

Adversary Village @ DEFCON 32



AN ARTIST GUIDE TO ADVERSARY FORGERY

COPY CAT



- Former Artist
- Military Intelligence Veteran
- Red Teamer, Threat Hunter @Target
- Lead macOS & Linux ATT&CK

@MITRE

WHAT ARE WE DOING TODAY?

Read through a report and reproduce what the
MITRE ATT&CK technique the report is describing

ADVERSARY EMULATION IS...



ATT&CKing Pandas: Drawing out ATT&CK in the Wild

Cast of Characters

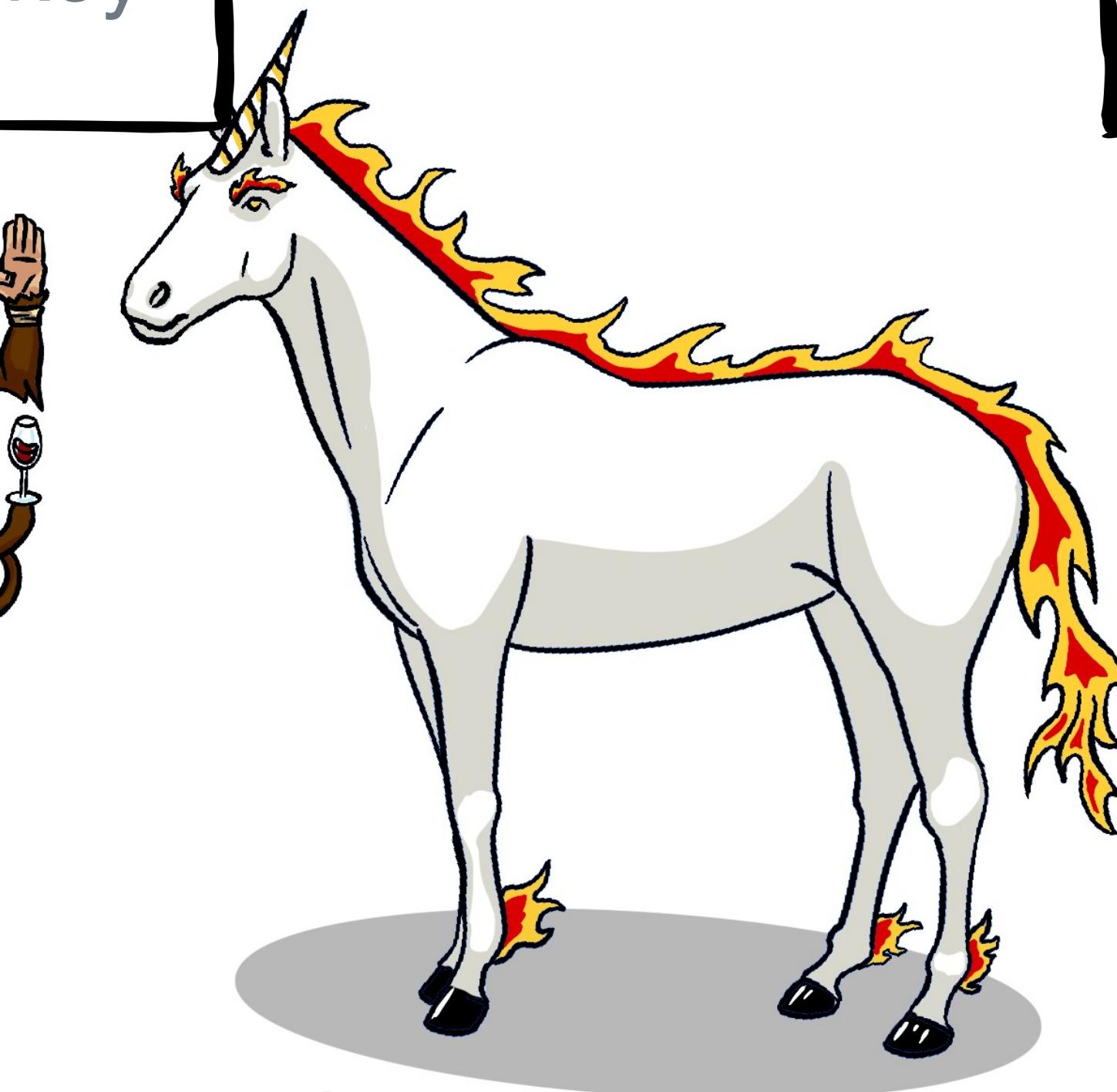
Ninja Panda &
threads



Drunken Monkey
Master



Kung Fu Master



A Pure Unicorn App

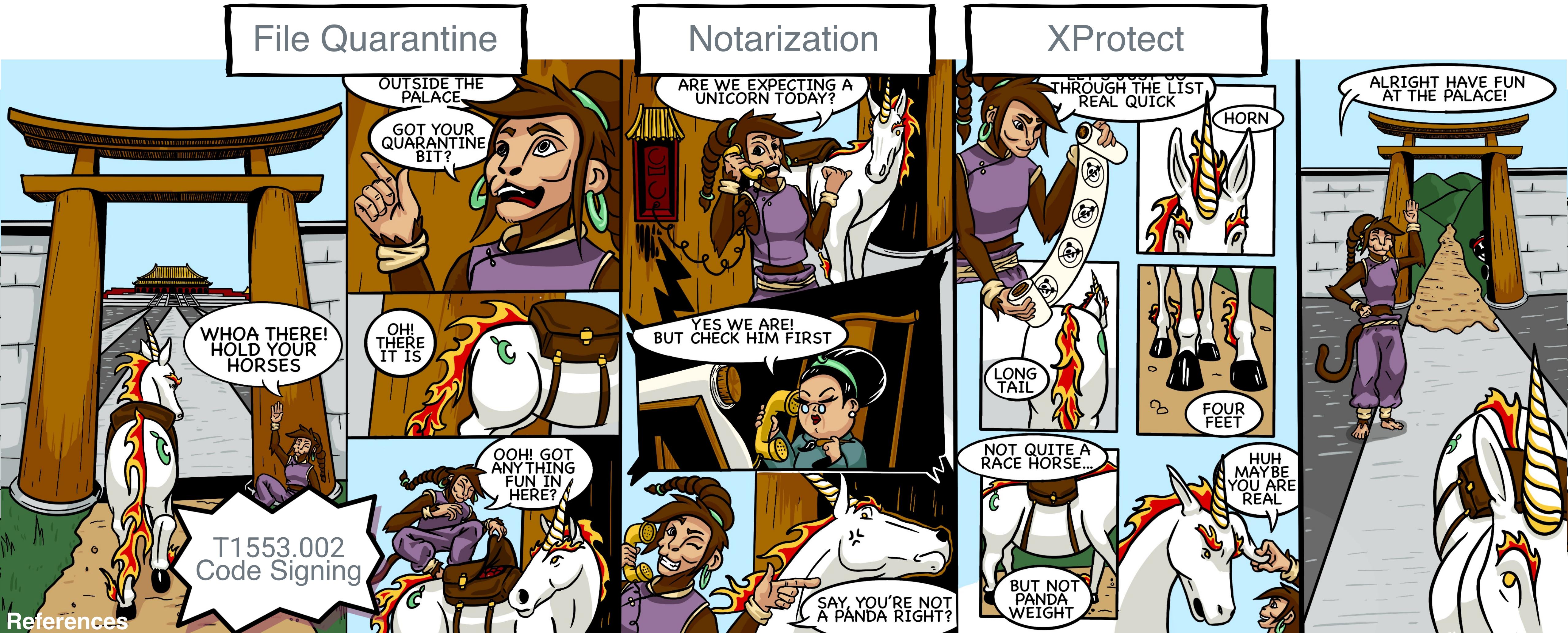
Helper Daemons



Artwork powered by
@MiscreantsHQ

@coolestcatiknow

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY



[Quarantine process, data locations](#)

[Code Signing process, data locations](#)

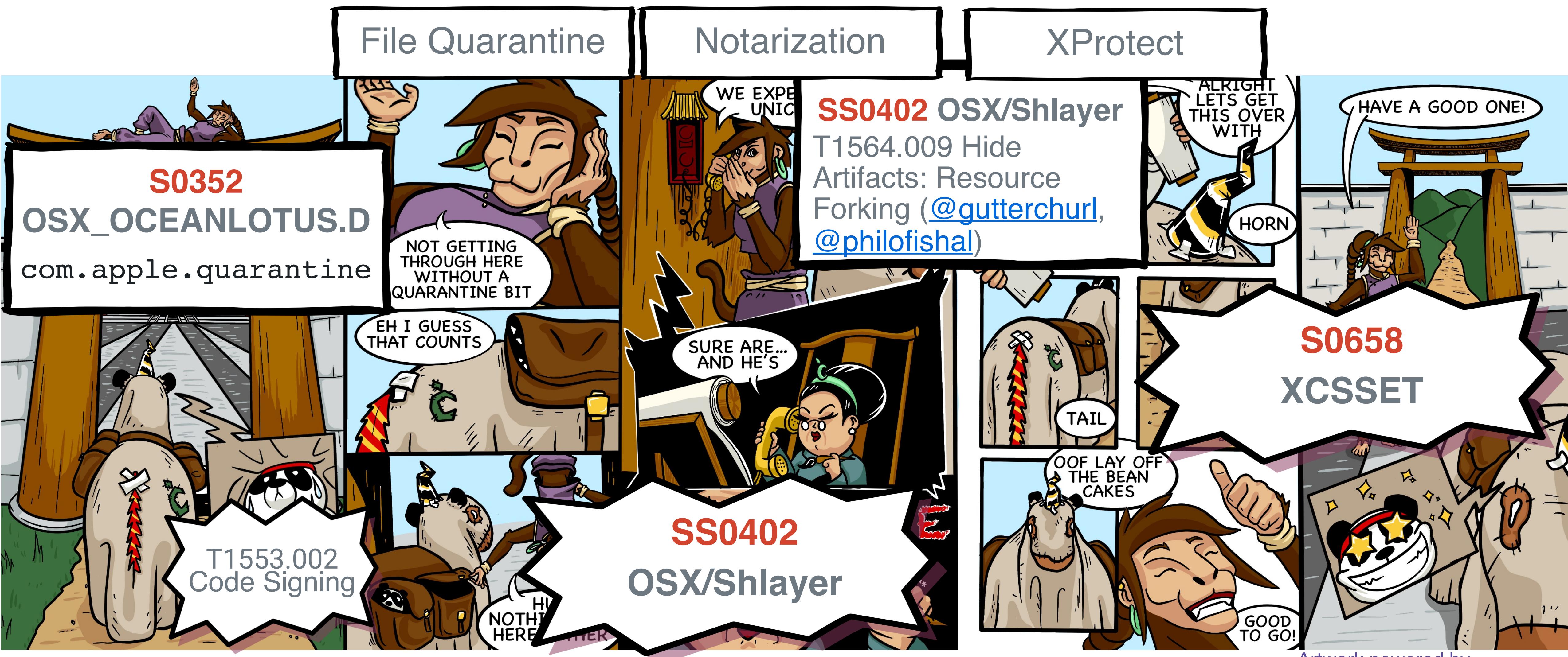
[Notarization process, data locations](#)

[Xprotect process, data locations](#)

Artwork powered by
@MiscreantsHQ

@coolestcatiknow

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY



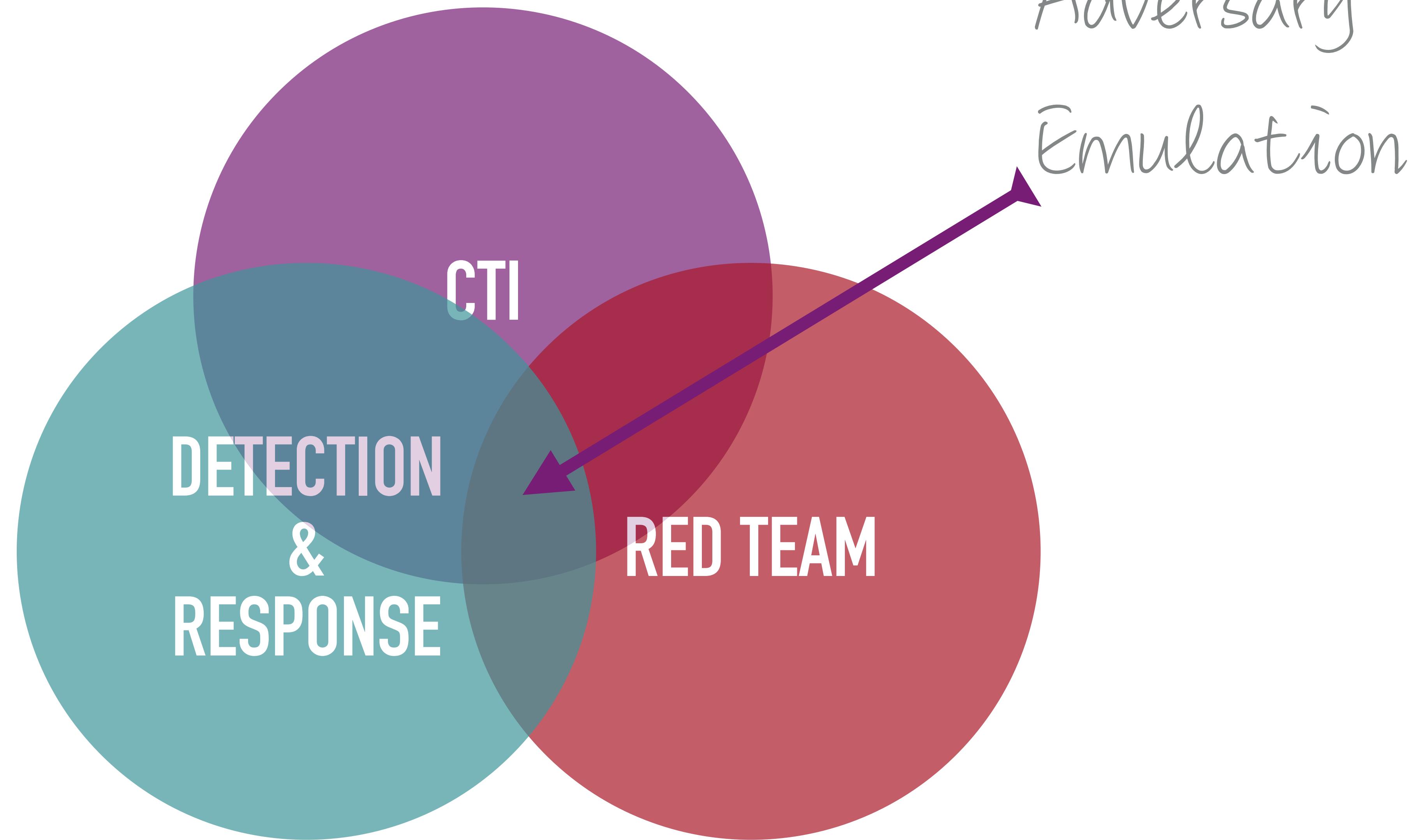
Artwork powered by

WHAT'S INVOLVED?

People

Process

Technology



<https://www.ibm.com/downloads/cas/QEBYPND1>

DIFFERENT WAYS TO USE AE

Atomic Testing	Micro Emulation	Full Emulation
Emulate single technique	Emulate compound behaviors across 2–3 techniques	Emulate adversary operation
 Executable in seconds	 Executable in seconds	 Executable in hours
<i>E.g., Atomic Red test for T1003.001 - LSASS Memory</i>	<i>E.g., Fork & Run Process Injection</i>	<i>E.g., FIN6 adversary emulation plan</i>
 Easy to automate	 Easy to automate	 Easy to automate
 Validate atomic analytics	 Validate atomic analytics	 Validate atomic analytics
 Validate chain analytics	 Validate chain analytics	 Validate chain analytics
 Evaluate SOC against a specific set of TTPs	 Evaluate SOC against a specific set of TTPs	 Evaluate SOC against a specific set of TTPs
 Evaluate SOC holistically against specific groups	 Evaluate SOC holistically against specific groups	 Evaluate SOC holistically against specific groups

Skoog, Ingrid. "Ahhh, This Emulation Is Just Right: Introducing Micro Emulation Plans." *Medium*, MITRE-Engenuity, 15 Sept. 2022, medium.com/mitre-engenuity/ahhh-this-emulation-is-just-right-introducing-micro-emulation-plans-7bf4c26451d3.

attack.mitre.org/groups/G0050/

MITRE | ATT&CK®

Matrices ▾ Tactics ▾ Techniques ▾ Defenses ▾ CTI ▾ Resources ▾ Benefactors

Blog ↗ Search 🔎

APT32

APT32 is a suspected Vietnam-based threat group. The group has targeted multiple private sector organizations and journalists with a strong focus on Southeast Asia, Laos, and Cambodia. They have extensive espionage against victims.^{[1][2][3]}

