secarma®
CYBERSECURITY EXPERTS

Hacking with Git

Recon
Post Exploitation
Infil (getting tools in)
Exfil (getting data out)
Reverse Shell

Thanks for Checking me out

- 14:00 – 17:00 - Embedded Firmware Exploitation by Aaron Guzman
- 14:45 - The Insider - Neil Lines
- 15:35 – CAAAAAAAKE!

What is Git?

- "Git is a ***version control system (VCS)*** for tracking changes in computer files and coordinating work on those files among multiple people."
  - https://en.wikipedia.org/wiki/Git
- It is open source and there are alternative version control systems:
  - Visual Studio Team Service, Subversion (SVN) etc
  - The techniques in this talk should be transferable to any allowing a distributed approach.
- It is by any measure the most "popular" VCS:
  - More than 20,000 questions on Stack Overflow
  - Around 87% of all the questions asked about the 5 most common VCS.
  - https://rhodecode.com/insights/version-control-systems-2016

HACKING with Git not “Git”

- VIDEO ON THIS SLIDE REDACTED. Included in full talk:
- https://www.youtube.com/watch?v=uolyWLeKqOc

Using git for Reconnaissance

secarma LABS

- **Reconnaissance**

- Occurs as the start of any good Pentest.
- Learn about your target as quietly as possible.
- Find information you can act on later.

- **Topics Coming Up:**

- Git Explorers [Pre-Existing]
- Git Scrapers [Pre-Existing]
- Using Git to Fingerprint [New Technique]

Git Explorers 1 [Pre-Existing]

- A form of ***passive*** reconnaissance. [No requests sent to target site]
- If you ***have*** access to a git repository simply "clone" it down.
- Use an "*explorer*" to search file contents and commit history for sensitive information.
- <u>IMPACT</u>:
  - A loss of confidentiality of sensitive information.
  - Potential for onward exploitation if the information is good enough.

Git Explorers 2 – A vulnerable Repo

- A vulnerable repo with some sensitive info

Branch: master ▾ sensitiveInfo / README.md

gitshells Added setup details

1 contributor

7 lines (5 sloc) | 136 Bytes

sensitiveInfo

A dummy repository which ends up with some sensitive data in it

Credentials

Username=admin Password=hasleakedviaGit

Git Explorers 3 – Existing Tools

- A plethora of tools available for this:
  - https://github.com/kootenpv/gittyleaks
    - Regexes for plaintext passwords and sensitive info in files.
  - https://github.com/dxa4481/truffleHog
    - Checks commit history for every branch.
    - Focused on API keys: AWS, SLACK, as well as PGP private keys etc.
  - https://github.com/michenriksen/gitrob
    - Very powerful. Needs a decent amount of setup to work but will find you lots of things.

Git Explorers 4 – Example: GittyLeaks

- **GittyLeaks**
  - Easy install with pip.
  - Easy to use

```
root@kali:~/Desktop/hacking-with-git# pip install gittyleaks
Collecting gittyleaks
Requirement already satisfied: scandir in /usr/local/lib/python2.7/dist-packages (from gittyleaks)
Requirement already satisfied: sh in /usr/local/lib/python2.7/dist-packages (from gittyleaks)
Installing collected packages: gittyleaks
Successfully installed gittyleaks-0.0.23
root@kali:~/Desktop/hacking-with-git# gittyleaks -user gitshells -repo sensitiveinfo
-----------------------------------------------------------------------------
                    gittyleaks' Bot Detective at work ...
-----------------------------------------------------------------------------
----------------------------------------
README.md: Username=admin
README.md: Password=hasleakedviaGit
root@kali:~/Desktop/hacking-with-git#
```

Secarma

@Secarma

@SecarmaLabs

CYBERSECURITY EXPERTS

Git Explorers: Defending Part 1

secarma LABS

- Prevent sensitive information being committed in the first place.
- Keep sensitive data inside configuration files i.e. “config.txt”
- Use “gitignore” to ensure that files (https://git-scm.com/docs/gitignore)
- Be mindful of the caveat below:

DESCRIPTION

A `gitignore` file specifies intentionally untracked files that Git should ignore. Files already tracked by Git are not affected; see the NOTES below for details.

