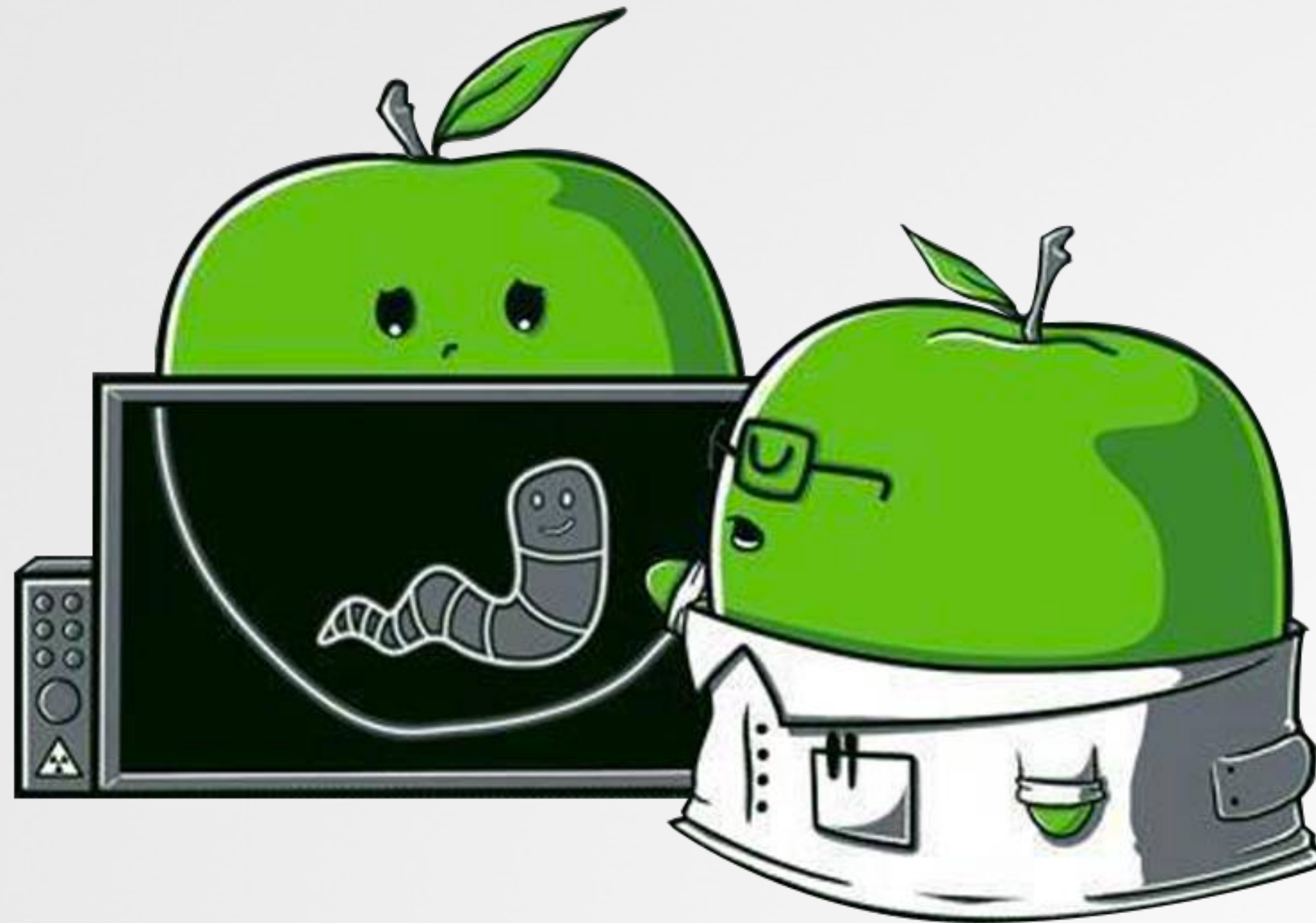


Made In America

Analyzing USA Spy Agencies' Mac Implants



WHOIS



RUNA SANDVIK



 (@RUNASAND)



PATRICK WARDLE

OBJECTIVE-SEE

 (@PATRICKWARDLE)

OUTLINE



...play along?