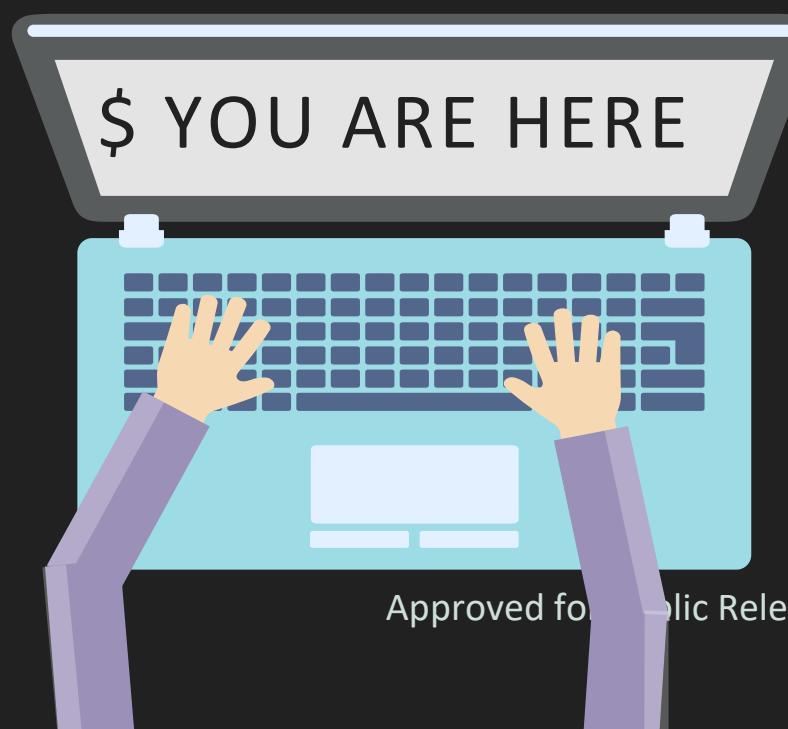
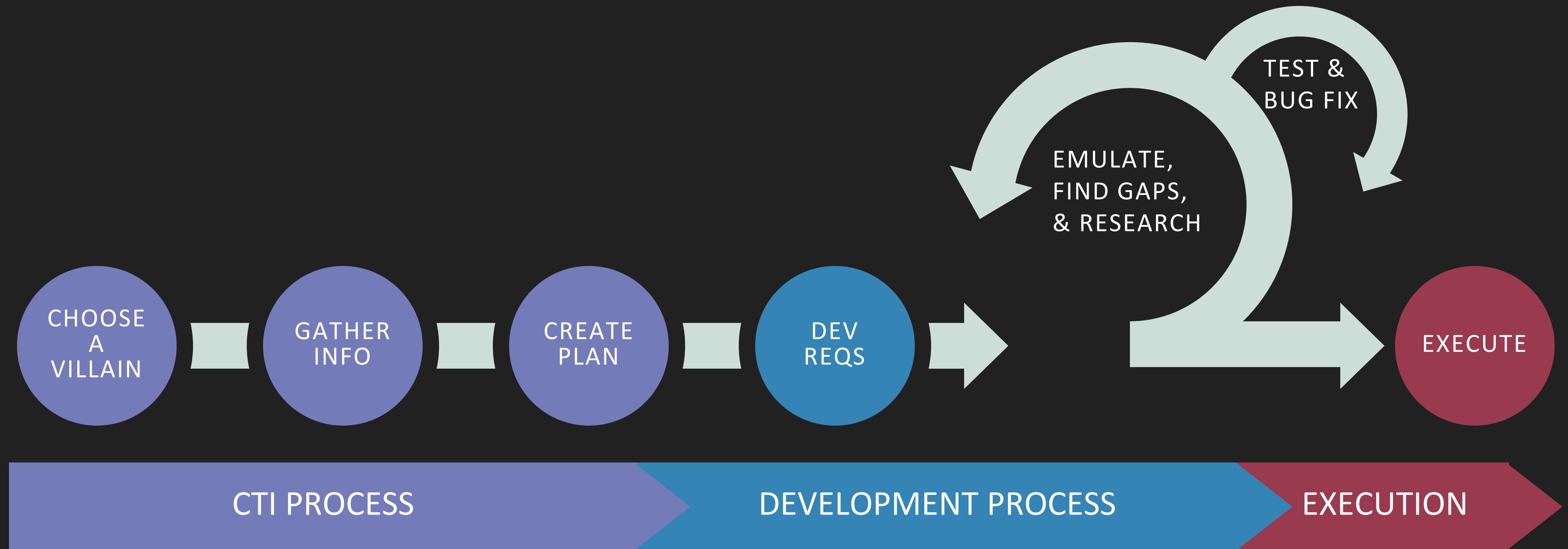
MITRE | ATT&CK®

APT32

ID: G0050
Associated Groups: SeaLotus, OceanLotus, APT-C-00, Canvas Cyclone, BISMUTH
Contributors: Romain Dumont, ESET
Version: 3.0
Created: 14 December 2017
Last Modified: 17 April 2024

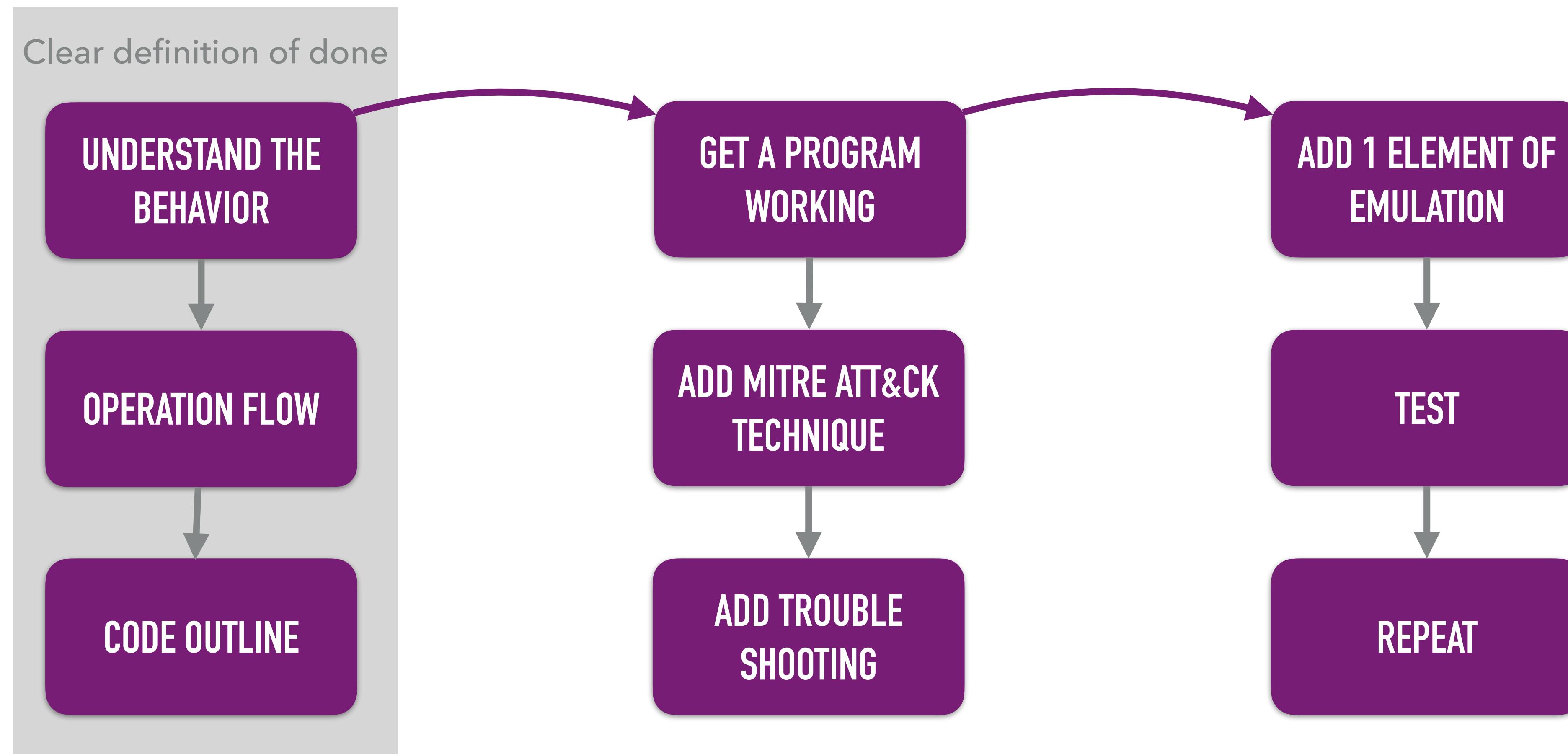
[Version Permalink](#)

BECOMING A DARK KNIGHT



<https://www.blackhat.com/us-23/briefings/schedule/#becoming-a-dark-knight-adversary-emulation-demonstration-for-attck-evaluations-33209>

PROCESS BREAKDOWN



EXERCISE: FIND WHERE THE MITRE ATT&CK TECHNIQUE SHARED MODULES T1129 IS DESCRIBED IN THE REPORTS.

- ▶ Review the [MITRE ATT&CK technique T1129](#)'s procedures. Look for `dlopen` & `dlsym` function calls to narrow it down.
- ▶ In reporting, look for words used to describe T1129 Shared Modules...*plugins, shared libs, dynamic libraries, library, & load*. Look for these words to isolate where the behavior is described in the reporting. *Mappings at the bottom of the report != 100% correct.*

Reports

1. https://www.trendmicro.com/en_us/research/20/k/new-macos-backdoor-connected-to-oceanlotus-surfaces.html
2. <https://unit42.paloaltonetworks.com/unit42-new-improved-macos-backdoor-oceanlotus/>
3. https://www.trendmicro.com/en_us/research/18/d/new-macos-backdoor-linked-to-oceanlotus-found.html
4. <https://www.welivesecurity.com/2019/04/09/oceanlotus-macos-malware-update/>

EVALUATING CTI REPORTS

EXPLICIT: "THE GOOD"

- ▶ Code/scripts
- ▶ C2 communication analysis
- ▶ Other artifacts (file paths, registry keys, etc.)

IMPLICIT: "THE GREAT"

- ▶ Lateral Movement
- ▶ Adversary actions on objectives
- ▶ Environment details

attack.mitre.org/techniques/T1129/

MITRE | ATT&CK®

Matrices ▾ Tactics ▾ Techniques ▾ Defenses ▾ CTI ▾ Resources ▾ Benefactors

Blog ↗ Search

Shared Modules

Shared Modules

Adversaries may execute malicious payloads via loading shared modules.

Shared modules are executable files that are loaded into processes to provide access to reusable code, such as specific custom functions or invoking OS API functions (i.e., Native API).

macOS can execute `.so` files, common practice uses `.dylib` files.^{[1][2][3][4]}

Version Permalink

The screenshot shows the MITRE ATT&CK website interface. At the top, there's a navigation bar with links for Matrices, Tactics, Techniques, Defenses, CTI, Resources, and Benefactors. Below the navigation is a search bar labeled "Search". On the left, the MITRE ATT&CK logo is displayed. The main content area features a section titled "Procedure Examples" with a table listing various techniques and their descriptions.

ID	Name	Description
S0373	Astaroth	Astaroth uses the LoadLibraryExW() function to load additional modules. ^[6]
S0438	Attor	Attor's dispatcher can execute additional plugins by loading the respective DLLs. ^[7]
S0520	BLINDINGCAN	BLINDINGCAN has loaded and executed DLLs in memory during runtime on a victim machine. ^[8]
S0415	BOOSTWRITE	BOOSTWRITE has used the DWriteCreateFactory() function to load additional modules. ^[9]
S1039	Bumblebee	Bumblebee can use LoadLibrary to attempt to execute GdiPlus.dll. ^[10]
S0673	DarkWatchman	DarkWatchman can load DLLs. ^[11]

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

28 April 2021

RotaJakiro: A long live secret backdoor with 0 VT detection

Alex Turing & Hui Wang

360 Netlab Blog - Network Security Research Lab at 360 — RotaJakiro

Reverse Analysis

The 4 RotaJakiro samples, with time distribution from : sample is selected for analysis in this blog, which has th

At the coding level, RotaJakiro uses techniques and communication protocols to counteract the binary & network traffic analysis.

At the functional level, RotaJakiro first determines whether the user is root or non-root at run time, with different execution policies for different accounts, then decrypts the relevant sensitive resources using AES& ROTATE.

RotaJakiro supports a total of 12 functions, three of which are related to the execution of specific Plugins. Unfortunately, we have no visibility to the plugins, and therefore do not know its true purpose. From a broad backdoor perspective, the functions can be grouped into the following four categories.

- Reporting device information
- Stealing sensitive information
- File/Plugin management (query, download, delete)
- Execution of specific Plugin

V),
dynamically linked (uses shared libs),
nux 2.6.32, stripped
EXPLICIT

At the coding level, RotaJakiro uses techniques and communication protocols to counteract the binary & network traffic analysis.

At the functional level, RotaJakiro first determines whether the user is root or non-root at run time, with different execution policies for different accounts, then decrypts the relevant sensitive resources using AES& ROTATE.

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

28 April 2021

[RotaJakiro: A long live secret backdoor with 0 VT detection](#)

Alex Turing & Hui Wang

The 4 RotaJakiro samples, with time distribution from 2015-12-09 06:24:24 to 2016-01-01 00:00:00. One sample is selected for analysis.

MD5: 64f6cfe44ba08b0babdd390423
ELF 64-bit LSB executable, x86-64
Packer: No

Reverse Analysis

At the coding level, RotaJakiro uses various obfuscation and anti-analysis techniques to counteract the binary analysis process. At the functional level, it implements different execution policies for different accounts, then decrypts the relevant sensitive resources using AES& ROTATE.

RotaJakiro supports a total of 12 functions, three of which are related to the execution of specific Plugins. Unfortunately, we have no visibility to the plugins,

RotaJakiro supports a total of 12 functions, three of which are related to the execution of specific Plugins. Unfortunately, we have no visibility to the plugins, and therefore do not know its true purpose. From a broad backdoor perspective, the functions can be grouped into the following four categories.

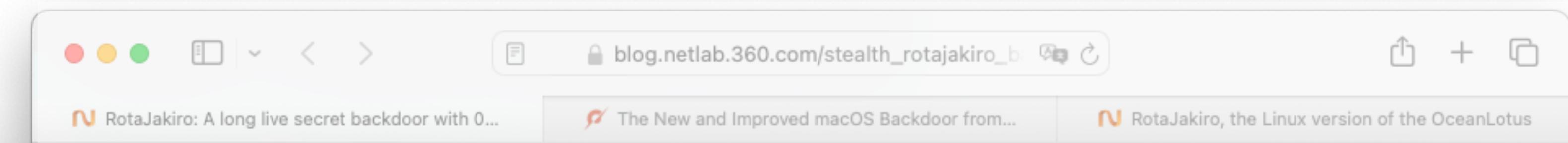
- Reporting device information
- Stealing sensitive information
- File/Plugin management (query, download, delete)
- Execution of specific Plugin



dynamically linked (uses shared libs),

EXPLICIT

28 April 2021



RotaJakiro supports a total of 12 functions, three of which are related to the execution of specific Plugins. Unfortunately, we have no visibility to the plugins,

A screenshot of a web browser displaying a blog post titled "Reverse Analysis" from the "360 Netlab Blog - Network Security Research Lab at 360 — RotaJakiro". The post discusses the analysis of four RotaJakiro samples, mentioning their MD5 hash, file type (ELF 64-bit LSB executable), and lack of packer. It also notes that one sample is selected for analysis.

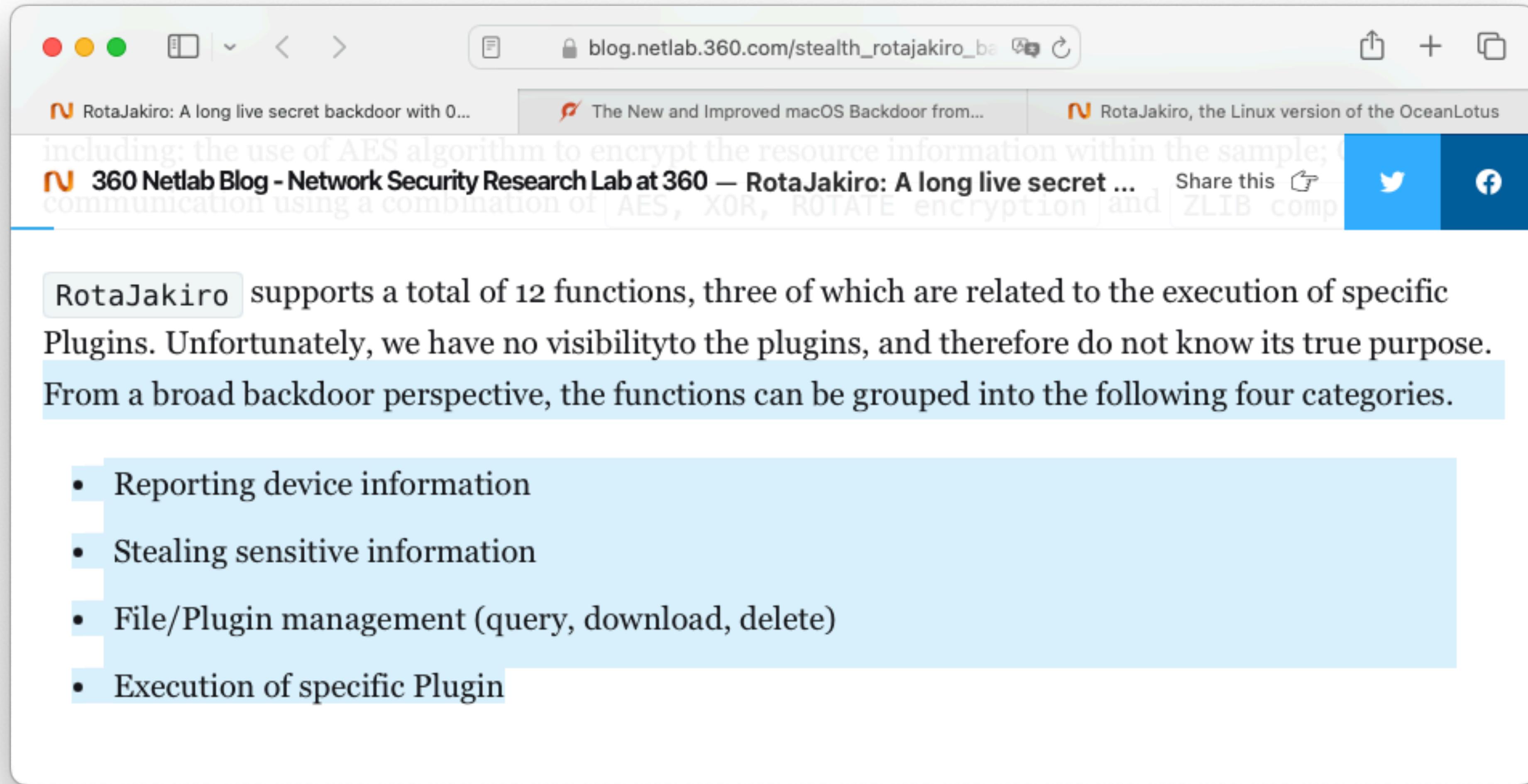
purpose. From a broad backdoor perspective, the functions can be grouped into the following four categories.

- Reporting device information
- Stealing sensitive information
- File/Plugin management (query, download, delete)
- Execution of specific Plugin

A screenshot of a web browser displaying a detailed analysis of a RotaJakiro sample. The analysis includes the MD5 hash (MD5:64f6cfe44ba08b0babdd390423), file type (ELF 64-bit LSB executable, x86-64), and packer status (Packer:No). A large black redaction box covers most of the content below this. A purple arrow points from the text "dynamically linked (uses shared libs)" to the word "EXPLICIT" in a purple box. The text "dynamically linked (uses shared libs)" is partially visible within the redacted area.