...

NOTES

The purpose of gitignore files is to ensure that certain files not tracked by Git remain untracked.

To stop tracking a file that is currently tracked, use *git rm --cached*.

Git Explorers: Defending Part 2

- Introduce checks to your commit and pull requests:
- https://github.com/ezekg/git-hound
  - Plugin for git
  - Checks for sensitive data before committing it
  - Using similar regex approach to the explorers themselves.
- https://github.com/auth0/repo-supervisor
  - Scans and checks at Pull Requests.

Git Scrapers 1 [Pre-Existing]

secarma LABS

- A form of ***active*** reconnaissance. [Some requests to target site]
- Target site allows access to "/.git" within the web root.
- As git maintains an index of files it versions:
  - Attacker get's directory listings again like it is 1999 folks!

- **IMPACT**:

"A directory listing provides an attacker with the complete index of all the resources located inside of the directory. The specific risks and consequences vary depending on which files are listed and accessible."

https://cwe.mitre.org/data/definitions/548.html

Git Scrapers 2 – Vulnerable Repository

- A folder you won't likely guess by brute-force with sensitive content:

gitshells / sensitiveInfo

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights

sensitiveInfo / nevergonnagiveyouup / letyoudown / privatekey.txt

Edit new file | Preview

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe4eCZ0FPqri0cb2JZfXJ/DgYSF6vUp
wmJG8wVQZKjeGcjDOL5UlsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5
1s1SPrCBkedbNf0Tp0GbMJDyR4e9T04ZZwIDAQABAoGAFijko56+qGyN8M0RVyaRAXz++xTqHBLh
3tx4VgMtrQ+WEgCjhoTwo23KMBAuJGSYnRmoBZM3lMfTKevIkAidPExvYCdm5dYq3XToLkkLv5L2
pIIVOFMDG+KESnAFV7l2c+cnzRMW0+b6f8mR1CJzZuxVLL6Q02fvLi55/mbSYxECQQDeAw6fiIQX
GukBI4eMZZt4nscy2o12KyYner3VpoeE+Np2q+Z3pvAMd/aNzQ/W9WaI+NRfcxUJrmfPwIGm63il
AkEAxCL5HQb2bQr4ByorcMWm/hEP2MZzROV73yF41hPsRC9m66KrheO9HPTJuo3/9s5p+sqGxOlF
L0NDt4SkosjgGwJAFklyR1uZ/wPJjj611cdBcztlPdqoxssQGnh85BzCj/u3WqBpE2vjvyyvyI5k
X6zk7S0ljKtt2jny2+00VsBerQJBAJGC1Mg5Oydo5NwD6BiROrPxGo2bpTbu/fhrT8ebHkTz2epl
U9VQQSQzY1oZMVX8i1m5WUTLPz2yLJIBQVdXqhMCQBGoiuSoSjafUhV7i1cEGpb88h5NBYZzWXGZ
37sJ5QsW+sJyoNde3xH8vdXhzU7eT82D6X/scw9RZz+/6rCJ4p0=
-----END RSA PRIVATE KEY-----
```

Git Scrapers 3 – Making a Vulnerable Site

- Creating a vulnerable target. Disable directory listings in Apache for more realism:

```
GNU nano 2.9.4                    /etc/apache2/apache2.conf                    Modified

        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

#<Directory /srv/>
#       Options Indexes FollowSymLinks
#       AllowOverride None

^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo
^X Exit       ^R Read File   ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line M-E Redo
```

Indexes enables directory listings
Delete that to stop listings.

Secarma
@Secarma
@SecarmaLabs

Git Scrapers 4 – Making a Vulnerable Site 2

- Cloning a git into the web root.