"Green Lambert"
(CIA)



"DoubleFantasy"
(NSA)

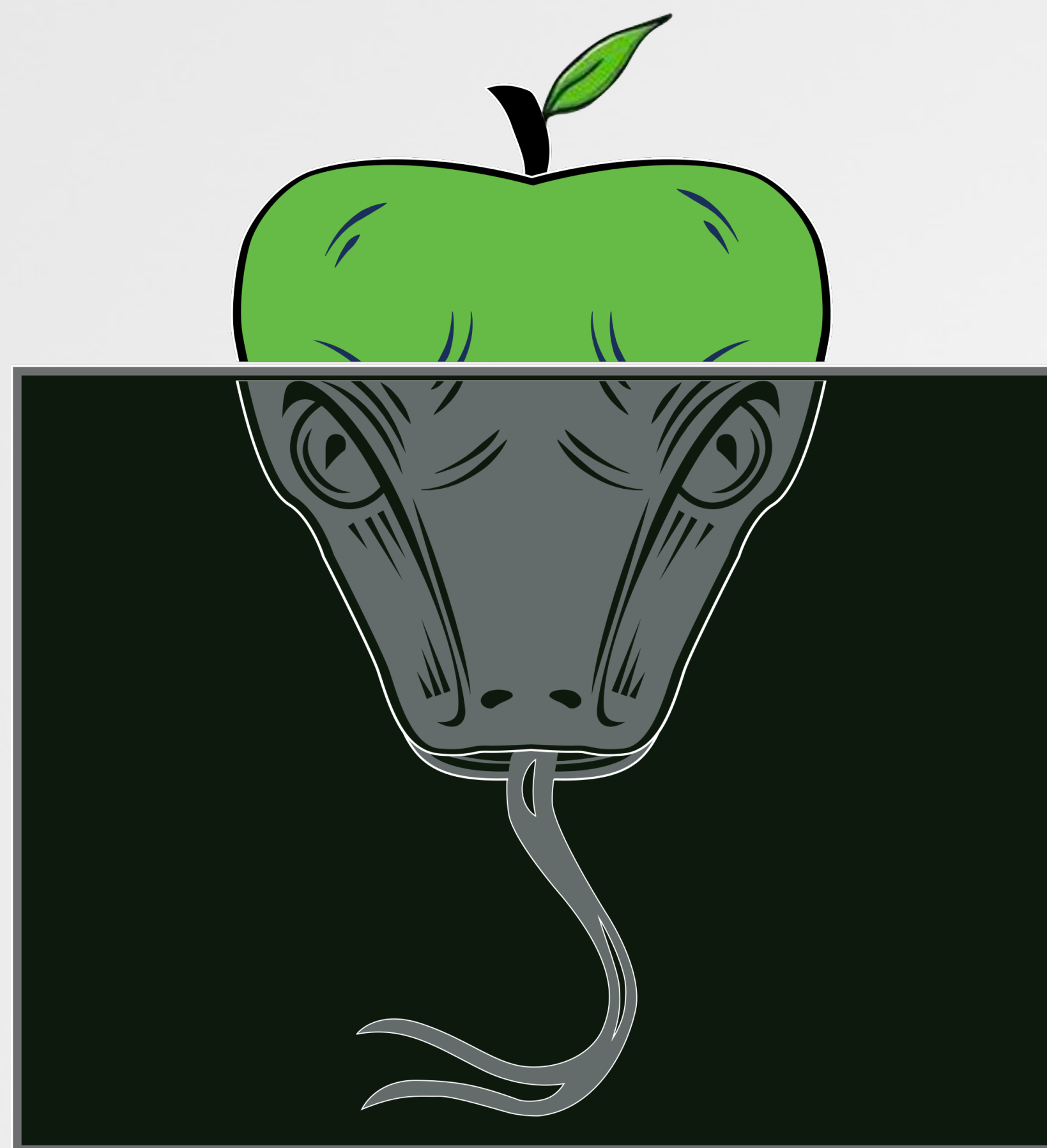
applicable to the analysis of other (macOS) samples!



Topics covered: macOS malware analysis via static & dynamic methods.