At the coding level, RotaJakiro uses various obfuscation and anti-analysis protocols to counteract the binary analysis tools. At the functional level, RotaJakiro implements different execution policies for different accounts, then decrypts the relevant sensitive resources using AES& ROTATE.

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

A screenshot of a web browser window. The address bar shows the URL: blog.netlab.360.com/stealth_rotajakiro_ba... The main content area displays a blog post titled "RotaJakiro: A long live secret backdoor with 0..." by "The New and Improved macOS Backdoor from...". The post discusses the RotaJakiro backdoor, mentioning its AES encryption and communication methods. Below the post, a section titled "RotaJakiro" lists four categories of functions: Reporting device information, Stealing sensitive information, File/Plugin management (query, download, delete), and Execution of specific Plugin.

RotaJakiro supports a total of 12 functions, three of which are related to the execution of specific Plugins. Unfortunately, we have no visibility to the plugins, and therefore do not know its true purpose. From a broad backdoor perspective, the functions can be grouped into the following four categories.

- Reporting device information
- Stealing sensitive information
- File/Plugin management (query, download, delete)
- Execution of specific Plugin

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

IMPLICIT

The screenshot shows a web browser window with the URL blog.netlab.360.com/stealth_rotajakiro_ba. The page content discusses the RotaJakiro backdoor, mentioning its support for 12 functions and four categories of functionality. A purple arrow points from the word 'IMPLICIT' in the top-left corner to the text 'Unfortunately, we have no visibility to the plugins, and therefore do not know its true purpose.' A blue box highlights this sentence.

RotaJakiro supports a total of 12 functions, three of which are related to the execution of specific Plugins. Unfortunately, we have no visibility to the plugins, and therefore do not know its true purpose. From a broad backdoor perspective, the functions can be grouped into the following four categories.

- Reporting device information
- Stealing sensitive information
- File/Plugin management (query, download, delete)
- Execution of specific Plugin

28 April 2021

[RotaJakiro: A long live secret backdoor with 0 VT detection](#)

Alex Turing & Hui Wang

MITRE

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

IMPLICIT

The screenshot shows a web browser window with the URL blog.netlab.360.com/stealth_rotajakiro_ba. The page content discusses the RotaJakiro backdoor, mentioning its long-lived nature and stealthy behavior. A sidebar on the left lists several functions:

- RotaJakiro
- Plugins. Unfo
- From a broad
- Reporting
- Stealing
- File/Plug
- Executio

A purple arrow points from the word "IMPLICIT" in the top-left corner to the "Reporting" item in the sidebar.

- Reporting device information
- Stealing sensitive information
- File/Plugin management (query, download, delete)
- Execution of specific Plugin

28 April 2021

RotaJakiro: A long live secret backdoor with 0 VT detection

Alex Turing & Hui Wang

MITRE

EXERCISE: FIND WHERE THE MITRE ATT&CK TECHNIQUE SHARED MODULES T1129 IS DESCRIBED IN THE REPORTS.

- ▶ Review the [MITRE ATT&CK technique T1129](#)'s procedures. Look for `dlopen` & `dlsym` function calls to narrow it down.
- ▶ In reporting, look for words used to describe T1129 Shared Modules...*plugins, shared libs, dynamic libraries, library, & load*. Look for these words to isolate where the behavior is described in the reporting. *Mappings at the bottom of the report != 100% correct.*

Reports

1. https://www.trendmicro.com/en_us/research/20/k/new-macos-backdoor-connected-to-oceanlotus-surfaces.html
2. <https://unit42.paloaltonetworks.com/unit42-new-improved-macos-backdoor-oceanlotus/>
3. https://www.trendmicro.com/en_us/research/18/d/new-macos-backdoor-linked-to-oceanlotus-found.html
4. <https://www.welivesecurity.com/2019/04/09/oceanlotus-macos-malware-update/>

22 JUNE 2017 - HERNANDEZ & TSECHANSKY - NEW AND IMPROVED MACOS BACKDOOR FROM OCEANLOTUS

EXPLICIT

Commands 0x25D5082, 0x1B25503, 0x1532E65

These commands load a dynamic library using *dlopen()* and obtains a function pointer to execute within that shared library using *dlsym()*. Unfortunately, we do not know which dynamic libraries or functions are used for each command since

These commands load a dynamic library using *dlopen()* and obtains a function pointer to execute within that shared library using *dlsym()*. Unfortunately, we do

same number of arguments with the first being a fairly large constant similar to the command constants, (see Figure 12) and the backdoor has a function for receiving files, it is possible that these functions correspond to a shared library that the server uploads to the victim host. This means that additional functionality can be added to this backdoor by loading modules directly from the C2 server.



22 JUNE 2017 - HERNANDEZ & TSECHANSKY - NEW AND IMPROVED MACOS BACKDOOR FROM OCEANLOTUS

Commands 0x25D5082, 0x1B25503, 0x1532E65

These commands load a dynamic library using *dlopen()* and obtains a function pointer to execute within that shared library using *dlsym()*. Unfortunately, we do not know which dynamic libraries or functions are used for each command since these are server supplied and we were not able to capture any communication that used these commands.

However, we can postulate that since the parameters to the functions have the same number of arguments with the first being a fairly large constant similar to the command constants, (see Figure 12) and the backdoor has a function for receiving files, it is possible that these functions correspond to a shared library that the server uploads to the victim host. This means that additional functionality can be added to this backdoor by loading modules directly from the C2 server.

IMPLICIT

22 JUNE 2017 - HERNANDEZ & TSECHANSKY - NEW AND IMPROVED MACOS BACKDOOR FROM OCEANLOTUS

The screenshot shows a web browser window with the URL unit42.paloaltonetworks.com/unit42-new-improved-macos-backdoor-oceanlotus.html. The main content on the page is titled "Commands 0x25D5082, 0x1B25503, 0x1532E65". Below the title, there is a note: "These commands load a dynamic library using dlopen() and obtains a function".

However, we can postulate that since the parameters to the functions have the same number of arguments with the first being a fairly large constant similar to the command constants, (see Figure 12) and the backdoor has a function for

However, we can postulate that since the parameters to the functions have the same number of arguments with the first being a fairly large constant similar to the command constants, (see Figure 12) and the backdoor has a function for receiving files, it is possible that these functions correspond to a shared library that the server uploads to the victim host. This means that additional functionality can be added to this backdoor by loading modules directly from the C2 server.

IMPLICIT

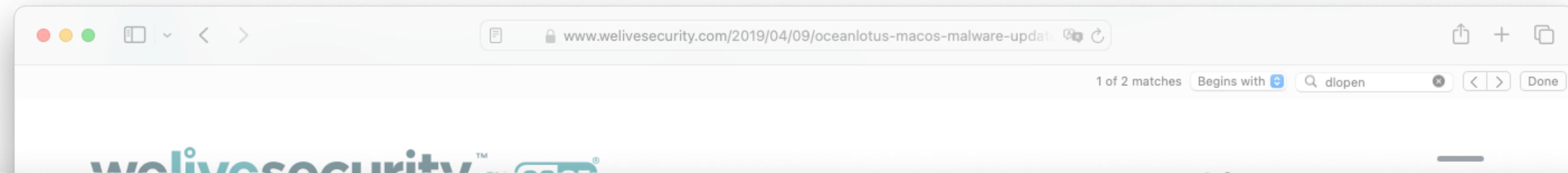
22 JUNE 2017 – HERNANDEZ & TSECHANSKY – NEW AND IMPROVED MACOS BACKDOOR FROM OCEANLOTUS

The screenshot shows a web browser window with the URL www.welivesecurity.com/2019/04/09/oceanlotus-macos-malware-update/. The page content discusses a configuration change in the malware sample where it no longer uses the `curl` library for network exfiltration, instead using an external library. It details how the backdoor tries to decrypt files in the current directory using AES-256-CBC with a specific key, pads them with zeroes, and saves them to `/tmp/store`. It then attempts to load these decrypted files as libraries using the `dlopen` function. The text also mentions two exported functions, `Boriry` and `ChadylonV`, which handle network communication.

On top of this configuration change, this sample does not use the `curl` library for network exfiltration. Instead, it uses an external library. To locate it, the backdoor tries to decrypt each file in the current directory using AES-256-CBC with the key gFjMXBgyXWULmVVVzyxy padded with zeroes. Each file is “decrypted” and saved as /tmp/store and an attempt to load it as a library made using the `dlopen` function. When a decryption attempt results in a successful call to `dlopen`, the backdoor then retrieves the exported functions `Boriry` and `ChadylonV`, which seem to be responsible for the network communication with the server. As we do not have the dropper or other files from the original sample’s location, we could not analyse this library. Moreover, since the component is encrypted, a YARA rule based on these strings would not match the file found on disk.

BOTH

22 JUNE 2017 – HERNANDEZ & TSECHANSKY – NEW AND IMPROVED MACOS BACKDOOR FROM OCEANLOTUS



/tmp/store and an attempt to load it as a library made using the **dlopen** function. When a decryption attempt results in a successful call to **dlopen**, the backdoor then retrieves the exported functions **Boriry** and **ChadylonV**, which seem to be responsible for the network communication with the server. As we do not have the dropper or other files from

gFjMXBgYXWULmVVVzyxy padded with zeroes. Each file is decrypted and saved as

/tmp/store and an attempt to load it as a library made using the **dlopen** function. When a decryption attempt results in a successful call to **dlopen**, the backdoor then retrieves the exported functions **Boriry** and **ChadylonV**, which seem to be responsible for the network communication with the server. As we do not have the dropper or other files from

the original sample's location, we could not analyse this library. Moreover, since the component is encrypted, a YARA rule based on these strings would not match the file found on disk.

BOTH

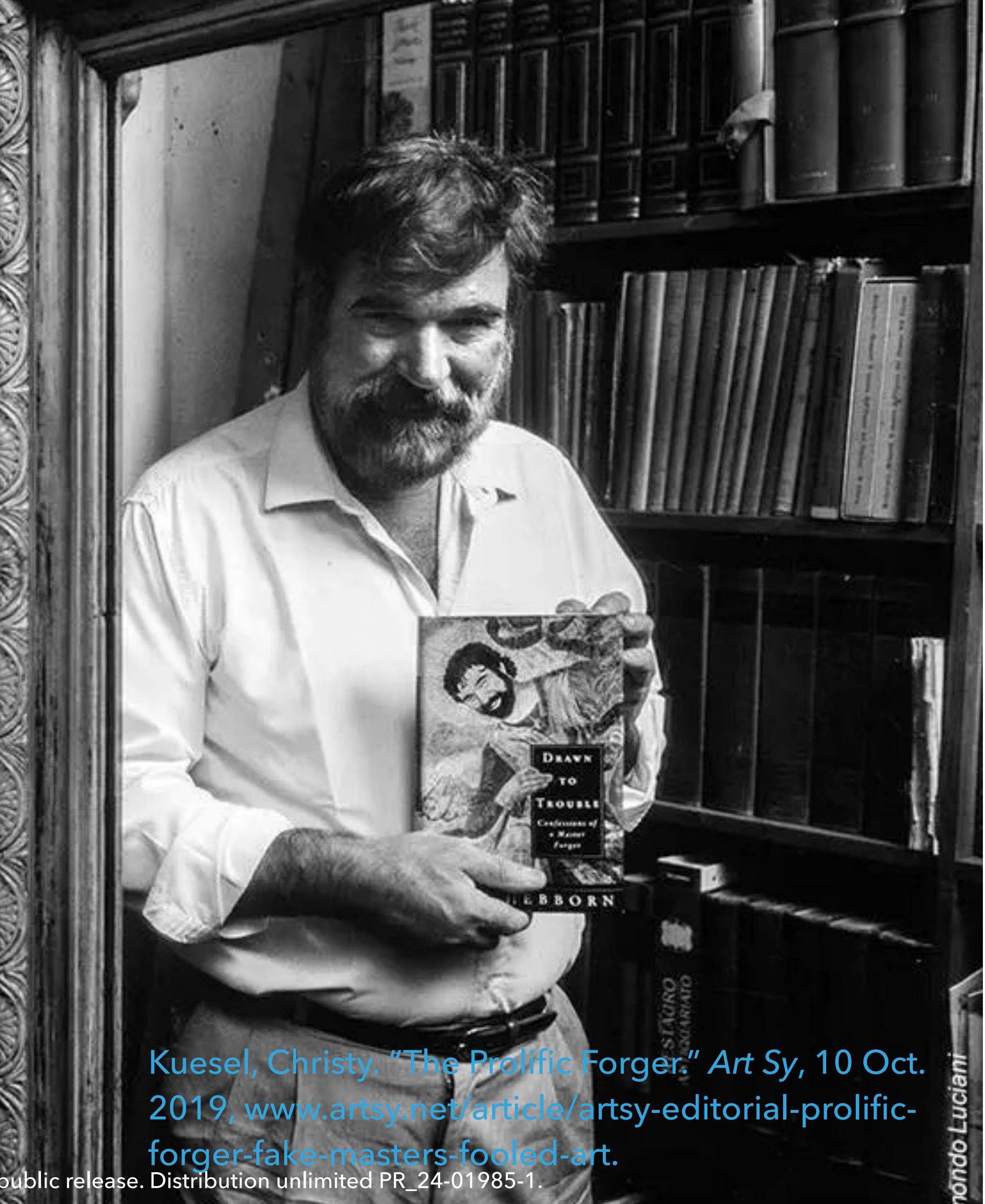
ANSWER: MITRE ATT&CK TECHNIQUE

Find the macOS version of this technique from the following reports

Reports

1. https://www.trendmicro.com/en_us/research/20/k/new-macos-backdoor-connected-to-oceanlotus-surfaces.html
2. <https://unit42.paloaltonetworks.com/unit42-new-improved-macos-backdoor-oceanlotus/>
3. https://www.trendmicro.com/en_us/research/18/d/new-macos-backdoor-linked-to-oceanlotus-found.html
4. <https://www.welivesecurity.com/2019/04/09/oceanlotus-macos-malware-update/>

1934-1996 ERIC HEBBORN

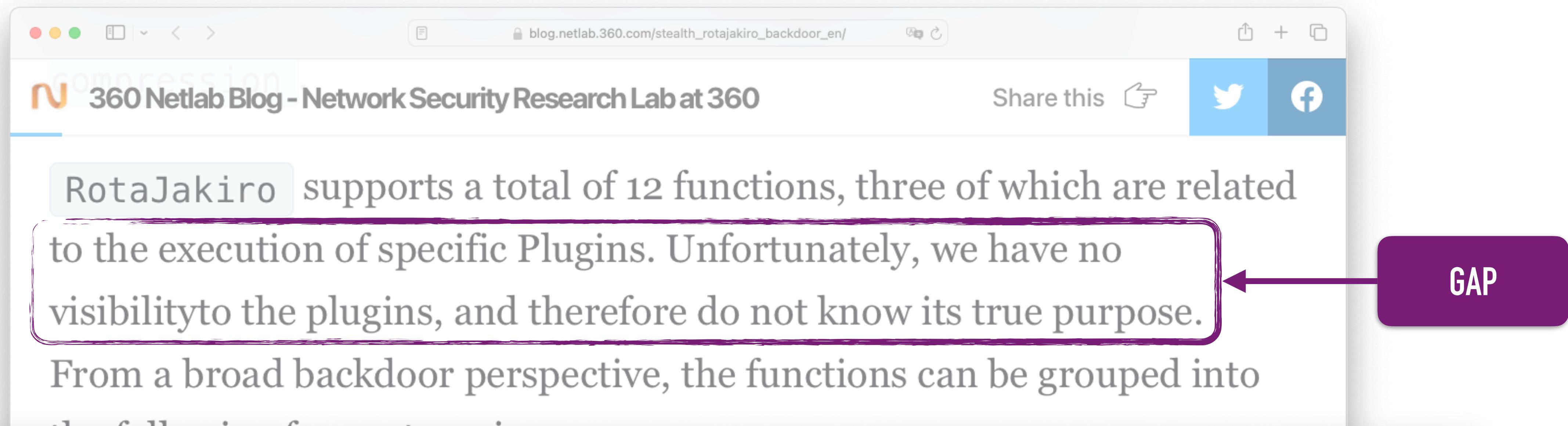


Kuesel, Christy. "The Prolific Forger." *Art Sy*, 10 Oct. 2019, www.artsy.net/article/artsy-editorial-prolific-forger-fake-masters-fooled-art.

EXERCISE: WRITE OUT IN PSEUDO CODE OR DIAGRAMS AN OUTLINE OF A PROGRAM TO EXECUTE THE T1129 SHARED MODULES MITRE ATT&CK TECHNIQUE

- ▶ What components need to be built?
- ▶ Think through the logic flow of what is needed to perform the T1129 & how to confirm it was executed
- ▶ Hint: .dylib is the macOS version of a .so file, how would you use dlopen & dlsym using .dylib files?

GAP ANALYSIS



RotaJakiro supports a total of 12 functions, three of which are related to the execution of specific Plugins. Unfortunately, we have no visibility to the plugins, and therefore do not know its true purpose.

From a broad backdoor perspective, the functions can be grouped into the following categories:

to the execution of specific Plugins. Unfortunately, we have no visibility to the plugins, and therefore do not know its true purpose.