```
root@kali:/var/www/html# git clone https://github.com/gitshells/sensitiveInfo.git
Cloning into 'sensitiveInfo'...
remote: Counting objects: 11, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 11 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (11/11), done.
root@kali:/var/www/html# ls
index.html  sensitiveInfo
root@kali:/var/www/html# mv sensitiveInfo/ admin
root@kali:/var/www/html# ls admin/.git
branches  config  description  HEAD  hooks  index  info  logs  objects  packed-refs  refs
root@kali:/var/www/html# service apache2 start
root@kali:/var/www/html#
```

Secarma

@Secarma

@SecarmaLabs

Cloning a git repo into a web servable folder

Making the folder easier to find

Starting Apache

secarma CYBERSECURITY EXPERTS

Git Scrapers 5 – Demonstration 1

- Directory brute force using “dirb” in default mode would locate “/admin” (not shown) and then “/admin/.git”

```
root@kali:~/Desktop/hacking-with-git# dirb http://localhost/admin

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Wed Apr  4 08:59:20 2018
URL_BASE: http://localhost/admin/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://localhost/admin/ ----
+ http://localhost/admin/.git/HEAD (CODE:200|SIZE:23)

-----------------
END_TIME: Wed Apr  4 08:59:21 2018
DOWNLOADED: 4612 - FOUND: 1
root@kali:~/Desktop/hacking-with-git#
```

Secarma
@Secarma
@SecarmaLabs

Git Scrapers 6 – Demonstration 2

- Many tools exist for this but I use “gin”.
  - https://github.com/chalstrick/gin
- Download the “.git/index” file from the target, then gin, grep & cut

```
root@kali:~/Desktop/hacking-with-git# wget http://localhost/admin/.git/index
--2018-04-04 09:53:39--  http://localhost/admin/.git/index
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 328
Saving to: ‘index’

index                       100%[===========================================>]     328  --.-KB/s    in 0s

2018-04-04 09:53:39 (72.3 MB/s) - ‘index’ saved [328/328]

root@kali:~/Desktop/hacking-with-git# gin index  | grep name | cut -d " " -f 5
README.md
nevergonnagiveyouup/letyoudown/privatekey.txt
root@kali:~/Desktop/hacking-with-git#
```

Download index with wget
Use gin and basic cut/grep to get directory listing

Git Scrapers - Defending part 1

- Be aware of what is in your web root.
- Check if you have a problem with “find”:

```
Syntax: find /path/to/web/root -name ".git"

Example: find /var/www/ -name ".git"
```

- IF you have .git folders check logs to see if anyone has requested it:

```
Syntax: grep ".git/index" /path/to/access.log

Example: grep ".git/index" /var/log/apache2/access.log
```

Git Scrapers – Defending Part 2

- Block access to ".git" folders within your web root.
- Multiple ways to do this.
- I tested enabling "*AllowOverride All*" for "*/var/www/*" to allow a "*.htaccess*" file to control access.

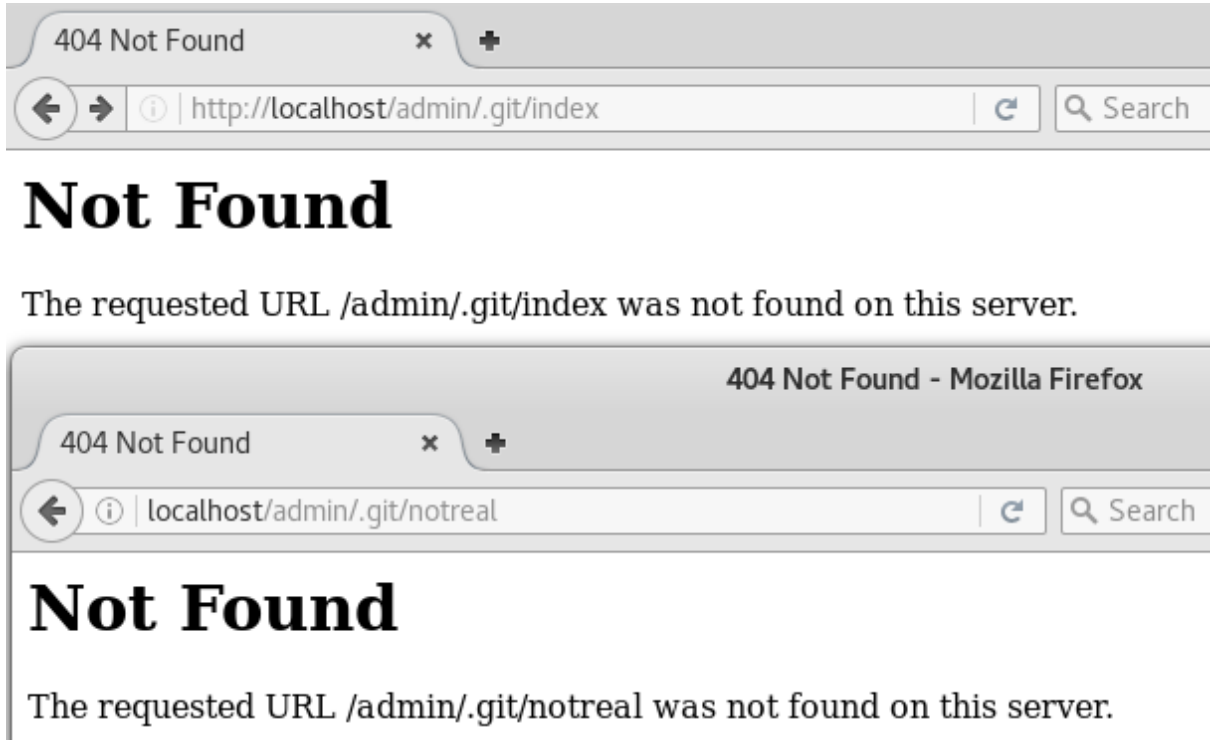
```
GNU nano 2.9.5                                  /etc/apache2/apache2.conf