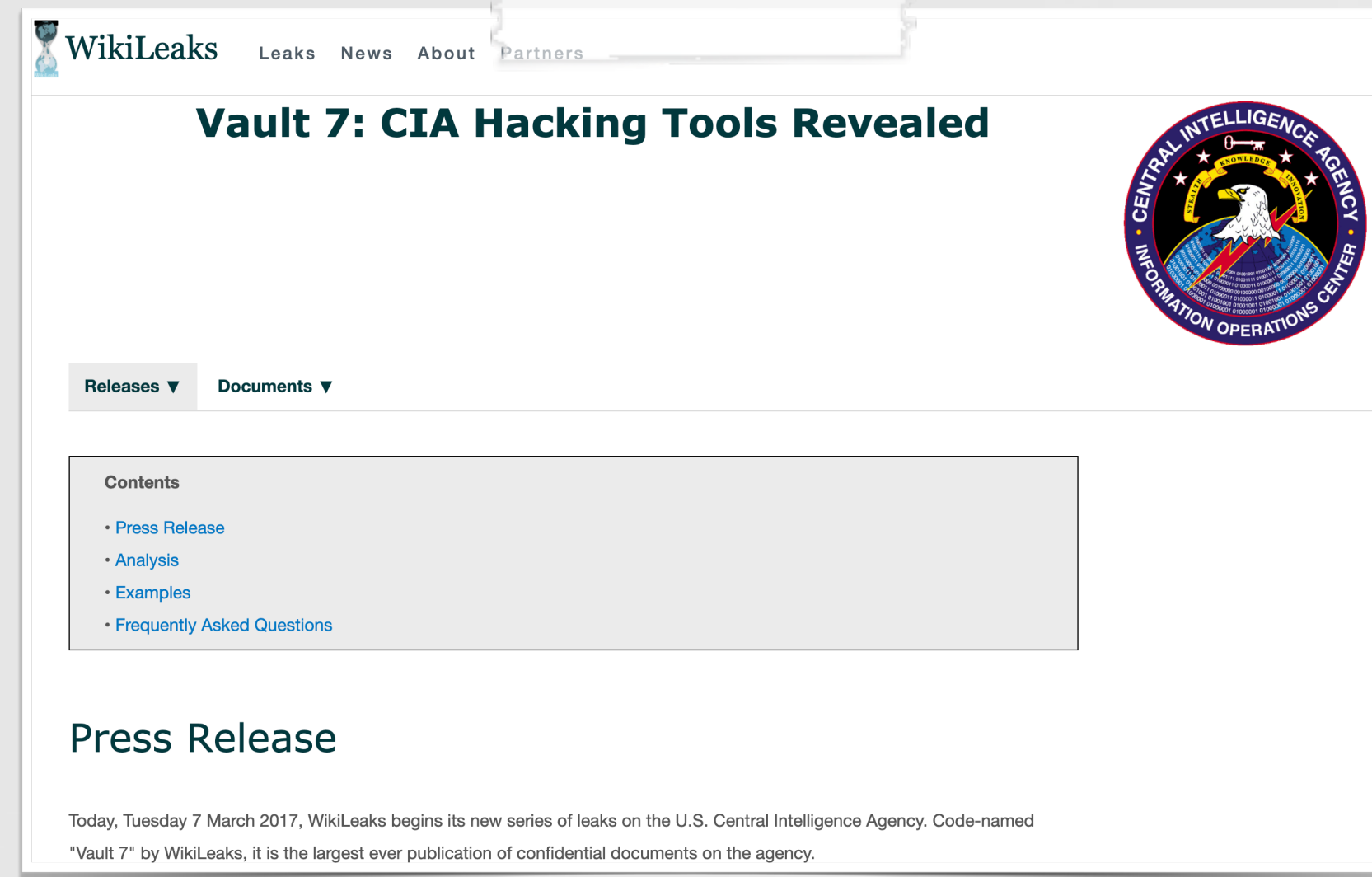
Green Lambert

...CIA...



INITIALLY DISCLOSED BY KASPERSKY

Following WikiLeaks: Vault 7



WikiLeaks Leaks News About Partners

Vault 7: CIA Hacking Tools Revealed

Releases Documents

Contents

- [Press Release](#)
- [Analysis](#)
- [Examples](#)
- [Frequently Asked Questions](#)

Press Release

Today, Tuesday 7 March 2017, WikiLeaks begins its new series of leaks on the U.S. Central Intelligence Agency. Code-named "Vault 7" by WikiLeaks, it is the largest ever publication of confidential documents on the agency.

March 7, 2017

Longhorn: Tools used by cyberespionage group linked to Vault 7



A L Johnson

04-10-2017 09:00 AM

Spying tools and operational protocols detailed in the recent Vault 7 leak have been used in cyberattacks against at least 40 targets in 16 different countries by a group Symantec calls Longhorn. Symantec has been protecting its customers from Longhorn's tools for the past three years and has continued to track the group in order to learn more about its tools, tactics, and procedures.

