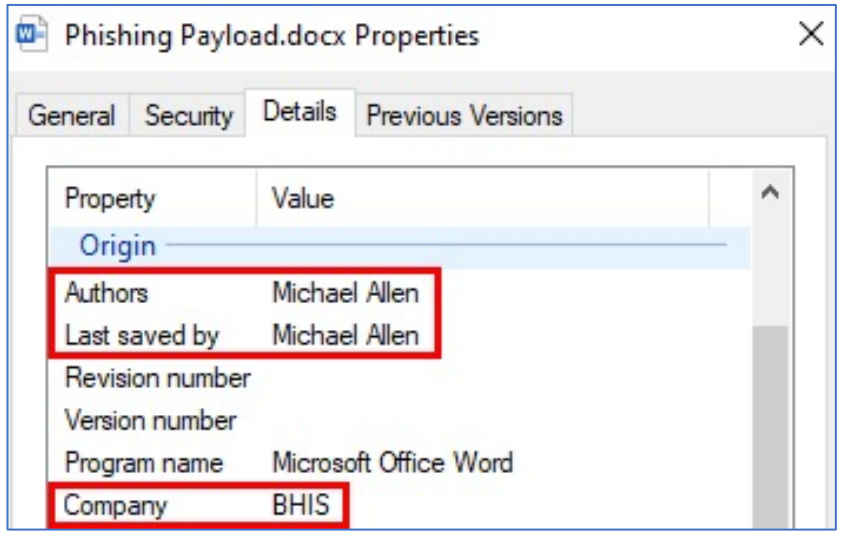
# Example: Cobalt Strike Team Server Study*

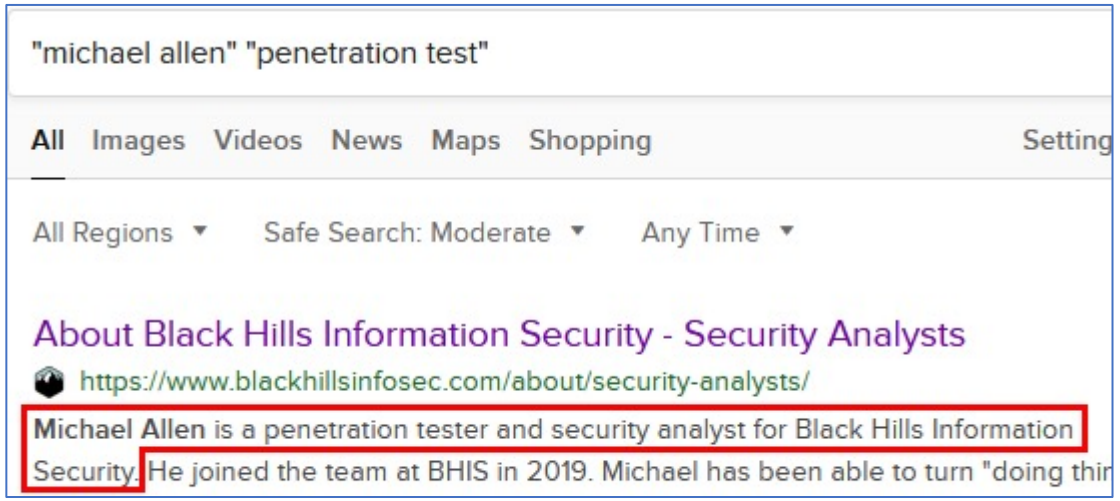
- **Indicator:** TCP Port 50050 - Default Cobalt Strike Team Server port
  - *Don't expose services to the Internet that aren't required*
  - *Use SSH port forwarding for access from the red team*
- **Indicator:** Cobalt Strike DNS server responds to requests with IP 0.0.0.0
  - *Change all the default settings on hacking tools that listen on an Internet-facing port*
- **Indicator:** Web root: 404 Not Found, no content, Content-Type: text/plain
  - Default response without content explicitly hosted at / or a web redirector.
- **Indicator:** JA3S service fingerprinting
  - *Use common web services (Apache, Nginx) to redirect HTTP/S traffic to servers run by hacking tools*

* https://blog.cobaltstrike.com/2019/02/19/cobalt-strike-team-server-population-study/

# Executable payloads - Metadata

- Commonly used by red teams to gather names, user names, and other details of the target organization
- Can also be used against the red team
- Common phishing payloads: MS Office documents, PDF, others...