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/>
        Options FollowSymLinks
        AllowOverride All
        Require all granted
</Directory>
```

- Create a ".htaccess" file in the web root containing:

```
RedirectMatch 404 /\.git
```

Git Scrapers – Testing Defences

- Succinct solution which hides the “.git” folder in a 404 response.



Git Fingerprint 1 – [New Technique]

secarma LABS

- **Problem**:
  - Your want to enumerate the version of an application you are targeting.
  - Pentesters do this to enable research of known security issues.
- **Solution**:
  - When the target is powered by an application using version controlled code.
  - Clone repository down.
  - Generate word list including all files in that repository.
  - Attempt to download those files from the target site.
  - Determine the specific commit for a bunch of files and then guess the version time!
  - Works for files which are not altered by download process like: .js, .css, .jpg etc

Git Fingerprint 2 – Demo Video!

- VIDEO ON THIS SLIDE REDACTED. Included in full talk:
- https://www.youtube.com/watch?v=uolyWLeKqOc

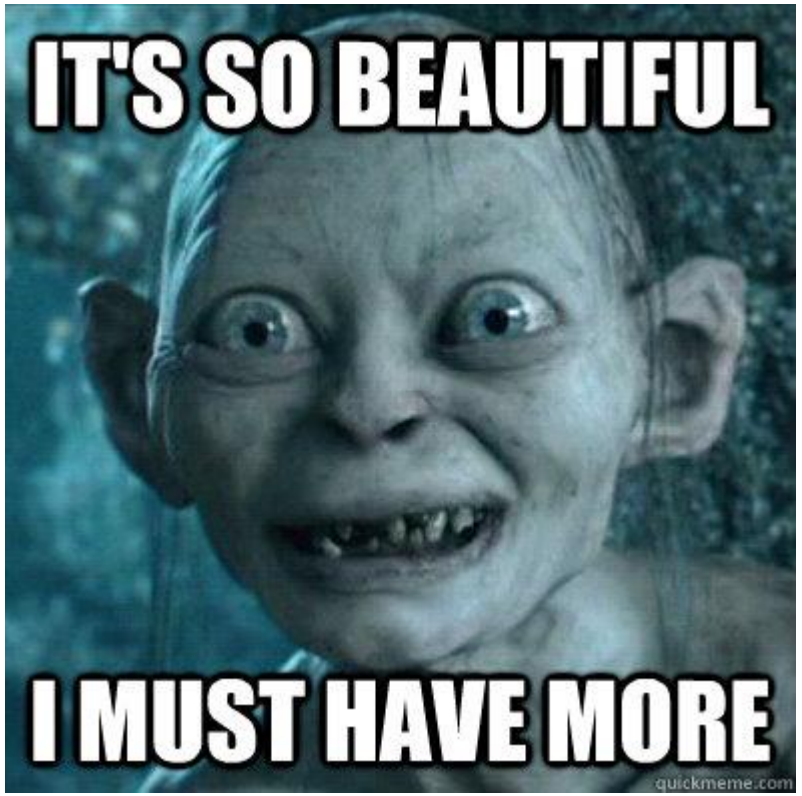
Git Fingerprint 3 – Next Steps

secarma LABS

- Check repo for the changelog, or version string around the date you found.
  - Bang you find your version!
  - The target will suffer from any vulnerabilities in newer commit messages.
- In the future Git-Fingerprint will support you better by:
  - Using “git log <file>” to review newer commit messages.
  - If you are lucky your target uses CVEs or some custom index for vulnerabilities.
- <u>To Recap</u>
  - Using git in this manner is a new technique for penetration testers to enjoy.
  - It will work wherever a target site uses version controlled code you can download.
  - The PoC is in Git but is transferable to other versioning software (CVS etc).

Using git for Post Exploitation

- **Post Exploitation**
- Occurs after you have compromised a target (and have a shell)
- Gather more information about the target and adjacent network.
- Seek to steal high value information.
- Search for lateral movement opportunities.



What “git” gives us for Post Exploitation?

secarma LABS

- Authentication options
  - **Plain-text passwords** – Home dir /.git-credentials
  - **Authentication Tokens** – Home dir /.git-credentials
  - **SSH Public Key** – Home dir /.ssh/id_rsa.pub
- Remote Repository Locations
  - Internal git repository servers
  - Possibly “private” repos on public servers such as “github”
- Source code to analyse for vulnerabilities
- Riding privileges of our compromised device poison a repo with malware.
- Possibly many more routes.

git_enum [New Metasploit Module]

- Checks for and dumps git authentication options in user “home” folders
- Find any “.git” folders on the target
  - Then prints the “remote” URL from the “.git/config” file