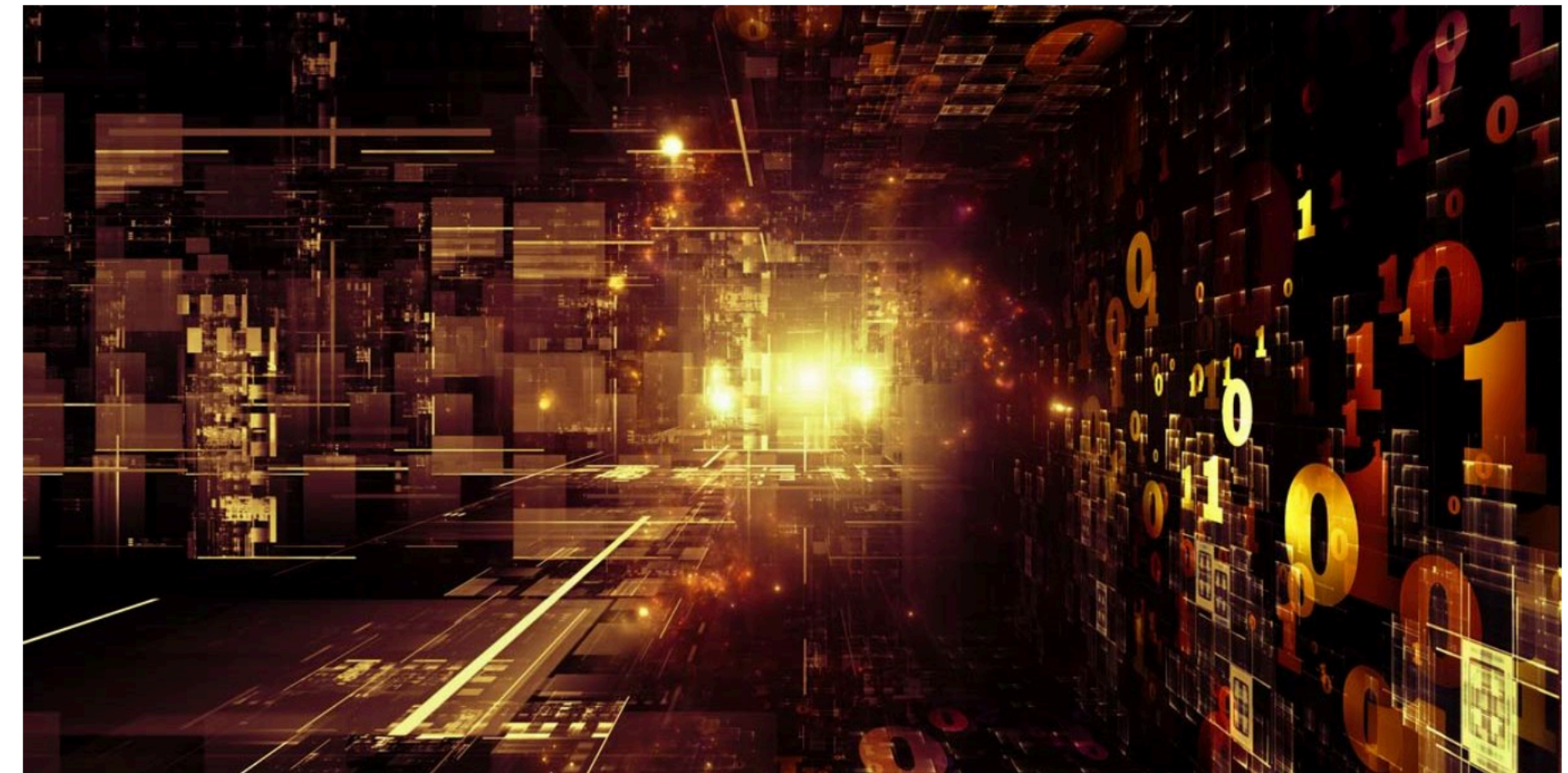
The tools used by Longhorn closely follow development timelines and technical specifications laid out in documents disclosed by WikiLeaks. The Longhorn group shares some of the same cryptographic protocols specified in the Vault 7 documents, in addition to following leaked guidelines on tactics to avoid detection. Given the close similarities between the tools and techniques, there can be little doubt that Longhorn's activities and the Vault 7 documents are the work of the same group.

April 10, 2017

Unraveling the Lamberts Toolkit

APT REPORTS 11 APR 2017

11 minute read



// AUTHORS

Expert GREAT

An Overview of a Color-coded Multi-Stage Arsenal

Yesterday, our colleagues from [Symantec published their analysis of Longhorn](#), an advanced threat actor that can be easily compared with Regin, ProjectSauron, Equation or Duqu2 in terms of its complexity.

Longhorn, which we internally refer to as "The Lamberts", first came to the attention of the ITSec community in 2014, when our colleagues from [FireEye discovered an attack using a zero day vulnerability \(CVE-2014-4148\)](#). The attack leveraged malware we called 'BlackLambert', which was used to target a high profile organization in Europe.