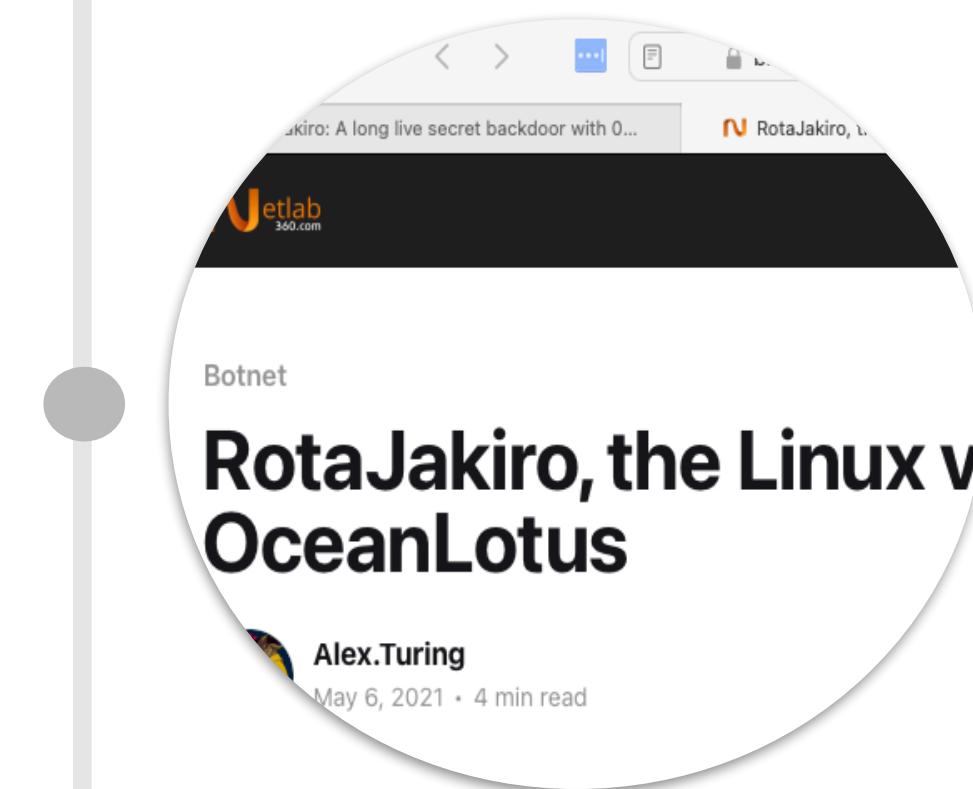
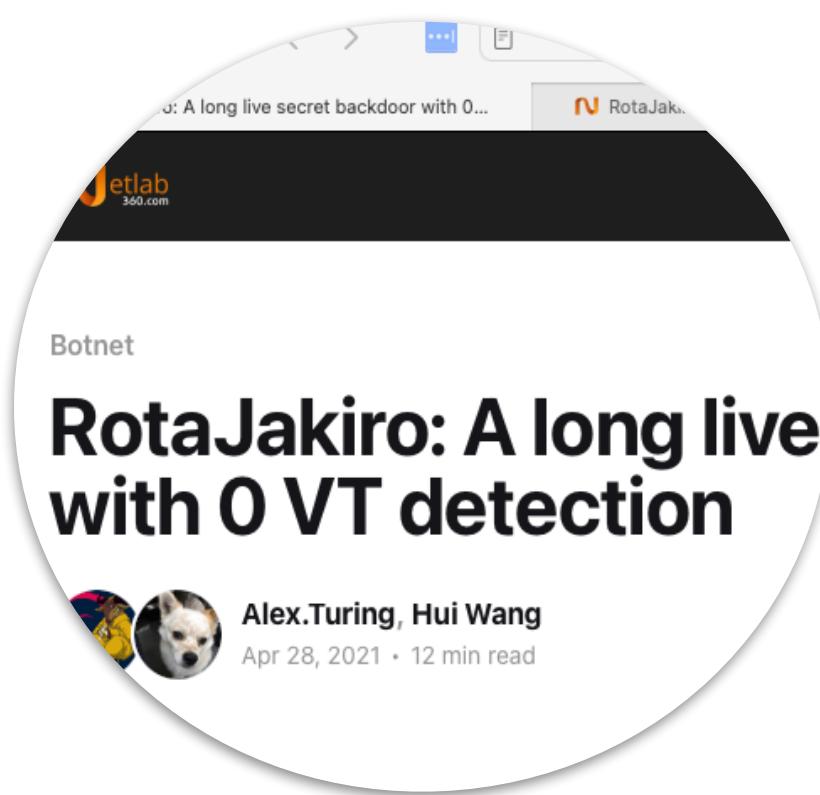
- Stealing sensitive information
- File/Plugin management (query, download, delete)
- Execution of specific Plugin

REPORTING TIMELINE

6 May 2021

Alex Turing

[RotaJakiro, the Linux
version of the OceanLotus](#)



28 April 2021

Alex Turing & Hui Wang

[RotaJakiro: A long live
secret backdoor with 0 VT
detection](#)

COPY CA

REPOI

6 May

Alex Turin

RotaJa

version

VT



Alex.Turing

May 6, 2021 • 4 min read

On Apr 28, we published our [RotaJakiro](#) backdoor blog, at that time, we didn't have the answer for a very important question, what is this backdoor exactly for? We asked the community for clues and two days ago we got a hint, PE (Thanks!) wrote the following comment on our blog post.



PE • 20 hours ago

Wow. Amazing work. Here's a real comparison for you Jerry... looks a lot like the C2 activity associated with OceanLotus from 2016...
[https://www.virustotal.com/...](https://www.virustotal.com/)

▲ ▾ • Reply • Share ›



MITRE

©2024 The MITRE Corporation. ALL RIGHTS RESERVED Approved for public release. Distribution unlimited PR_24-01985-1.

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

REPORTING TIMELINE

22 June 2017

Erie Hernandez & Danny Tsechansky

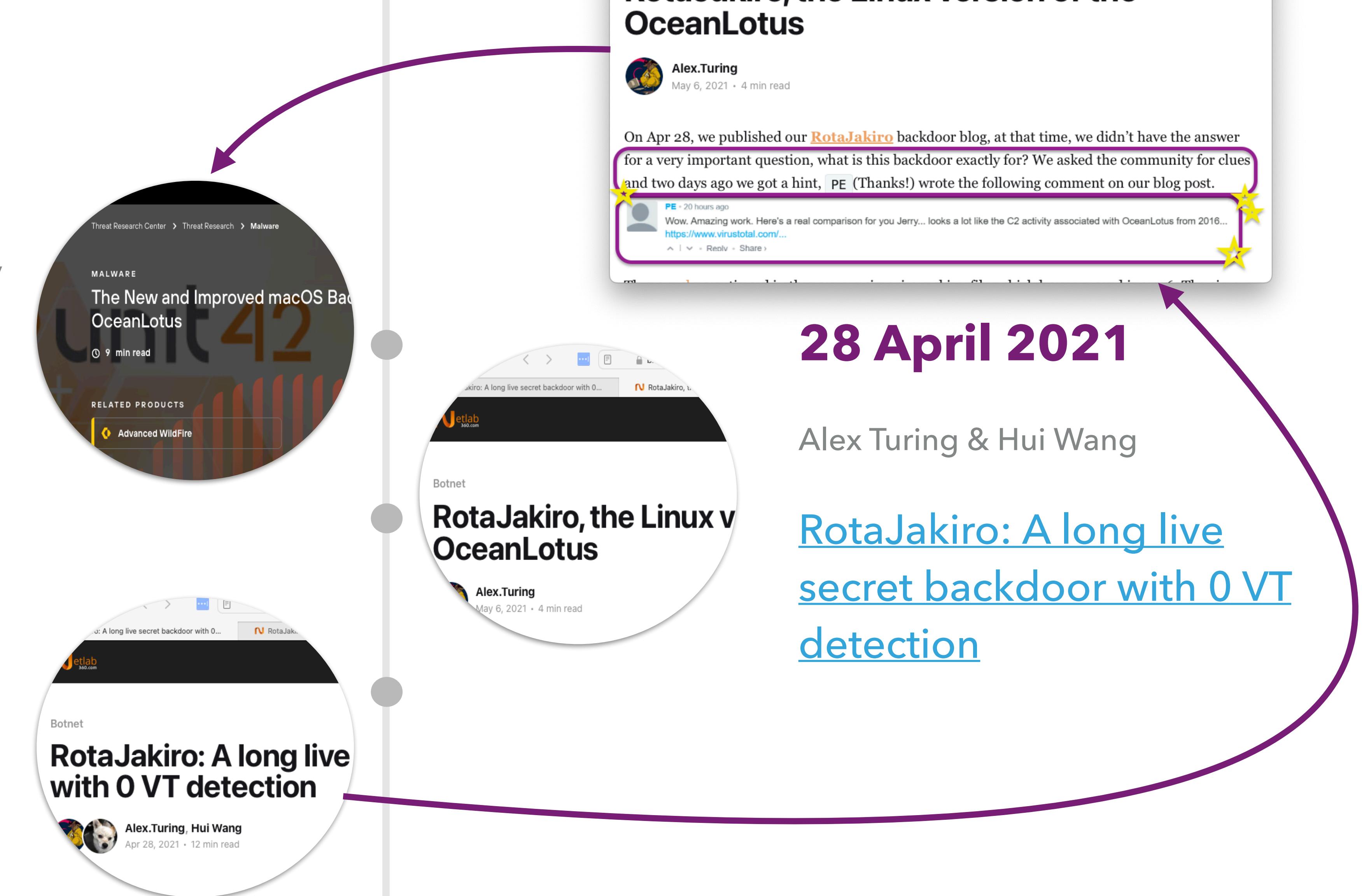
The New and Improved
macOS Backdoor from
OceanLotus

6 May 2021

Alex Turing

RotaJakiro, the Linux
version of the OceanLotus

MITRE



RotaJakiro, the Linux version of the OceanLotus

Alex.Turing
May 6, 2021 • 4 min read

On Apr 28, we published our [RotaJakiro](#) backdoor blog, at that time, we didn't have the answer for a very important question, what is this backdoor exactly for? We asked the community for clues and two days ago we got a hint, [PE](#) (Thanks!) wrote the following comment on our blog post.

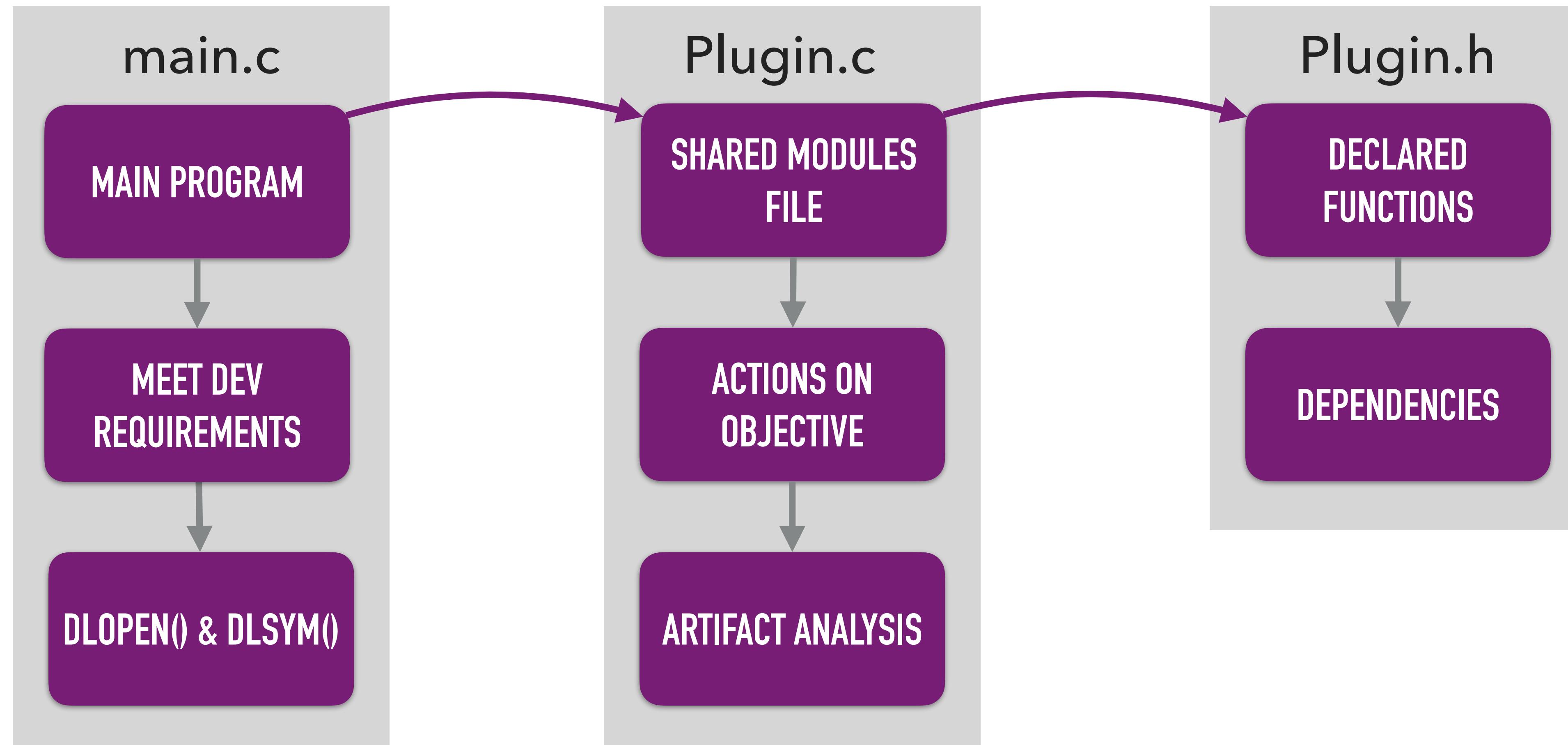
PE • 20 hours ago
Wow. Amazing work. Here's a real comparison for you Jerry... looks a lot like the C2 activity associated with OceanLotus from 2016...
[https://www.virustotal.com/...](https://www.virustotal.com/)
▲ | ▾ + Replv Share ▾

28 April 2021

Alex Turing & Hui Wang

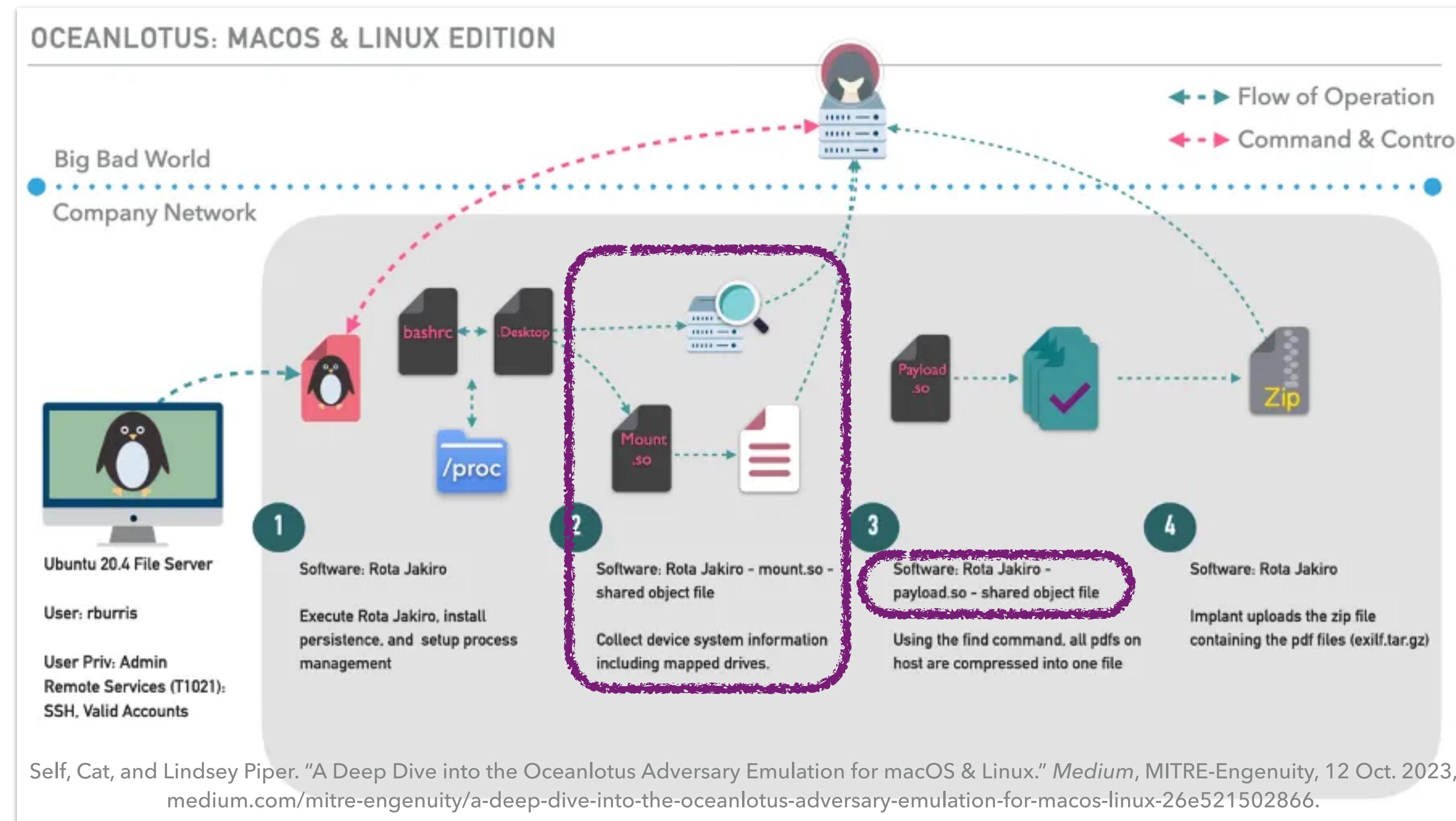
RotaJakiro: A long live
secret backdoor with 0 VT
detection