```
msf post(multi/gather/git_enum) > exploit

[*] Finding user .gitconfig file and looting them
[+] /root/.gitconfig - root - root@localhost.localdomain
[+] Downloaded -> /root/.msf4/loot/20180422234808_default_172.16.
[*] Finding user .git-credentials file and looting them
[+] https://[redacted]:@github.com
[+] Downloaded -> /root/.msf4/loot/20180422234821_default_172.16.
[*] Finding user .ssh/id_rsa.pub file and looting them
[-] No users found with a .ssh/id_rsa.pub file
[*] Finding all .git folders
[*] Found 43 .git folders to explore
```

Rogue Employee Scenario: Infil/Exfil

- Customers: ”what can a standard user do with their privileges?”
- Meaning in around 2012-2013 I first used GitHub.com to bypass corporate filtering.
- Target was behind 2 VPNs
- No DNS/ICMP/TCP/UDP routes for covert comms.
- Proxy was restrictive, but permitted github.com to enable developers.
- <u>IMPACT</u>
  - It is a risk giving employees any Internet connection. Risk Accepted!

Infiltration – Getting tools in part 1

- An attacker can make a repo like this
- Including whatever tooling they fancy.
- Hiding your tools somewhat:
  - Base64 encode binaries
  - Save it as a .txt in repo
  - Use “certutil” on Windows to decode

gitshells Update README.md

binaries	Delete a
wordlists	Delete a
README.md	Update README.md

README.md

infiltrating