April 11, 2017

DEVELOPMENT TRADECRAFT

DOs and DON'Ts



Runa Sandvik
@runasand

In March 2017, Wikileaks published documents detailing the CIA's spying operations and hacking tools. I decided to dig into Development Tradecraft DOs and DON'Ts to see how the guidance changed over time. Here's the end result. 🕵️ #Vault7

A history of Development Tradecraft DOs and DON'Ts from...
Sheet1
Type, Directive, Rationale, Added, Removed, Modified, Classifi...
docs.google.com

11:48 AM · Aug 9, 2021 · Twitter Web App

	A	B	C	D	E	F	G
1	Type	Directive	Rationale	Added	Removed	Modified	Classification
2	Title	Development Tradecraft DOs and DON'Ts		Version 1		Version 8	Secret
3	Title	Draft Development Tradecraft DOs and DON'Ts		Version 8		Version 46	Secret
4	Title	Development Tradecraft DOs and DON'Ts		Version 46			Secret
5							
6	Classification	SECRET//NOFORN		Version 1		Version 48	Secret
7	Classification	TOP SECRET//NOFORN		Version 48		Version 49	Top Secret
8	Classification	SECRET//NOFORN		Version 49			Secret
9	(U) General			Version 51			
10	General	DO remove all data that demonstrates CIA, USG, or its witting partner companies involvement in the creation or use of the binary/tool/etc.	Attribution of binary/tool/etc by an adversary can cause irreversible impacts to past, present and future USG operations and equities.	Version 1		Version 10	Secret
11	General	DO NOT have data that demonstrates CIA, USG, or its witting partner companies involvement in the creation or use of the binary/tool/etc in the binary.	Attribution of binary/tool/etc by an adversary can cause irreversible impacts to past, present and future USG operations and equities.	Version 10		Version 34	Secret
12	General	DO NOT leave data in a binary file that demonstrates CIA, USG, or its witting partner companies involvement in the creation or use of the binary/tool.	Attribution of binary/tool/etc by an adversary can cause irreversible impacts to past, present and future USG operations and equities.	Version 34		Version 50	Secret
		(S//NF) DO NOT leave data in a	(S//NF) Attribution of binary/tool/etc by an adversary can cause				

Always interesting to see how people do what they do

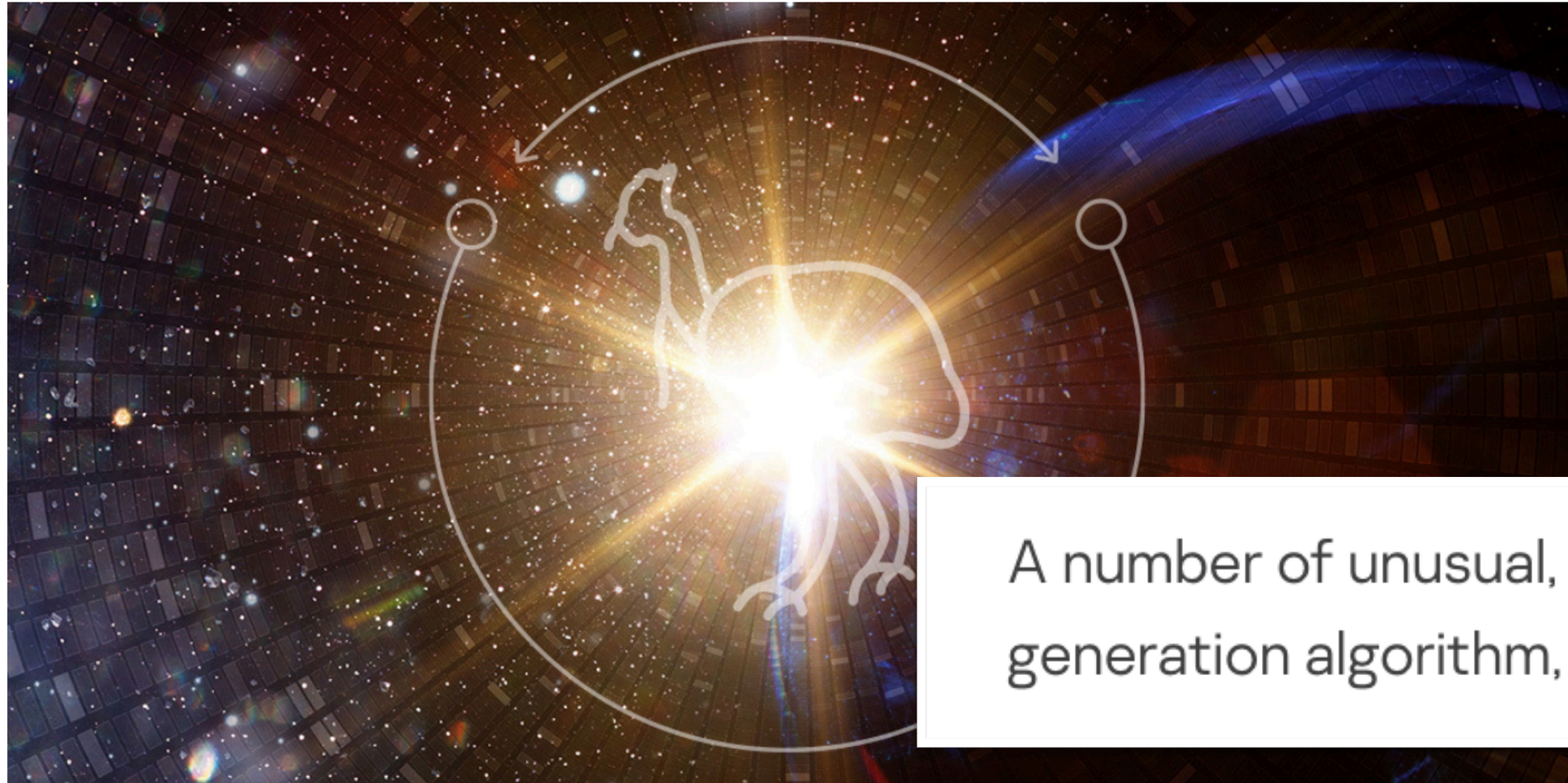
BUT WHY THO?