LINUX PROGRAMMING OUTLINE



COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

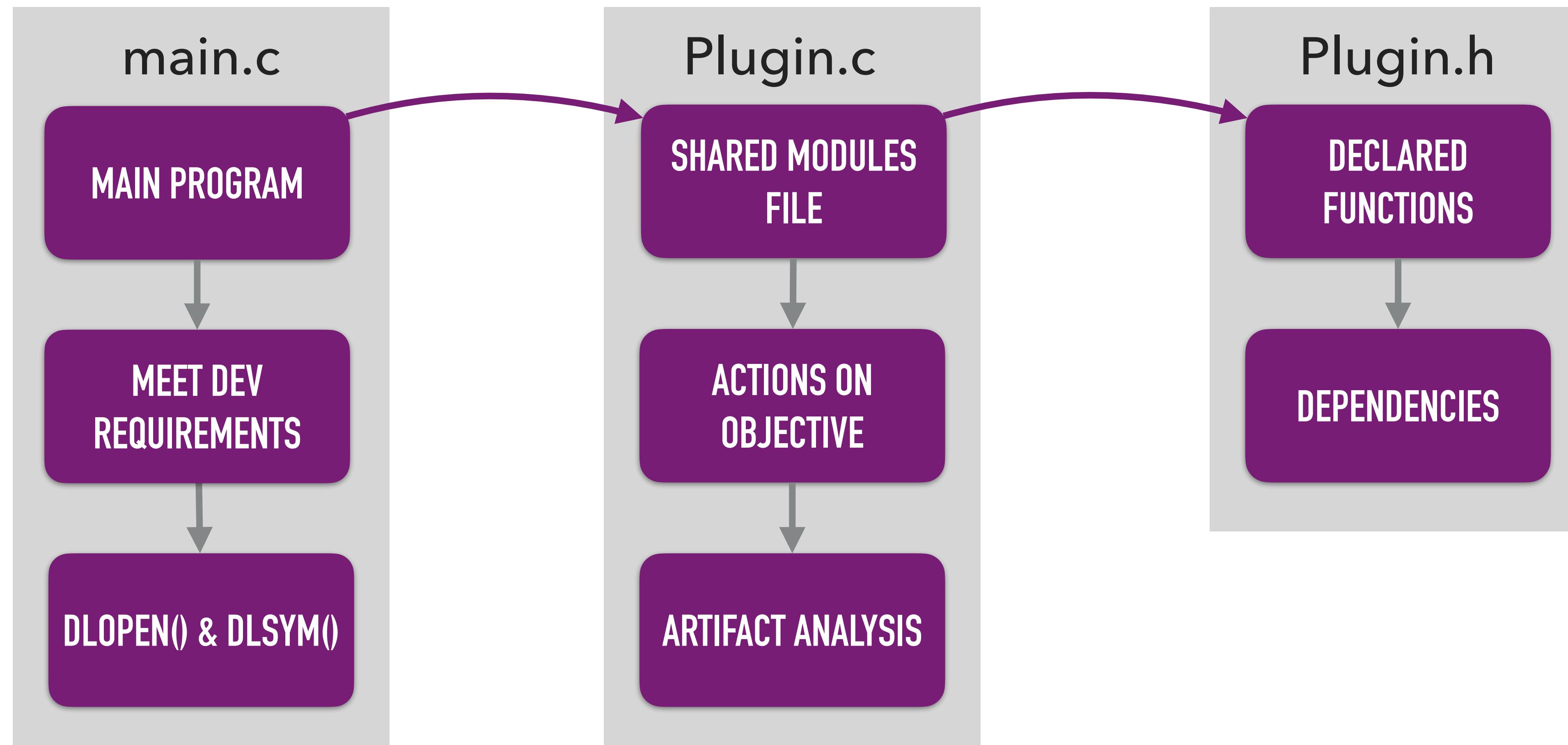
PROCEDURE FLOW



EXERCISE: WRITE OUT IN PSEUDO CODE OR DIAGRAMS AN OUTLINE OF A PROGRAM TO EXECUTE THE T1129 SHARED MODULES MITRE ATT&CK TECHNIQUE

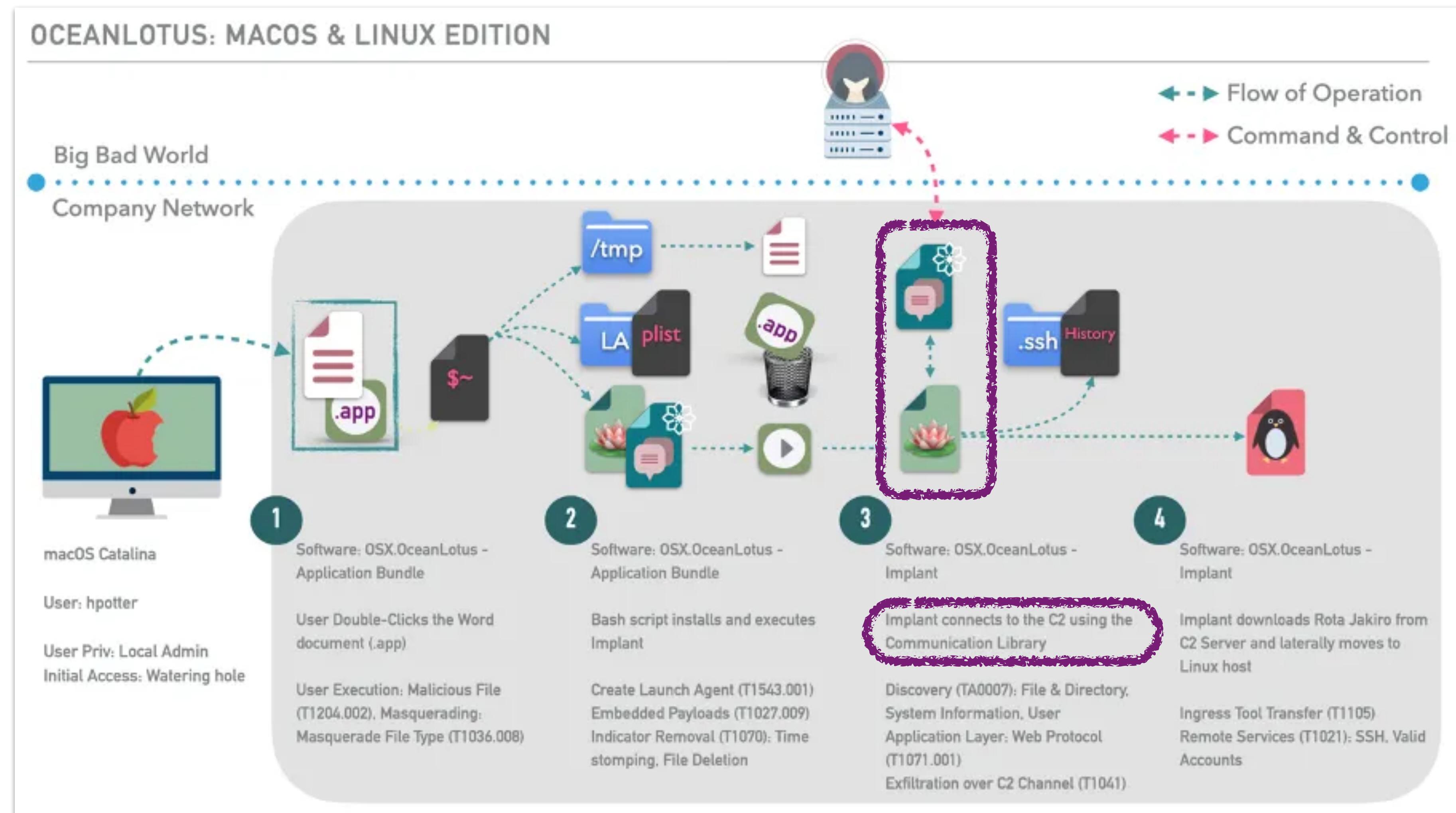
- ▶ What components need to be built?
- ▶ Think through the logic flow of what is needed to perform the T1129 & how to confirm it was executed
- ▶ Hint: .dylib is the macOS version of a .so file, how would you use dlopen & dlsym using .dylib files?

MACOS PROGRAMMING OUTLINE



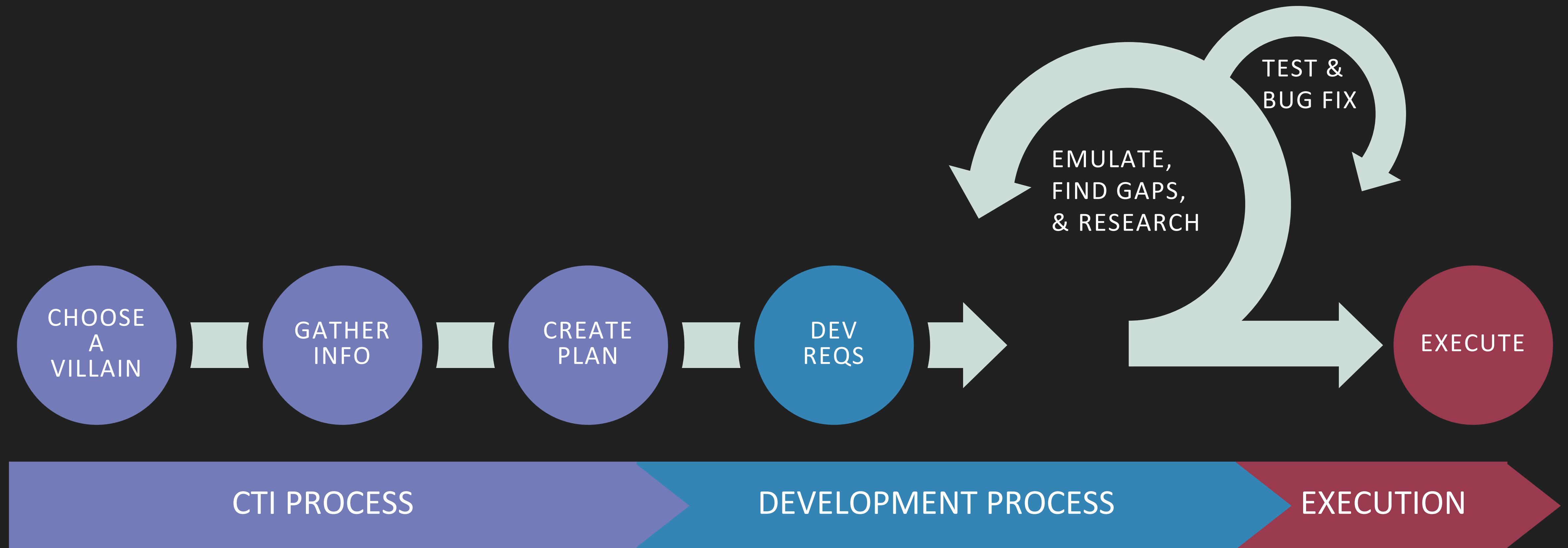
COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

MACOS PROCEDURE FLOW



Self, Cat, and Lindsey Piper. "A Deep Dive into the Oceanlotus Adversary Emulation for macOS & Linux." Medium, MITRE-Engenuity, 12 Oct. 2023, medium.com/mitre-engenuity/a-deep-dive-into-the-oceanlotus-adversary-emulation-for-macos-linux-26e521502866.

BECOMING A DARK KNIGHT



Cat [@coolestcatiknow](#)

Kate [@phish4answers](#)

EXERCISE: CODE A PROCEDURE FOR MACOS

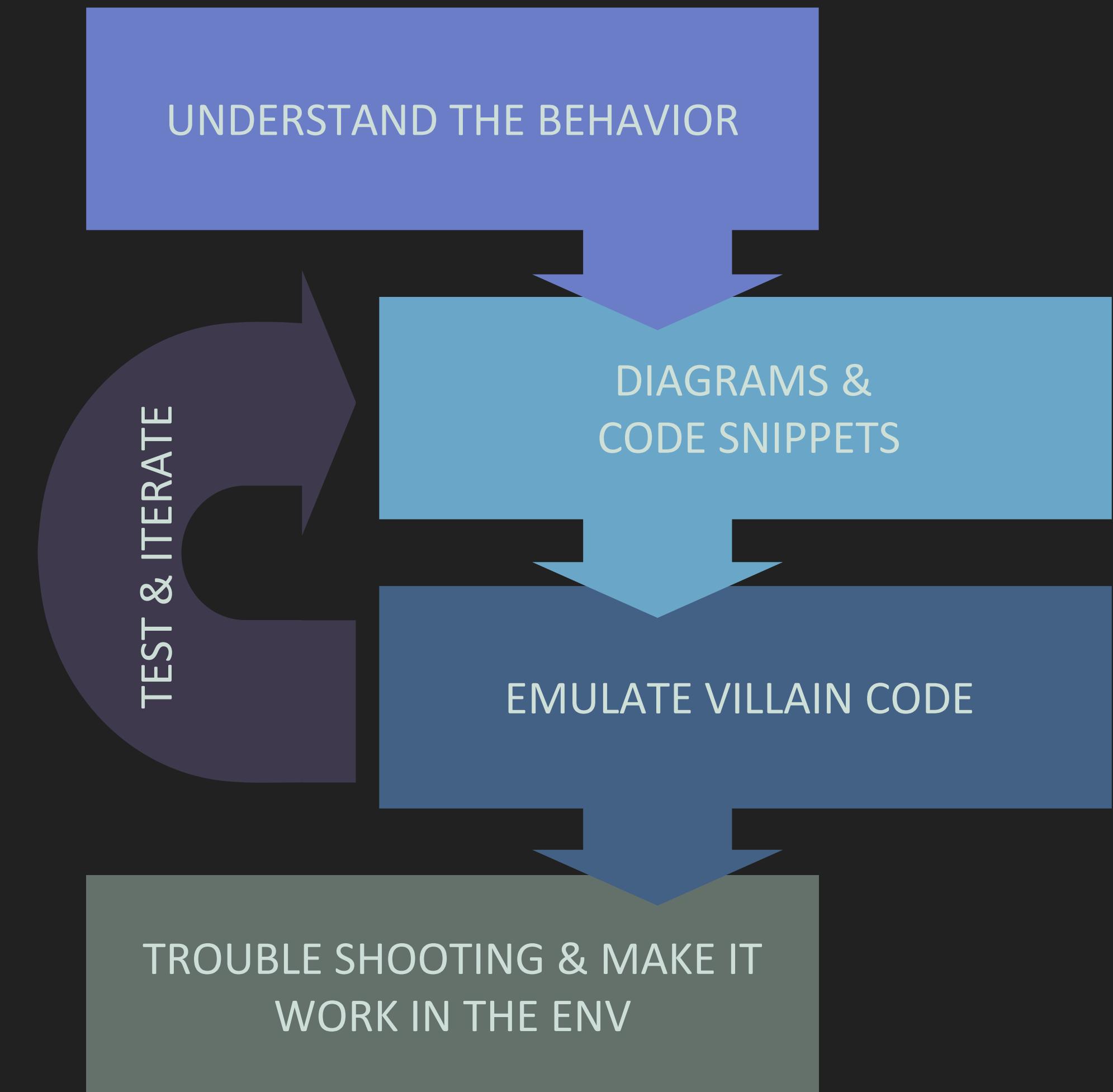
Build out a small program that executes the shared module technique.

Hint: Start by recreating the Linux example, then modify to use a dylib

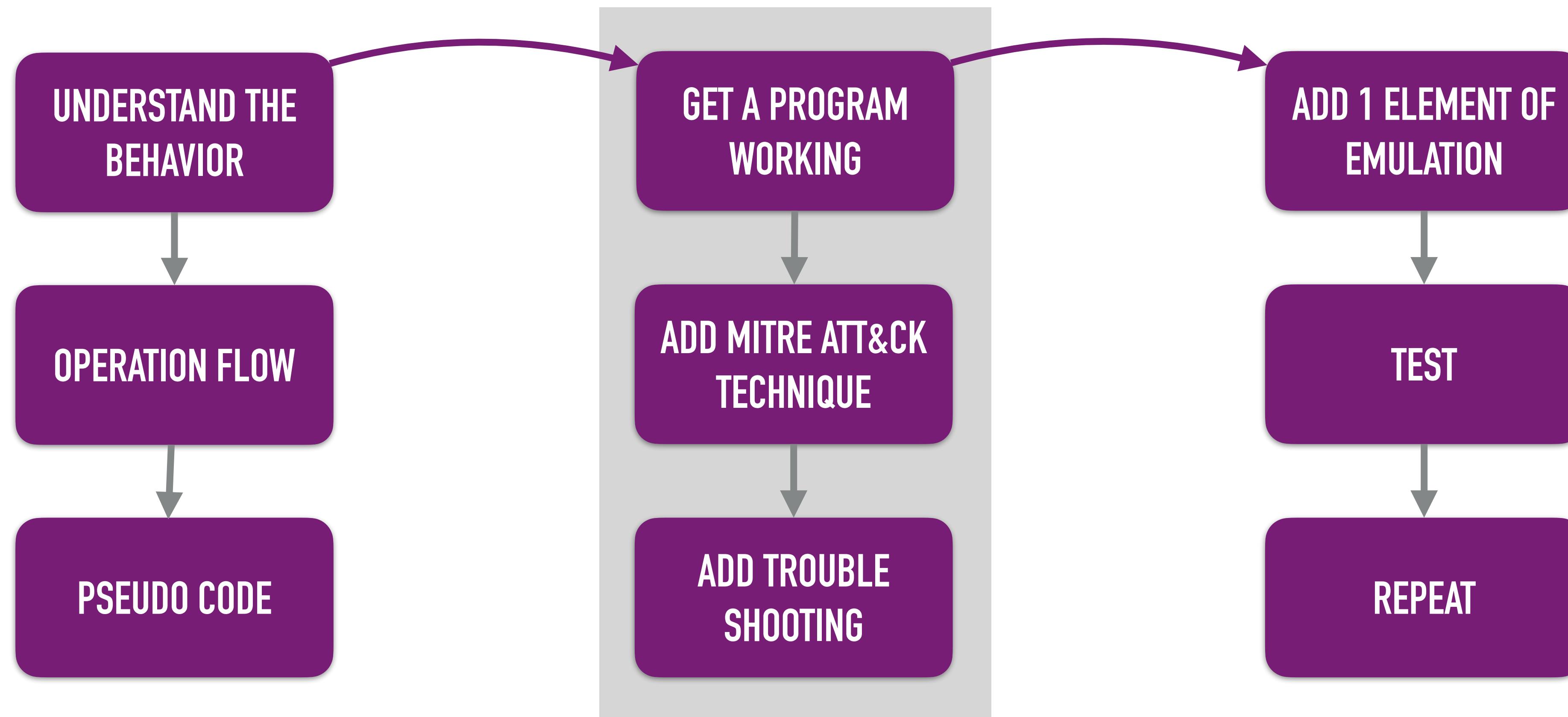
Useful Resources

1. https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/ocean_lotus/Resources/OSX.OceanLotus
2. <https://unit42.paloaltonetworks.com/unit42-new-improved-macos-backdoor-oceanlotus/>
3. <https://www.welivesecurity.com/2019/04/09/oceanlotus-macos-malware-update/>