Demo repository for infiltrating

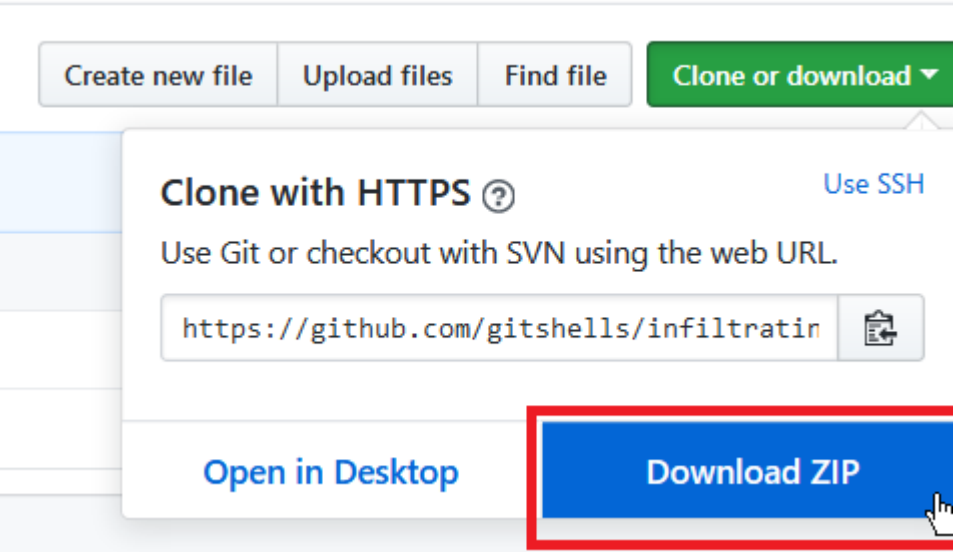
Contents

/wordlists - common usernames, and common passwords

/binaries - common tools, psexec, gsecdump etc

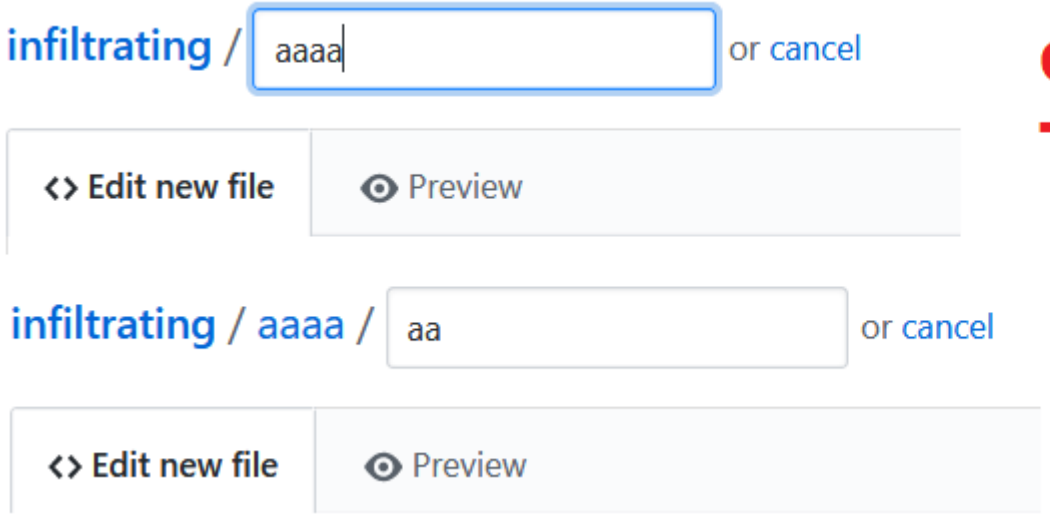
Inflitration – Getting tools in part 2

- Use “download ZIP” option
- No dependency on git being installed on the workstation anymore!



Exfiltration – Getting data out

- How to create a new folder via GitWeb.
- Browser text editor for copy/paste data.
- Or file upload options to add files



Create new file

Type a path

/ creates a new dir

Reverse Shell over Git?

- If you have a route IN and OUT it would be rude not to make a shell work!
- Demo Video Time.

Reverse Shell – Demo Video

- VIDEO ON THIS SLIDE REDACTED. Included in full talk:
- https://www.youtube.com/watch?v=uolyWLeKqOc

Roundup

- Recon
  - **Git Explorers** – Find sensitive info in a repository you can download
  - **Git Scrapers** – Scrape Directory Listings from a website with “.git” in web root
  - **Git Fingerprinting** – Gather information to enable a reasonable version guess.
- Post
  - Authentication harvesting
  - Enumerating git Remotes
- Infil/Exfil – Through corporate proxies
  - Using only a web browser
- Reverse Shell
  - Using a public repository as the communication channel.

Questions?

- Thanks for having me.