Sunburst backdoor – code overlaps with Kazuar

APT REPORTS

11 JAN 2021

⌚ 25 minute read



A number of unusual, shared features between Sunburst and Kazuar include the victim UID generation algorithm, the sleeping algorithm and the extensive usage of the FNV-1a hash.

// AUTHORS

Expert GEORGY KUCHERIN

Expert IGOR KUZNETSOV

COSTIN RAIU

Introduction

On December 13, 2020, FireEye [published a blog post](#) detailing a supply chain attack leveraging Orion IT, an infrastructure monitoring and management platform by SolarWinds. In parallel, Volexity [published an article](#) with their analysis of related attacks, attributed to an actor named “Dark Halo”. FireEye did not link this activity to any known actor; instead, they gave it an unknown, temporary moniker – “UNC2452”.

IT MATCHES VAULT 7

According to Symantec

Longhorn: Tools used by cyberespionage group linked to Vault 7

04-10-2017 09:00 AM



A L Johnson

Spying tools and operational protocols detailed in the recent Vault 7 leak have been used in cyberattacks against at least 40 targets in 16 different countries by a group Symantec calls Longhorn. Symantec has been protecting its customers from Longhorn's tools for the past three years and has continued to track the group in order to learn more about its tools, tactics, and procedures.

The tools used by Longhorn closely follow development timelines and technical specifications laid out in documents disclosed by WikiLeaks. The Longhorn group shares some of the same cryptographic protocols specified in the Vault 7 documents, in addition to following leaked guidelines on tactics to avoid detection. Given the close similarities between the tools and techniques, there can be little doubt that Longhorn's activities and the Vault 7 documents are the work of the same group.

Is this true for Green Lambert on OS X?

VICTIMOLOGY

The Lamberts

Who is Longhorn?

Longhorn has been active since at least 2011. It has used a range of back door Trojans in addition to zero-day vulnerabilities to compromise its targets. Longhorn has infiltrated governments and internationally operating organizations, in addition to targets in the financial, telecoms, energy, aerospace, information technology, education, and natural resources sectors. All of the organizations targeted would be of interest to a nation-state attacker.

Longhorn has infected 40 targets in at least 16 countries across the Middle East, Europe, Asia, and Africa. On one occasion a computer in the United States was compromised but, following infection, an uninstaller was launched within hours, which may indicate this victim was infected unintentionally.

Since at least 2008, The Lamberts have used multiple sophisticated attack tools against high-profile victims. Their arsenal includes network-driven backdoors, several generations of modular backdoors, harvesting tools, and wipers. Versions for both Windows and OSX are known at this time, with the latest samples created in 2016.

Kaspersky

The Red Raindrop team of Qi'anxin Threat Intelligence Center conducted research on historically exposed CIA cyber weapons and related materials, and discovered a variety of cyber weapons files, and based on the results of the analysis, they correlated and determined the contents of the existing public information. And we also found that these cyber weapons have been used to attack targeted personnel and institutions in China, and their related attacks mainly occurred from 2012 to 2017 (which coincides with the disclosure time of Vault7 data), and the related information was exposed until 2018. At the end of the year, some attacks were still maintained, and the target may involve the domestic aviation industry.