# Executable payload testing

1. Make sure the payload works - execute in the "Attack" VM
2. Test keying and antivirus-bypass
   a. Make Test VM match the target environment (domain/host/user names)
   b. Install and update relevant defensive software on the Test VM - Online
   c. Disconnect the Test VM from the Internet
   d. Take a snapshot
   e. Execute the payload file to confirm keying and AV-bypass are working
3. Revert Test VM to the snapshot created in step "d", connect to the Internet, and perform final testing

# Payload hosting

- Filter incoming web requests by User-Agent
  - Redirect User-Agents: "bot", "google", "crawl", "search", "curl", "wget"
  - Stage 1 redirect with JavaScript - may help to weed out non-browser traffic
  - Stage 2 redirect - server-side scripts/mod_rewrite rules*
- Desktop browsers -> payload file download
- Mobile browsers -> credential harvesting page
- Redirect all others to content that supports the domain's categorization

* https://bluescreenofjeff.com/2016-04-05-operating-system-based-redirection-with-apache-mod_rewrite/

# 5. Testing new tools

# Vet the tool

- Read the source code
- Understand the configuration options
- Run the tool in a VM
- Observe the traffic the tool generates:
  - Packet capture - WireShark
  - Intercepting proxy - Burp Suite
  - Simulated target - Ncat/Netcat or target application/service

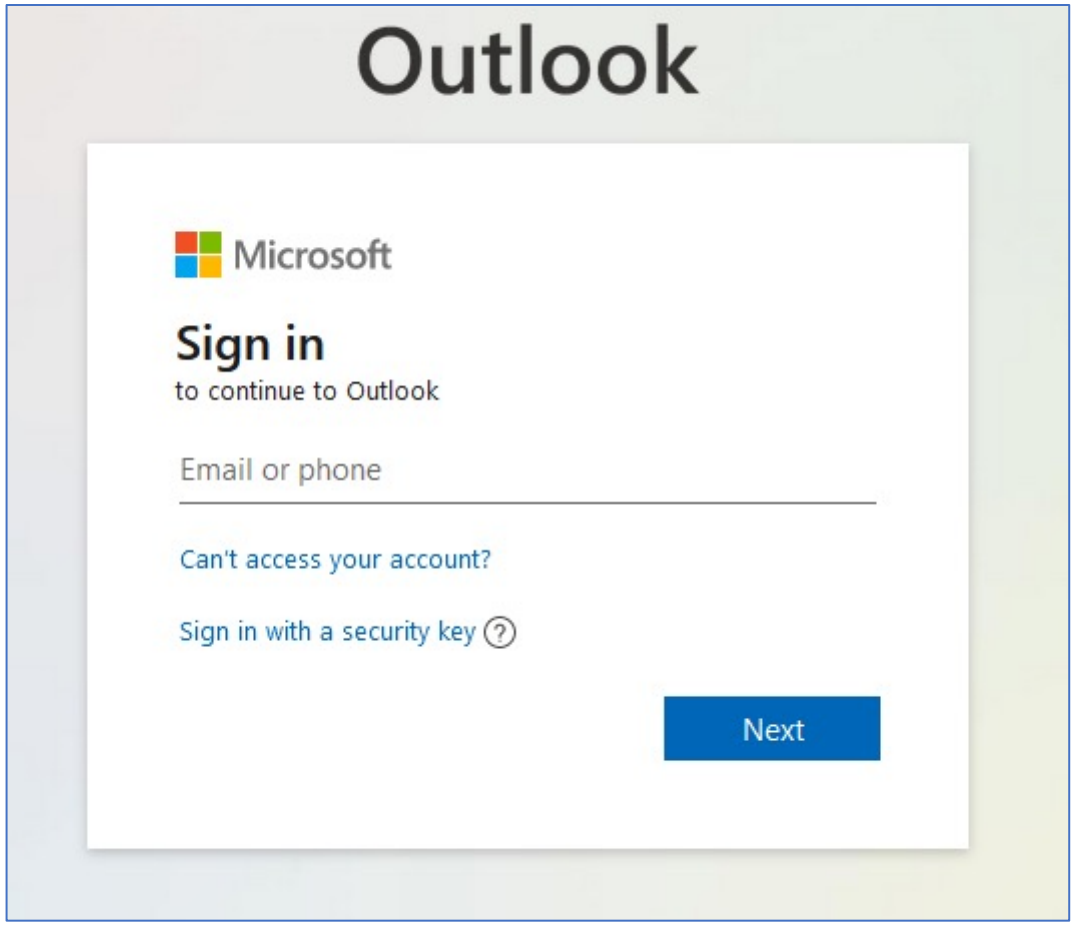
# Vet the tool - Goals:

1. Confirm that the tool isn't harming you.
2. Confirm that how the tool/attack looks from the defender perspective isn't suspicious.
3. Confirm that the attack looks believable from the perspective of the target user.

# Example: Evilginx

# Me, running EvilGinx for the first time:

1. Download the latest precompiled release from GitHub
2. Configure a phishlet to target Office 365
3. Generate a "lure" (landing URL)
4. Visit the lure in the browser
5. Everything looks ok -->



# Me, pasting the lure URL into Google Chat:

- Everything does **NOT** look ok

# Me, pasting the lure URL into Google Chat:

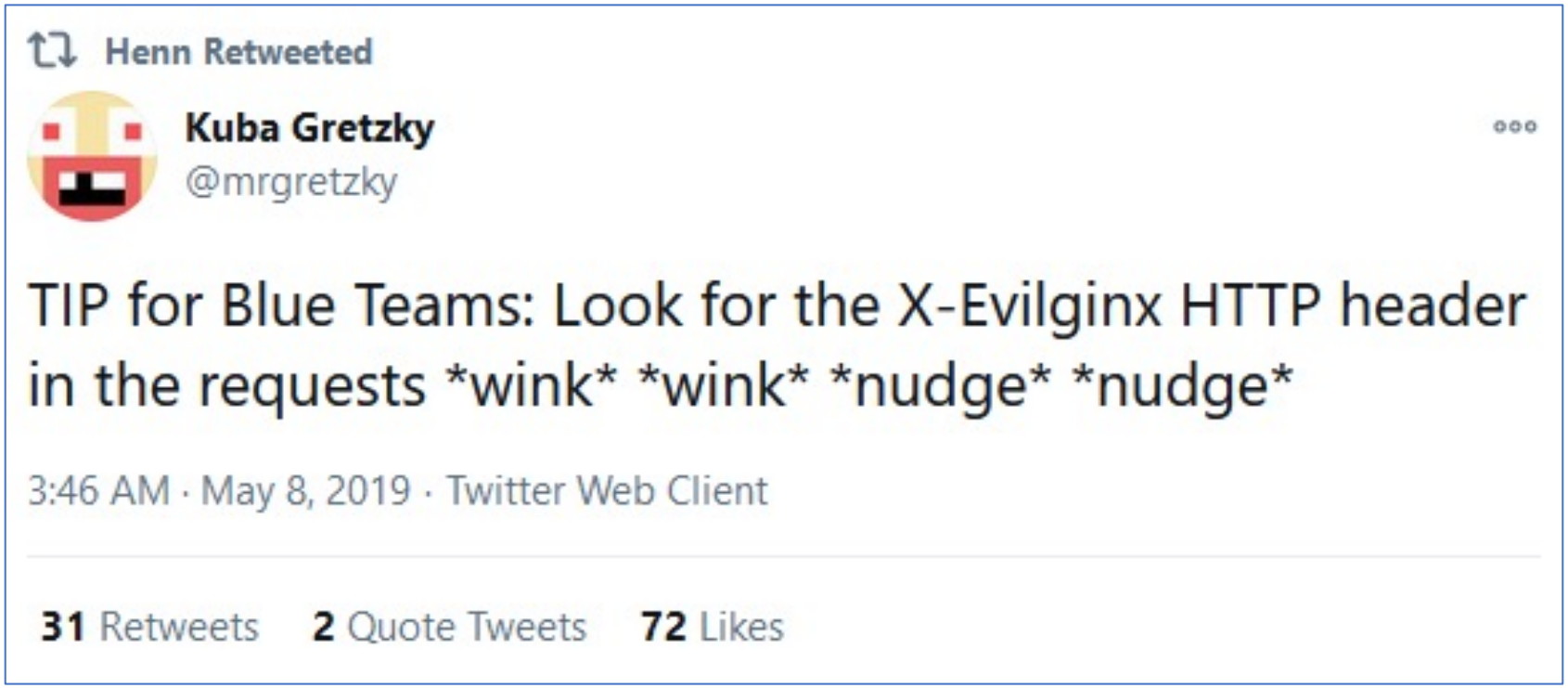
- Everything does **NOT** look ok


- Easy fix:

```
config redirect_url "https://www.office.com"
```

- Lesson learned:
  - **Read the source code**
  - **Understand the configuration options**

# But wait, there's more...



# Inspecting Evilginx traffic with Burp:

**Request**

Pretty | Raw | \n | Actions

```
1  GET / HTTP/1.1
2  Host: login.microsoftonline.com
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
   Gecko) Chrome/87.0.4280.141 Safari/537.36
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Cookie:
7  Upgrade-Insecure-Requests: 1
8  X-Evilginx: login.attacker-domain.com
9  Accept-Encoding: gzip, deflate
10 Connection: close
11
```

# Removing the X-Evilginx header

- No settings found to change/disable the header
- No mention in the documentation
- No references found when searching the source-code

```