BREAKING DOWN EACH STEP AKA RABBIT HOLE PROCESS



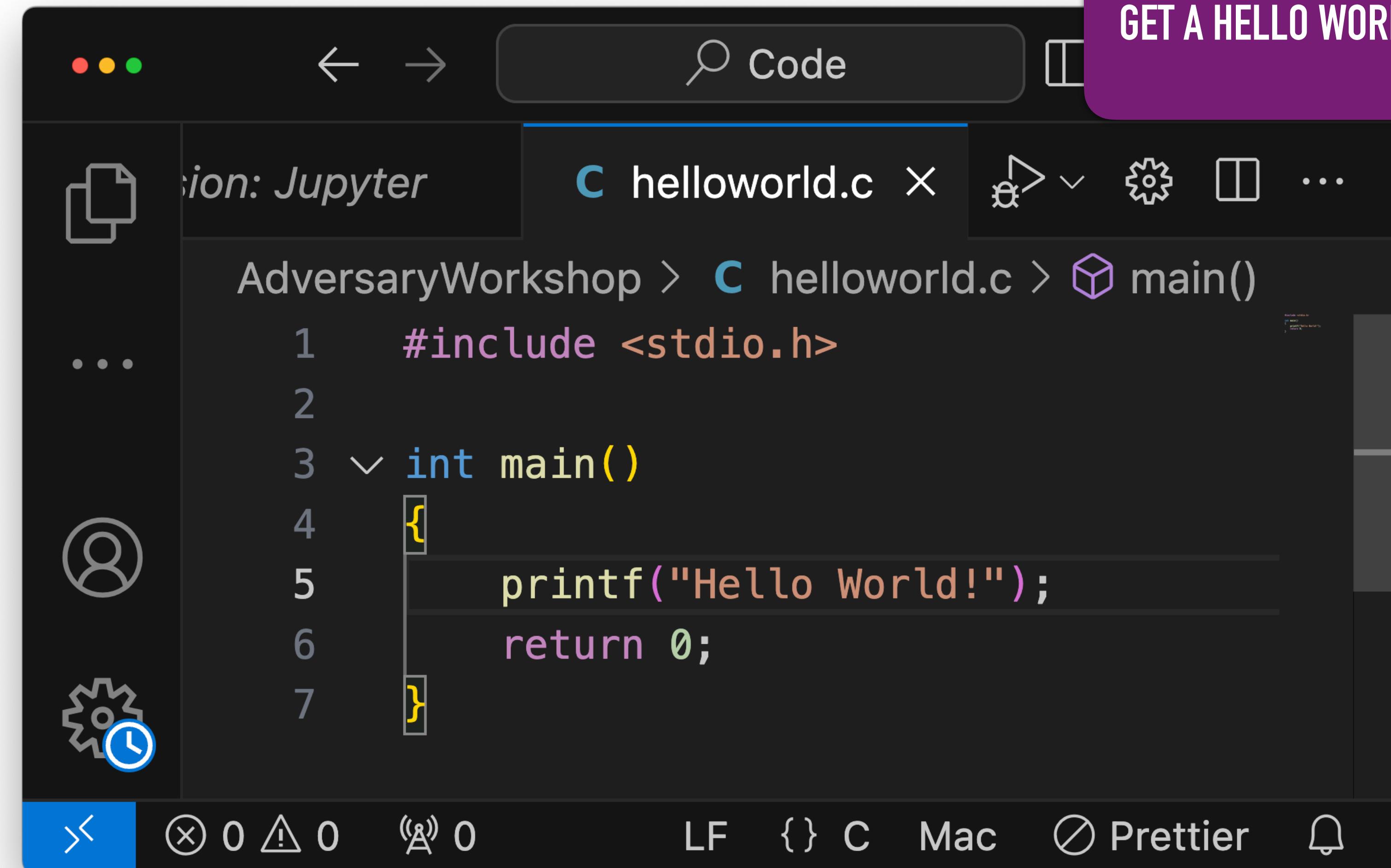
LINUX PROGRAMMING PROCESS



BUILD A HELLO WORLD PROGRAM

1

GET A HELLO WORLD PROGRAM WORKING



A screenshot of a code editor interface, likely Jupyter Notebook, showing a C program for "Hello World". The code is as follows:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World!");
6     return 0;
7 }
```

The code editor has a dark theme with light-colored syntax highlighting. The status bar at the bottom shows file statistics: 0 errors, 0 warnings, 0 info messages, LF line endings, and Mac OS X file format.

STRUCTURE

2

CREATE TWO FILES (MAIN.C & PLUGIN.C)

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows a project structure with a folder named "AdversaryWorkshop". Inside this folder are two files: "main.c" and "plugin.c". Both files are highlighted with a pink box.
- Code Editor:** Two tabs are open:
 - main.c:** Contains the following code:

```
1 #include <stdio.h>
2
3 int main (){
4     printf("Hello hackers!");
5     return 0;
6 }
```
 - plugin.c:** Contains the following code:

```
1 #include <stdio.h>
2
3 void plugin_func(){
4     printf("Hello from the plugin!\n");
5 }
```
- Bottom Bar:** Includes icons for search, file operations, and status indicators like line count (Ln 2, Col 1 (1)).

3

CREATE A HELLO WORLD PLUGIN FUNCTION

Fisher, James. "How to Make Plugins with `Dlopen`." Jamesfisher.com, 24 Aug. 2017, jamesfisher.com/2017/08/24/dlopen/. Accessed 2 Aug. 2024.

MAIN FUNCTIONALITY

- ▶ Use manuals to understand program logic - [dlopen\(\)](#) & [dlsym\(\)](#)
- ▶ Keep it simple - then build on

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with icons for file operations (1), search, sharing, and other settings. The main area displays a C file named 'main.c'.

The code in 'main.c' is as follows:

```
1 #include "plugin.h"
2
3 int main() {
4     // Open the needed object
5     void* handle = dlopen("plugin.so", RTLD_LAZY);
6     // Return address of function to execute
7     void (*functionPtr)() = dlsym(handle, "plugin_func");
8     printf("Calling plugin\n");
9     // Call the function
10    functionPtr();
11    // Clean up
12    dlclose(handle);
13
14 }
```

A large green callout bubble with the number '4' and the text 'CREATE THE SHARED MODULE FUNCTION IN MAIN' is positioned over the code. A pink arrow points from the word 'dlopen' in line 5 to the 'dlopen' call in the code. Another pink arrow points from the word 'dlsym' in line 7 to its call in the code. A third pink arrow points from the word 'dlclose' in line 12 to its call in the code.

ERROR HANDLING

The screenshot shows the AdversaryWorkshop IDE interface. The top bar includes navigation icons, a search bar labeled "Code", and window controls. The left sidebar has icons for file operations, search, and other tools. The main workspace shows three tabs: "C plugin.h", "C main.c", and "C plugin.c". The "main.c" tab is active, showing the file structure: "AdversaryWorkshop > C main.c > main()". The code editor displays the following C code:

```
int main() {  
    // Open the needed object  
    void* handle = dlopen("plugin.so", RTLD_LAZY);  
    if (handle == NULL) {  
        fprintf(stderr, "Could not open plugin: %s\n", dlerror());  
        return 1;  
    }  
  
    // Return the pointer of the function to execute from teh char  
    void (*functionPtr)() = dlsym(handle, "function_name");  
    if (functionPtr == NULL) {  
        (char [32])"Could not find plugin_func: %s\n"  
    }  
}
```

A purple callout bubble with the number "5" and the text "ADD ERROR HANDLING FOR A GOLD ⭐" points to the error handling code starting at line 7. The entire code block is highlighted with a pink rounded rectangle.

WHAT YOU SHOULD HAVE AT THIS POINT

Code

Two .c Files

1 main.c

2 plugin.c

AdversaryWorkshop > C main.c > main()

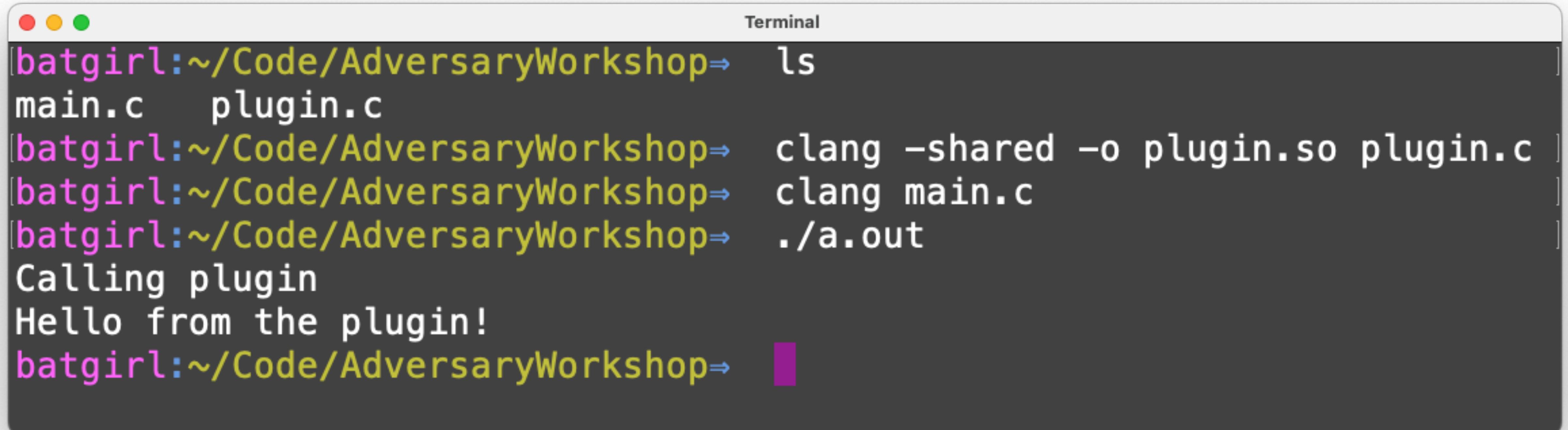
```
1 #include <dlfcn.h>
2 #include <stdio.h>
3
4 int main() {
5
6     // Open the needed object
7     void* handle = dlopen("plugin.so",
8     if (handle == NULL) {
9         fprintf(stderr, "Could not open plugin\n");
10    return 1;
11 }
12
13 // Return the pointer of the function
14 void (*functionPtr)() = dlsym(handle, "plugin_func");
15 if (functionPtr == NULL) {
16     fprintf(stderr, "Could not find function\n");
17 }
```

AdversaryWorkshop > C plugin.c > plugin_func()

```
1 #include <stdio.h>
2
3 void plugin_func(){
4     printf("Hello from the plugin!\n");
5 }
```

©2024 The MITRE Corporation. ALL RIGHTS RESERVED Approved for public release. Distribution unlimited PR_24-01985-1.

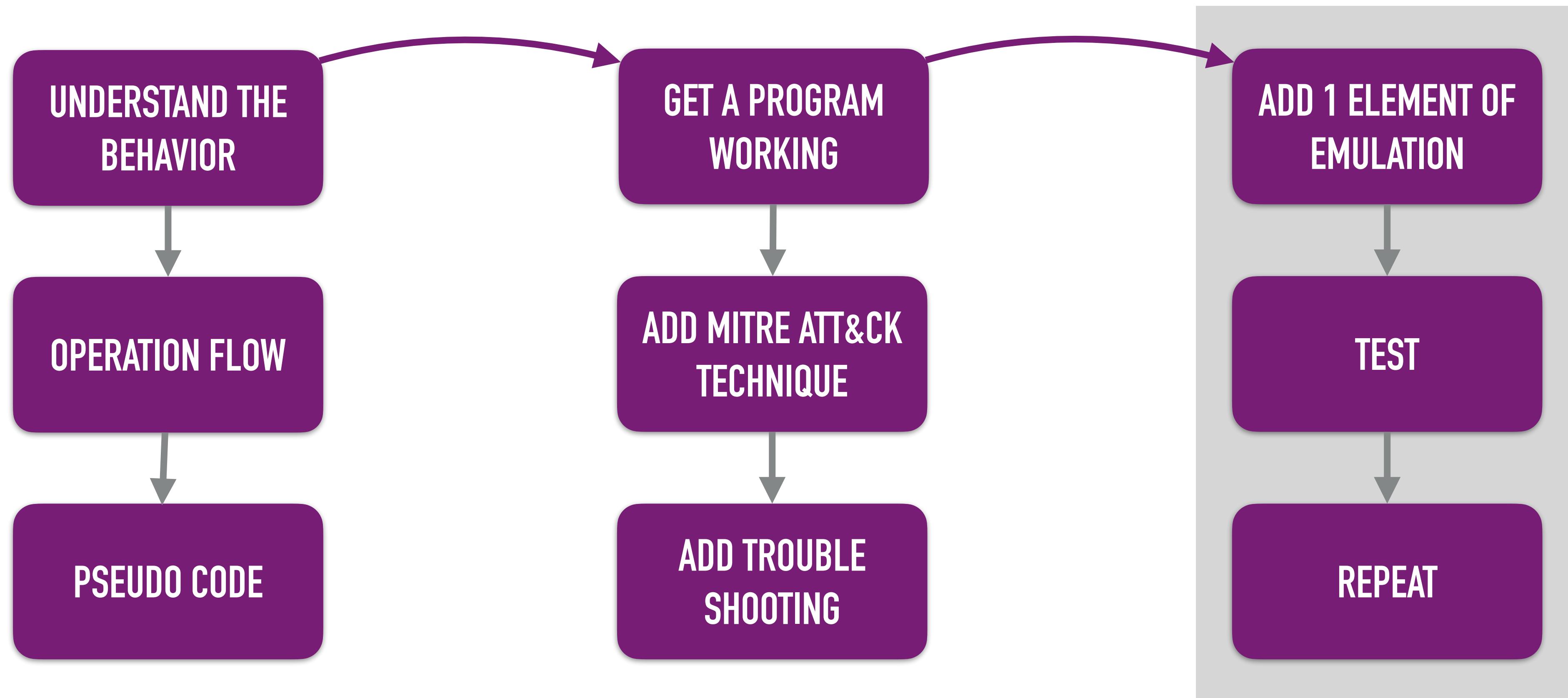
GET IT WORKING



A screenshot of a macOS Terminal window titled "Terminal". The window shows a sequence of commands being run in a directory named "AdversaryWorkshop". The commands include listing files, compiling a plugin, linking it with the main program, and finally executing the resulting binary. The output of the execution shows the plugin's message: "Hello from the plugin!". The terminal has its characteristic red, yellow, and green window control buttons at the top left.

```
batgirl:~/Code/AdversaryWorkshop⇒ ls
main.c  plugin.c
batgirl:~/Code/AdversaryWorkshop⇒ clang -shared -o plugin.so plugin.c
batgirl:~/Code/AdversaryWorkshop⇒ clang main.c
batgirl:~/Code/AdversaryWorkshop⇒ ./a.out
Calling plugin
Hello from the plugin!
batgirl:~/Code/AdversaryWorkshop⇒
```

LINUX PROGRAMMING PROCESS



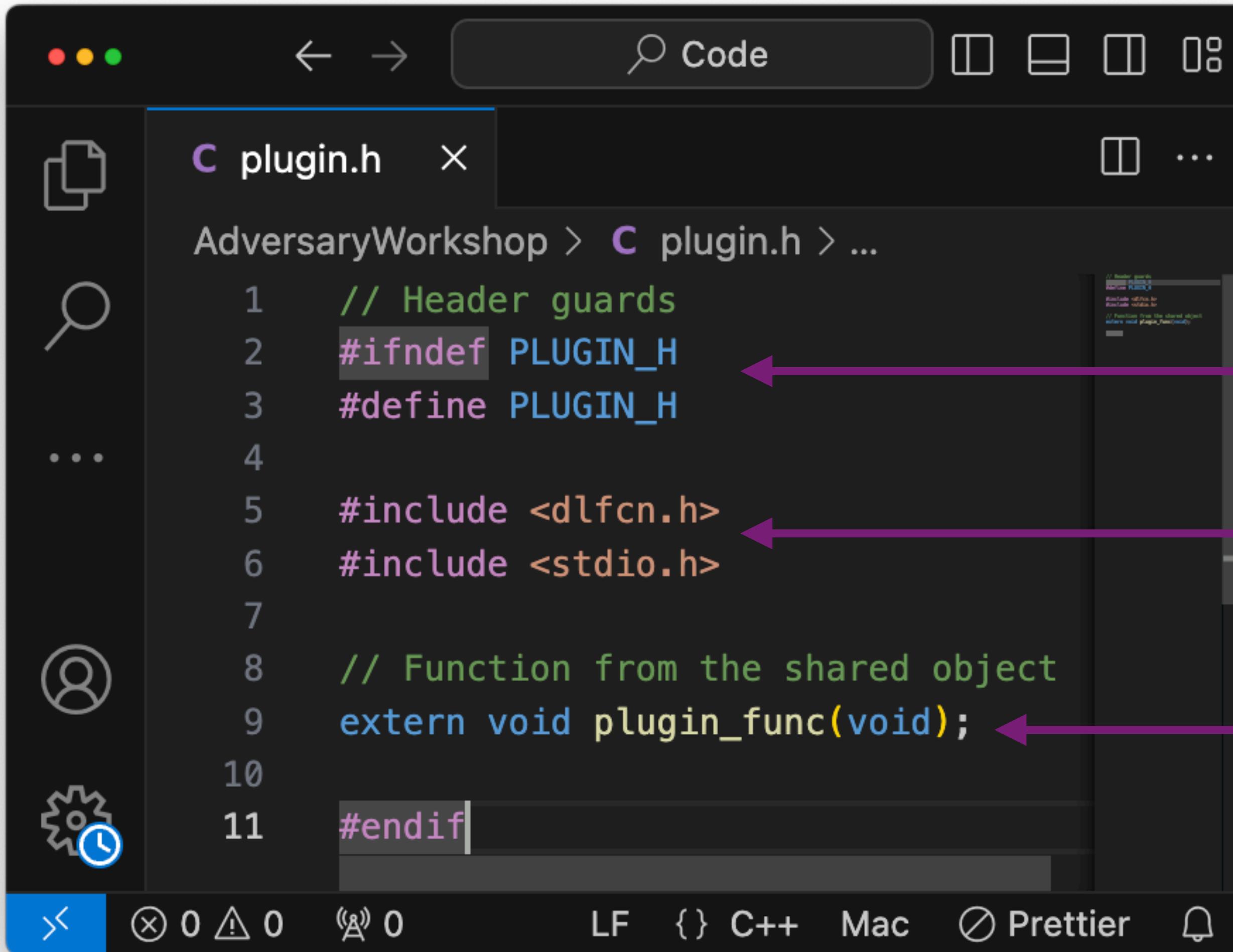
BREAK OUT THE SHARED LIBRARY WITH A HEADER FILE

CREATE A HEADER FILE → MOVE LIBRARIES → ADD HEADER GUARDS → EXPORT THE FUNCTION

The screenshot shows a code editor interface with three files open:

- main.c**: Includes `#include "plugin.h"`. It contains a `main()` function that opens a shared library and prints a message.
- plugin.c**: Includes `#include "plugin.h"`. It contains a `plugin_func()` function that prints a message.
- plugin.h**: Contains header guards (`#ifndef PLUGIN_H`, `#define PLUGIN_H`) and function declarations (`extern void plugin_func(void);`).

WHAT YOU SHOULD HAVE AT THIS POINT



```
Code
```

C plugin.h X

AdversaryWorkshop > C plugin.h > ...