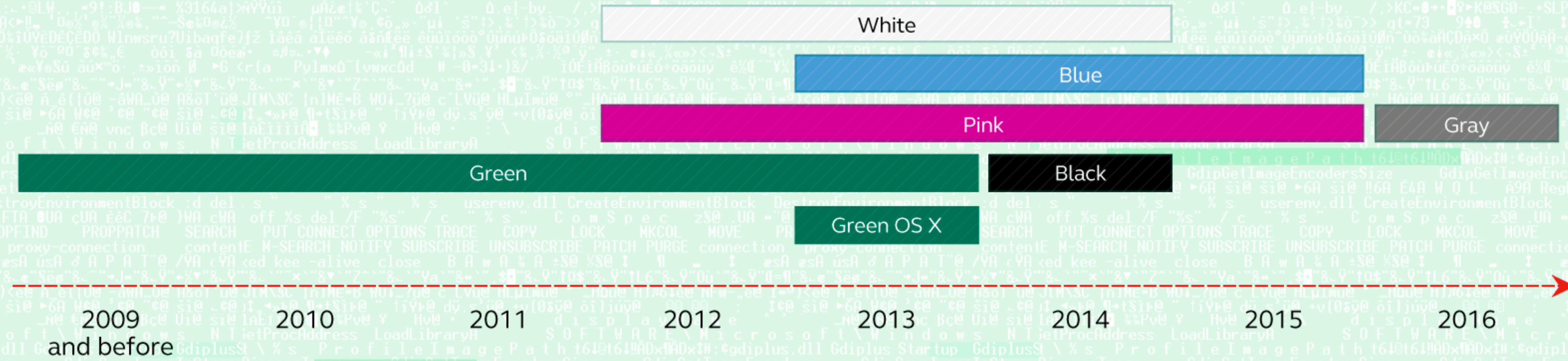
QI-ANXIN

Symantec

THE LAMBERT FAMILY MALWARE

A timeline of activity for known Lamberts

The Lamberts have used many sophisticated attack tools against high profile victims since at least 2008.



© 2017 Kaspersky Lab. All Rights Reserved.



Kaspersky: "Green Lambert is the oldest and longest-running... only one where non-Windows variants have been found."

ON VIRUSTOTAL

Since 2014

Date	Name	Source	Country
2014-09-03 12:16:53	GrowlHelper	8e375a22 - web	RU
2017-04-24 17:17:03	af7c395426649c57e44eac0bb6c6a109ac649763065ff5b2b23db71839bac655	99e28311 - community	SG
2017-04-24 17:22:46	af7c395426649c57e44eac0bb6c6a109ac649763065ff5b2b23db71839bac655	99e28311 - community	SG

...from Russia

2014: 'GrowlHelper'
(SHA-1: 3fcdbd3c5fa34fb8e8d58038fa1d1f13d37e8a4b)

Previous Analyses	Date order
2014-09-03T12:16:53	0 / 53
2015-07-31T00:09:45	0 / 56
2016-09-26T12:03:28	0 / 55
2016-10-27T19:44:56	1 / 54
2016-11-02T15:52:28	2 / 54

...detected in 2016

DETECTION DETAILS RELATIONS BEHAVIOR CONTENT

Security vendors' analysis on 2016-10-27T19:44:56

Kaspersky ! HEUR:Trojan.Multi.ColoredLambert.gen

...by Kaspersky
(and AegisLab)

INITIAL TRIAGE

What can we learn?

Growl

Growl is a notification system for OS X. Growl has been around since 2004, and was originally called Global Notifications Center. The name was changed to Growl (like the noise a dog makes) since we felt the name Notifications Center was too geeky. We were wrong about that haha.

2004 - 2020

note: few dependencies

```
% file GrowlHelper
GrowlHelper: Mach-O executable i386

% codesign -dvv GrowlHelper
GrowlHelper: code object is not signed at all

% du -h GrowlHelper
208K
```

file & code-signing info

```
% otool -L GrowlHelper
/System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation
/System/Library/Frameworks/CoreServices.framework/Versions/A/CoreServices
/System/Library/Frameworks/Security.framework/Versions/A/Security
/System/Library/Frameworks/SystemConfiguration.framework/Versions/A/SystemConfiguration
/usr/lib/libSystem.B.dylib
/usr/lib/libgcc_s.1.dylib
```

STRINGS

A few clues

```
% strings - GrowlHelper

LoginItem
LaunchAgent
/Library/LaunchDaemons

www.google.com
Error from libevent when adding event...
1.3a

_SecKeychainFindInternetPassword
_SecKeychainItemCopyAttributesAndData
_kSCPropNetProxiesHTTPProxy
_kSCPropNetProxiesProxyAutoConfigEnable
_kSCPropNetProxiesProxyAutoConfigURLString
```

embedded strings

---> Options for gaining persistence

---> Event notification library, used in
Tor, v. 1.3a released in Feb 2007

---> Auto-determines proxy settings

---> Xcode 2.2, released in Nov 2005

VM FOR A 32-BIT EXECUTABLE

Which version of OS X?

```
% nm GrowlHelper  
  
U _CFArrayAppendValue  
U _CFArrayCreateMutable  
U _CFArrayCreateMutableCopy  
U _CFArrayGetCount  
U _CFArrayGetValueAtIndex  
U _CFArrayRemoveValueAtIndex  
U _CFDictionaryCreate  
U _CFDictionaryGetValue  
U _CFGetTypeID  
U _CFNumberGetTypeID  
...
```