unknown@*:~/evilginx2$ grep -Ri 'X-Evilginx' *
unknown@*:~/evilginx2$
```

# Looking more closelier…

```
unknown@*:~/evilginx2$ cat patched_lines.txt
http_proxy.go 0183 // egg2 := req.Host
http_proxy.go 0350 // hg := []byte{0x94, 0xE1, 0x89, 0xBA, 0xA5, 0xA0, 0xAB, 0xA5, 0xA2, 0xB4}
http_proxy.go 0377 // for n, b := range hg {
http_proxy.go 0378 // hg[n] = b ^ 0xCC
http_proxy.go 0379 // }
http_proxy.go 0381 // e_host := req.Host
http_proxy.go 0407 // req.Header.Set(string(hg), egg2)
http_proxy.go 0562 // e := []byte{208, 165, 205, 254, 225, 228, 239, 225, 230, 240}
http_proxy.go 0563 // for n, b := range e {
http_proxy.go 0564 // e[n] = b ^ 0x88
http_proxy.go 0565 // }
http_proxy.go 0566 // req.Header.Set(string(e), e_host)
http_proxy.go 0580 // p.cantFindMe(req, e_host)
http_proxy.go 1450 // func (p *HttpProxy) cantFindMe(req *http.Request, nothing_to_see_here string) {
http_proxy.go 1451 // var b []byte = []byte("\x1dh\x003,)\",+=")
http_proxy.go 1452 // for n, c := range b {
http_proxy.go 1453 // b[n] = c ^ 0x45
http_proxy.go 1454 // }
http_proxy.go 1455 // req.Header.Set(string(b), nothing_to_see_here)
http_proxy.go 1456 // }
```

X-Evilginx header gets added to the request 3 separate times

Line numbers. Relevant lines were spread throughout the file.

# Final result

1. Remove all source code lines that set the X-Evilginx header
2. Compile the modified source code
3. Inspect the outgoing traffic again
4. No more X-Evilginx header! :)

- Lesson learned: **Always inspect the network traffic**