```
1 // Header guards
2 #ifndef PLUGIN_H
3 #define PLUGIN_H
4
5 #include <dlfcn.h>
6 #include <stdio.h>
7
8 // Function from the shared object
9 extern void plugin_func(void);
10
11#endif
```

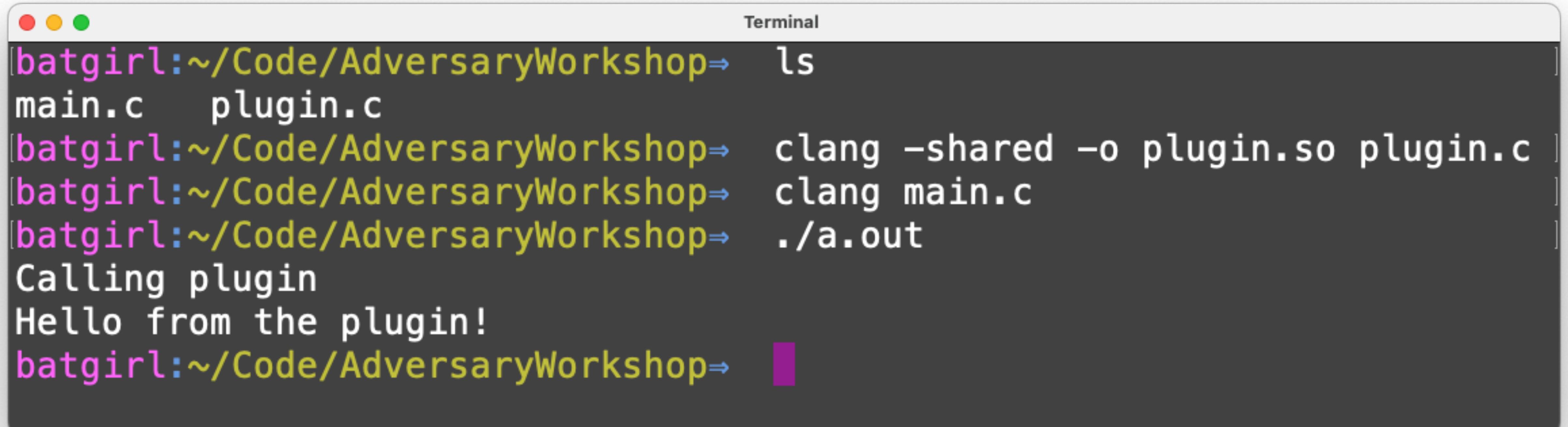
LF {} C++ Mac ⚡ Prettier 📰

HEADER GUARDS

DEPENDENCIES

FUNCTION

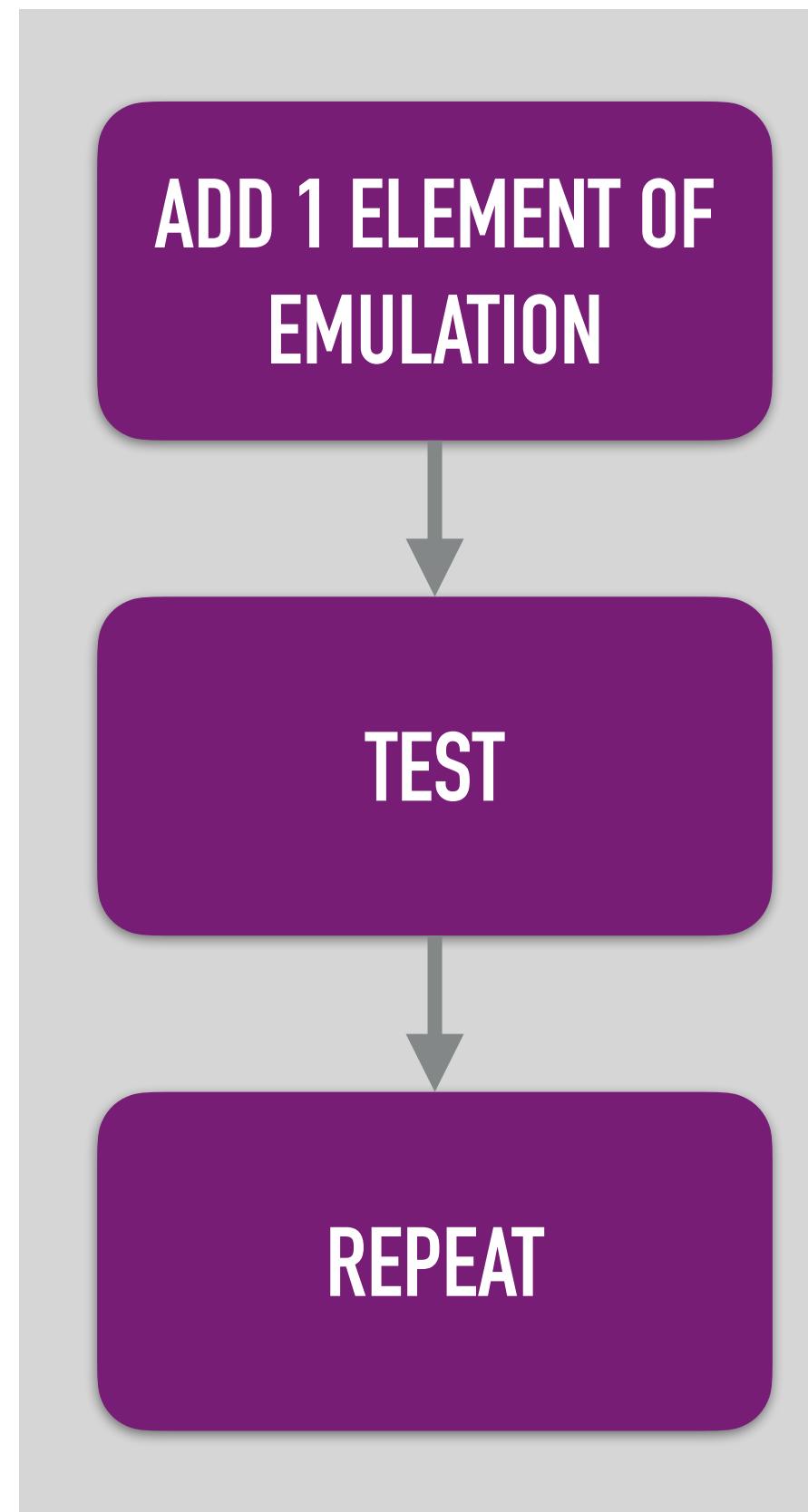
GET IT WORKING



A screenshot of a macOS Terminal window titled "Terminal". The window shows a sequence of commands being run in a directory named "AdversaryWorkshop". The commands include listing files, compiling a plugin, linking it with the main program, and finally executing the resulting binary. The output of the execution shows the plugin's message: "Hello from the plugin!". The terminal has its characteristic red, yellow, and green window control buttons at the top left.

```
batgirl:~/Code/AdversaryWorkshop⇒ ls
main.c    plugin.c
batgirl:~/Code/AdversaryWorkshop⇒ clang -shared -o plugin.so plugin.c
batgirl:~/Code/AdversaryWorkshop⇒ clang main.c
batgirl:~/Code/AdversaryWorkshop⇒ ./a.out
Calling plugin
Hello from the plugin!
batgirl:~/Code/AdversaryWorkshop⇒
```

BUILD THE SHARED MODULE



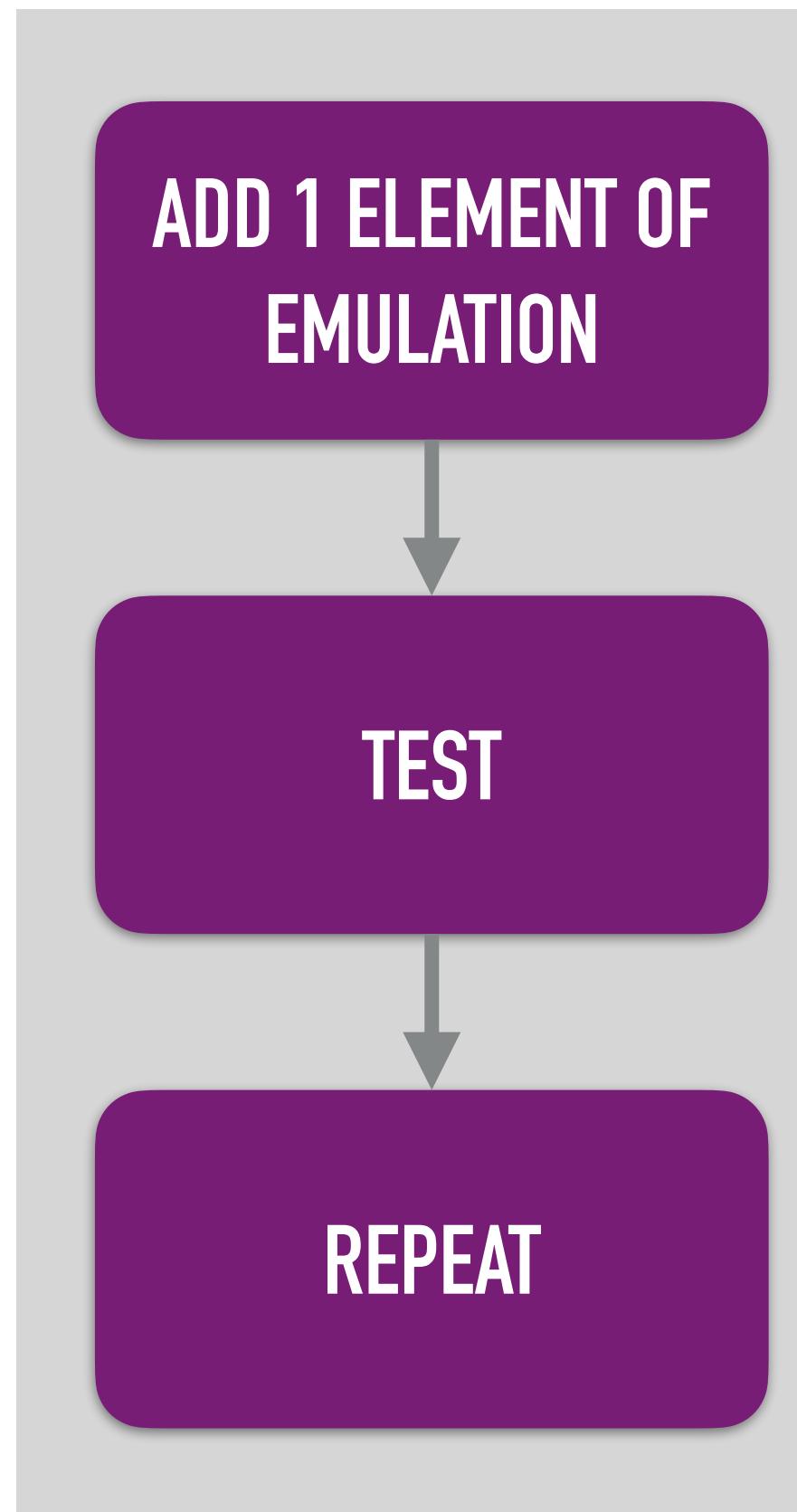
github.com/center-for-threat-informed-defense/

Code **Blame** **Raw** **Copy** **Download** **Edit** **Diff**

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 void create_dir() {
6     system("mkdir -p /tmp/.rota");
7 }
8
9 void copy_pdfs() {
10    system("find /home/ -name *.pdf -exec cp {} /tmp/.rota \\; 2>/dev/null");
11 }
12
13 void tar() {
14     system("cd /tmp/ && tar -czvf rota.tar.gz /tmp/.rota");
15 }
16
17 // this is loaded via dlsym.
18 // the argument to the rota_c2_run_plugin_1 command
19 extern void update(void) {
20     create_dir();
21     copy_pdfs();
22     sleep(10);
23 }
```

https://github.com/center-for-threat-informed-defense/adversary_emulation_library/blob/master/ocean_lotus/Resources/rota/src/c2_commands.c#L153

BUILD THE SHARED MODULE

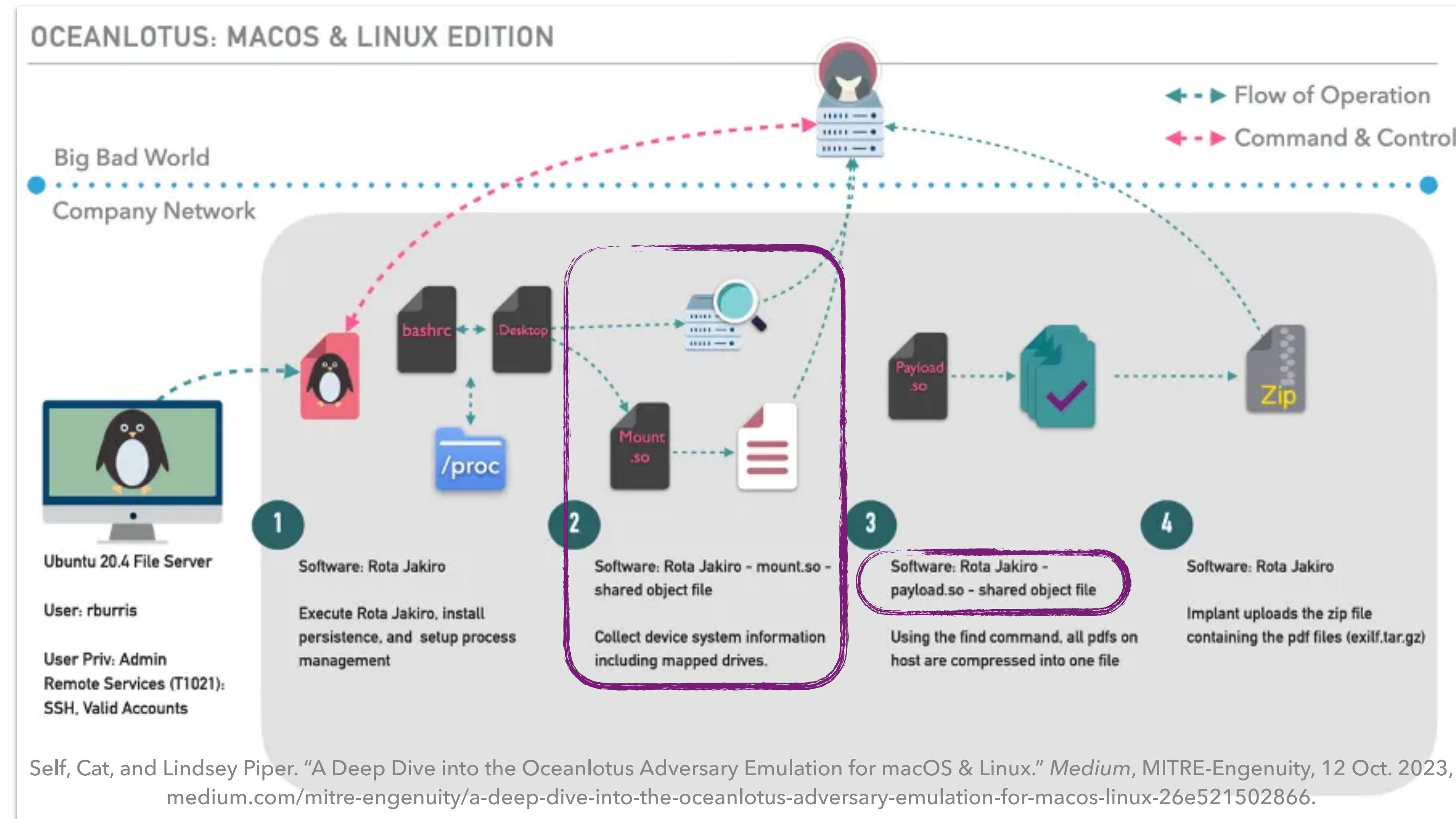


The screenshot shows a GitHub code editor interface. At the top, there are status indicators (red, yellow, green), a file navigation bar, and a search bar containing the URL "github.com/center-for-threat-i...". Below the header, it says "1 of 13 matches" and "Begins with". The main area displays a file named "c2_commands.c" under the "master" branch of the "adversary_emulation_library / ocean_lotus / Resources" directory. The "Code" tab is selected, showing the following C code:

```
145
146 void c2_run_plugin_1(char *funcName) {
147
148     // SO expected to be uploaded via rota_c2_upload_file()
149     char *soPath = "./local_rota_file.so";
150     void *handle = dlopen(soPath, RTLD_LAZY);
151     | void (*func_ptr)() = dlsym(handle, funcName);
152     // execution of shared object
153     func_ptr();
154
155 }
```

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

PROCEDURE FLOW



EXERCISE: CODE A PROCEDURE FOR MACOS

Build out a small program that executes the shared module technique.

Hint: Start by recreating the Linux example, then modify to use a dylib

Useful Resources

1. https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/ocean_lotus/Resources/OSX.OceanLotus
2. <https://unit42.paloaltonetworks.com/unit42-new-improved-macos-backdoor-oceanlotus/>
3. <https://www.welivesecurity.com/2019/04/09/oceanlotus-macos-malware-update/>

SAME EXACT STEPS AS LINUX EXCEPT...

The screenshot shows a VS Code interface with the following details:

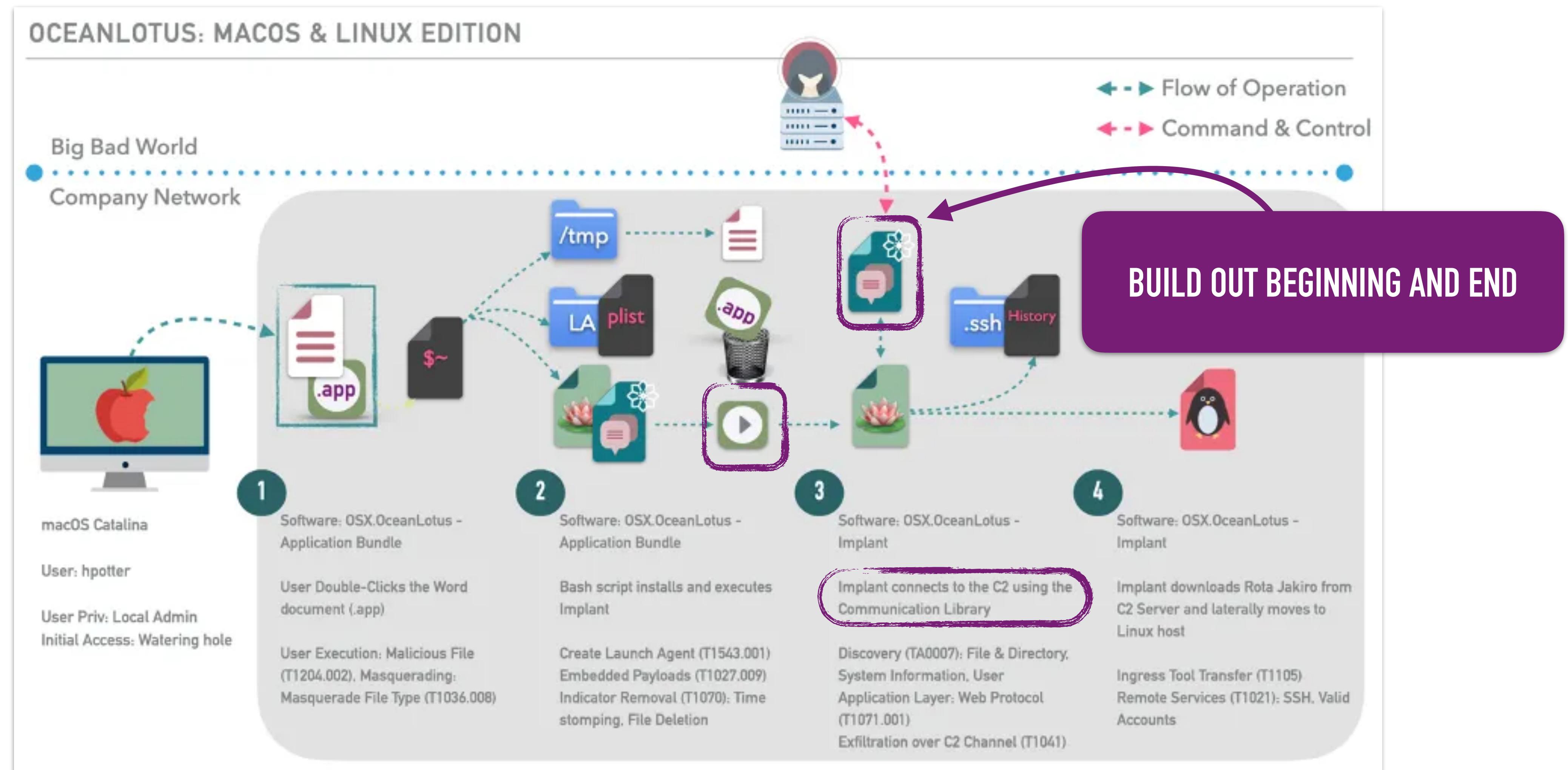
- EXPLORER**: Shows files: .vscode, AdversaryWorkshop, a.out, main.c, plugin.c, **plugin.dylib** (highlighted with a pink rectangle), plugin.h, attackJupyter, emojis, secret.
- CODE**: File **main.c** is open, showing the following code:

```
1 #include "plugin.h"
2
3 int main() {
4
5     // Open the needed object
6     void* handle = dlopen("plugin.dylib", RTLD_LAZY);
7     if (handle == NULL) {
8         fprintf(stderr, "Could not open plugin: %s\n", dlerror());
9         return 1;
10    }
11
12    // Return the pointer of the function to execute from teh sha
13    void (*functionPtr)() = dlsym(handle, "plugin_func");
14    if (functionPtr == NULL) {
15        fprintf(stderr, "Could not find plugin_func: %s\n", dlerror());
```
- A callout bubble with a purple border and arrow points from the line `void* handle = dlopen("plugin.dylib", RTLD_LAZY);` to the text **APPLE DOCS PROVIDES ADDITIONAL CODE EXAMPLES**.
- APPLE DOCS PROVIDES ADDITIONAL CODE EXAMPLES** is displayed in a purple callout bubble.
- Code** search bar is visible at the top right.
- Bottom right corner contains copyright and distribution information: ©2024 The MITRE Corporation. ALL RIGHTS RESERVED. Approved for public release. Distribution unlimited. PR_24-01985-1.

TEST IT IN YOUR MACOS VM

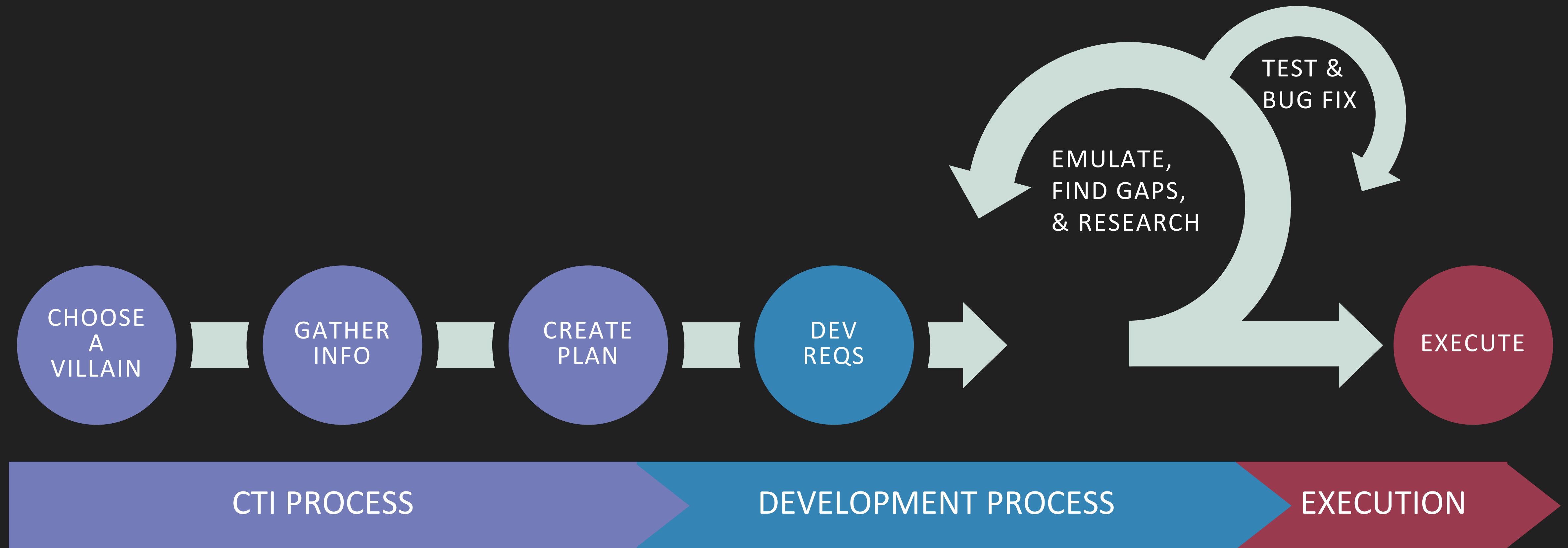
```
batgirl:~/Code/AdversaryWorkshop⇒ clang -dynamiclib plugin.c -o plugin.dylib  
batgirl:~/Code/AdversaryWorkshop⇒ clang main.c  
batgirl:~/Code/AdversaryWorkshop⇒ ./a.out  
Calling plugin  
Now you can put any function here  
batgirl:~/Code/AdversaryWorkshop⇒
```

MACOS PROCEDURE FLOW



Self, Cat, and Lindsey Piper. "A Deep Dive into the Oceanlotus Adversary Emulation for macOS & Linux." Medium, MITRE-Engenuity, 12 Oct. 2023, medium.com/mitre-engenuity/a-deep-dive-into-the-oceanlotus-adversary-emulation-for-macos-linux-26e521502866.

BECOMING A DARK KNIGHT



Cat [@coolestcatiknow](#)

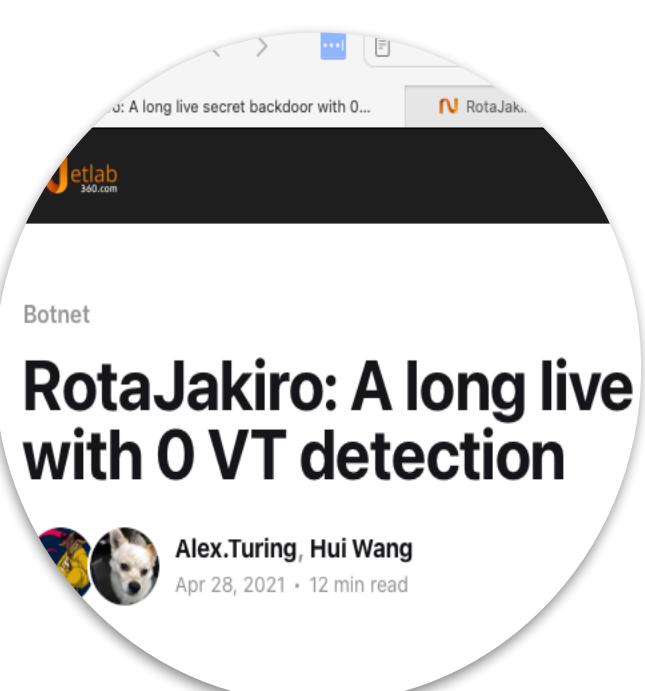
Kate [@phish4answers](#)

Approved for Public Release; Distribution Unlimited. Public Release Case Number PR_22-00490-11. © 2023. The MITRE Corporation. ALL RIGHTS RESERVED.

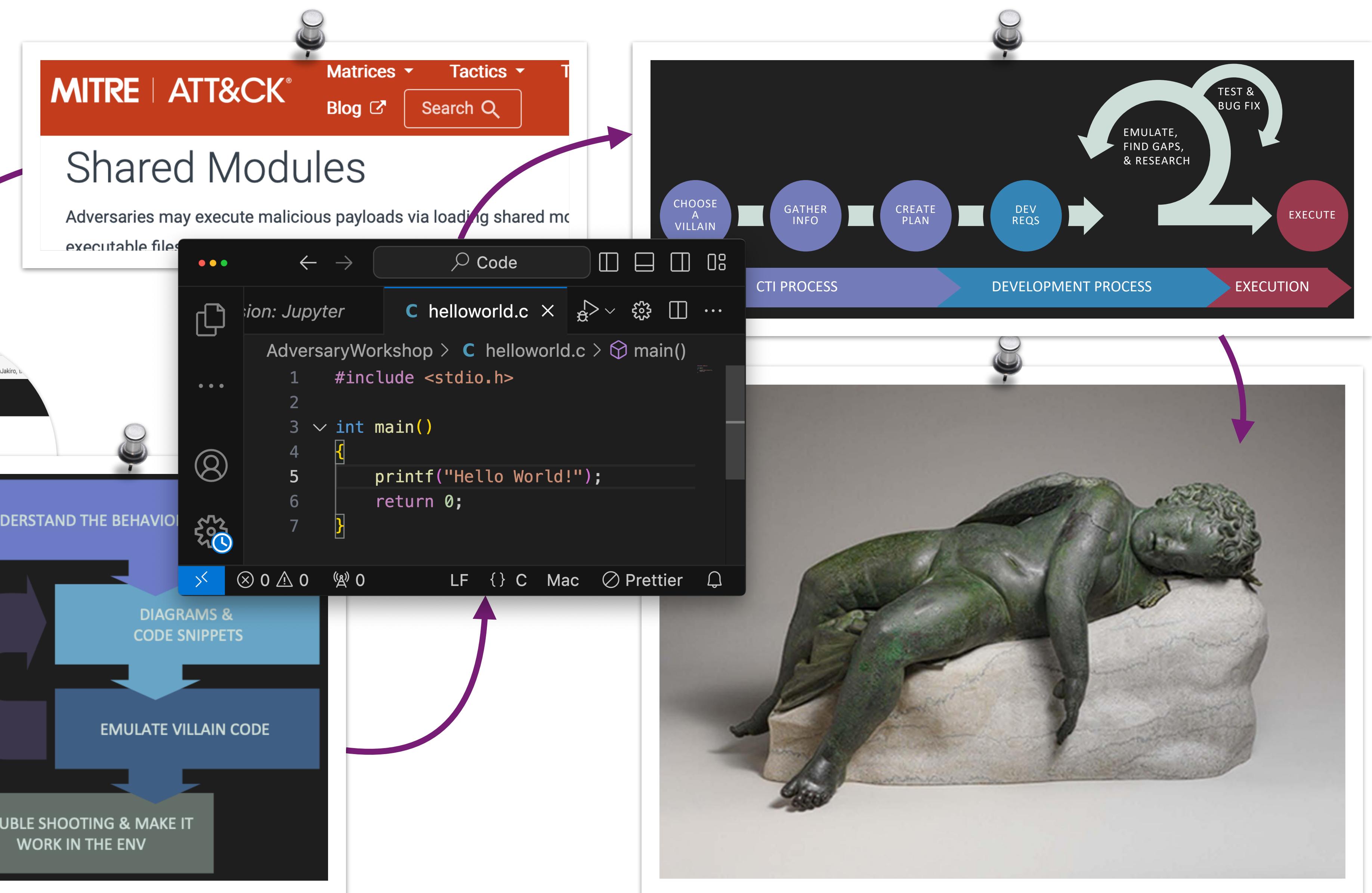


COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

WHAT WE COVERED



MITRE

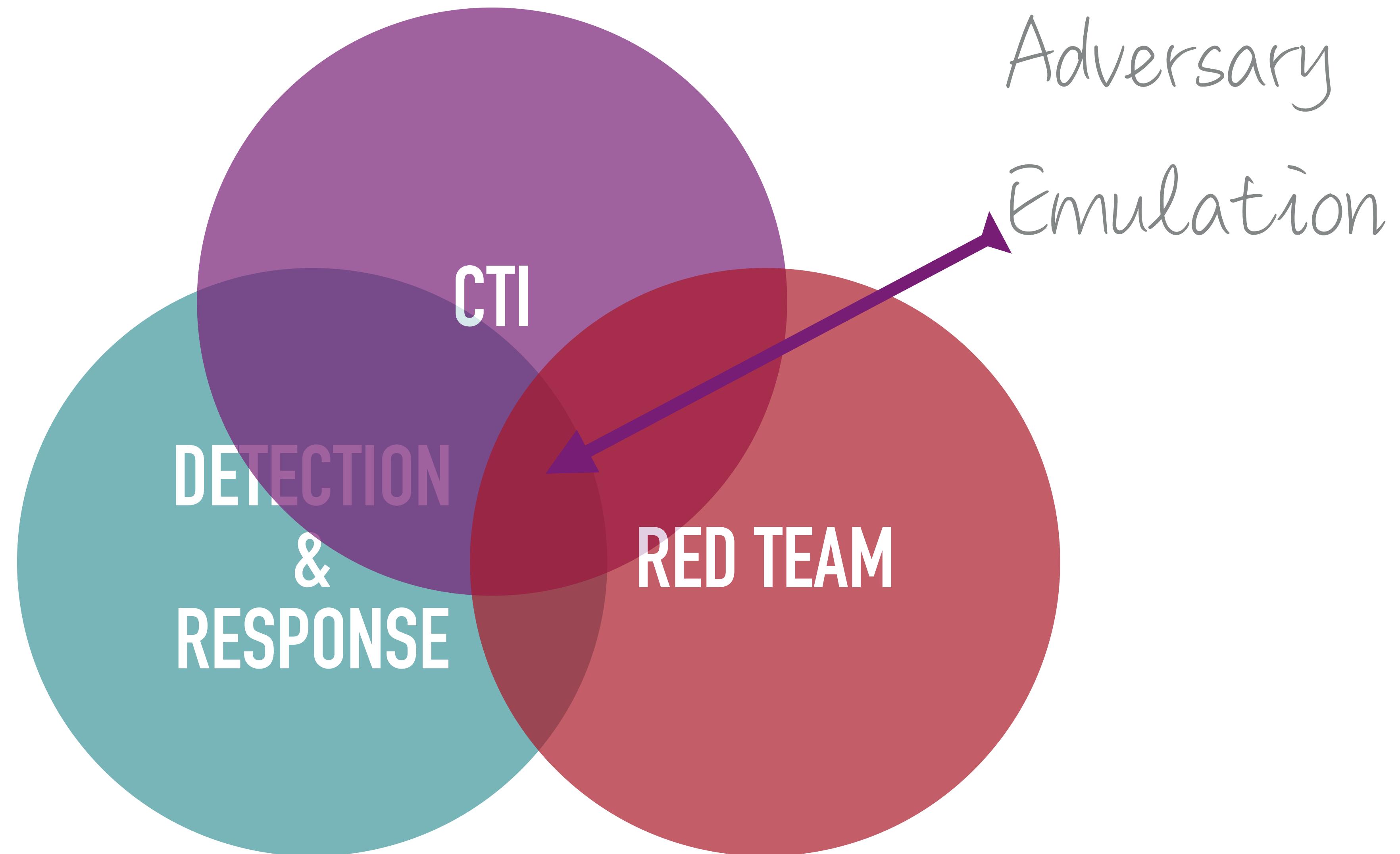


WHY USE AE?

People

Process

Technology



<https://www.ibm.com/downloads/cas/QEBYPND1>

COPY CAT: AN ARTIST GUIDE TO ADVERSARY FORGERY

SWAG

The screenshot shows a web browser window displaying a RedBubble shop page. The URL in the address bar is www.redbubble.com/people/mitreattack/shop. The page header includes the RedBubble logo, a search bar containing "Cyber punk mechanical anatomy posters", and links for "Sell your art", "Login", "Signup", a heart icon, and a shopping cart icon. Below the header is a navigation menu with categories: Explore, Clothing, Stickers, Phone Cases, Wall Art, Home & Living, Kids & Babies, Accessories, Stationery & Office, and Gifts. The main feature is a large, bold title "MITRE | ATT&CK®" in white on a red background, with a red ampersand symbol "&" positioned below it. Below the title, the text "MITRE ATT&CK" is displayed, along with "VA, United States • 21 designs • [View artist profile](#)". There are two buttons: "Shop products" (underlined) and "Explore designs". A "Share" button is located on the right side of the title area. The "Featured collection" section is titled "2024 MITRE ATT&CK Stickers" and includes five preview images of the stickers: "ADVERSARY IN-THE-MIDDLE", "FINANCIAL THEFT", "BRUTE FORCE", "STEAL WEB SESSION COOKIE", and "INDICATOR REMOVAL: Timestomp". A "Shop all >" link is located to the right of the preview images. At the bottom of the page, there is a small "MITRE" logo.

ATT&CK™



Cat Self

attack@mitre.org

@coolestcatiknow

[linkedin.com/in/coolestcatiknow](https://www.linkedin.com/in/coolestcatiknow)

www.catalystcode.org