20	U _CFStringGetTypeID				
21	U _FSGetCatalogInfo	Maybe used to determine whether a file has changed (?)	Deprecated, macOS 10.0–10.8	https://developer.apple.com/documentation	
22	U _FSPathMakeRef		Deprecated, macOS 10.0–10.8	https://developer.apple.com/documentation	
23	U _FSSetCatalogInfo	Used to interact with files, folders, and volumes	Deprecated, macOS 10.0–10.8	https://developer.apple.com/documentation	
24	0003018c D _NXArgc	Standard Xcode / C symbol			
25	00030188 D _NXArgv	Standard Xcode / C symbol			
26	U _SCDynamicStoreCopyProxies	Returns the current internet proxy settings		https://developer.apple.com/documentation	
27	U _SecKeychainFindInternetPassword	Finds the first Internet password based on given attributes		https://developer.apple.com/documentation	
28	U _SecKeychainItemCopyAttributesAndData	Retrieves data stored in the given keychain item		https://developer.apple.com/documentation	
29	U _SecKeychainItemFreeAttributesAndData	Releases the memory used by the keychain		https://developer.apple.com/documentation	
30	U _SecKeychainItemFreeContent	Releases the memory used by the keychain		https://developer.apple.com/documentation	
31	U _SecKeychainSearchCopyNext	Finds the next keychain item matching the search criteria	Deprecated, macOS 10.0–10.7	https://developer.apple.com/documentation	
32	U _SecKeychainSearchCreateFromAttributes	Creates a search object	Deprecated, macOS 10.0–10.7	https://developer.apple.com/documentation	
33	U _SecKeychainSetUserInteractionAllowed	Enables or disables the user interface for the keychain	Deprecated, macOS 10.2–12.0	https://developer.apple.com/documentation	
34	U __DefaultRuneLocale				

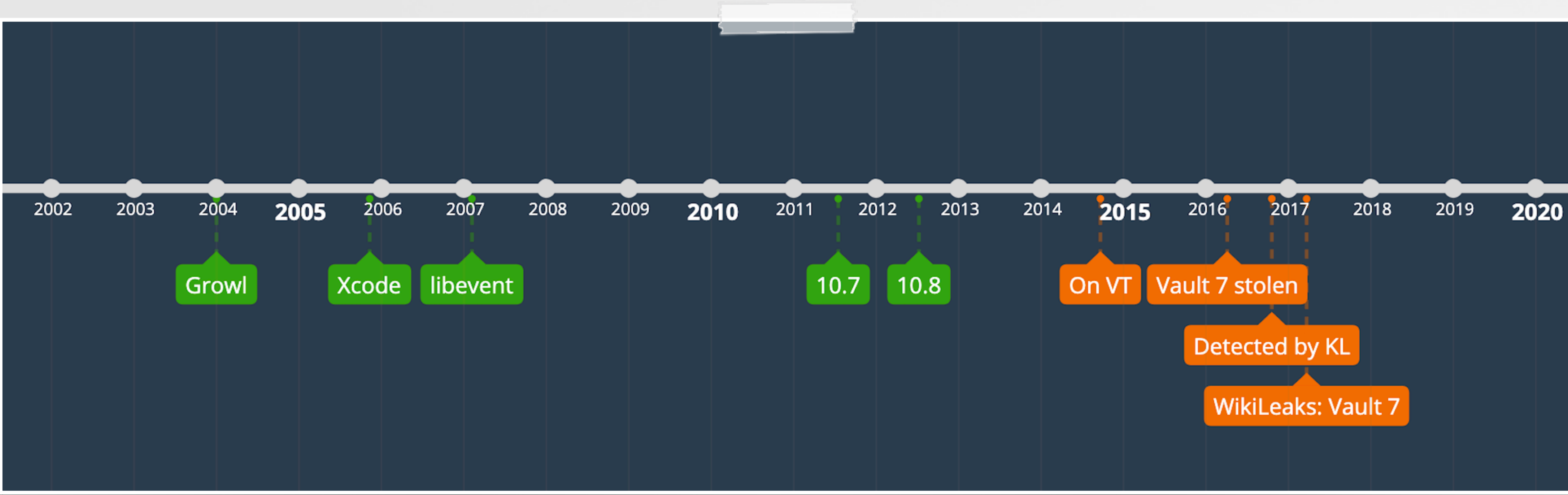
Google + Spreadsheet ❤️

Symbols supported in (at least) 10.7 - Lion
(though the sample will run in 10.8 as well)

View symbols with nm

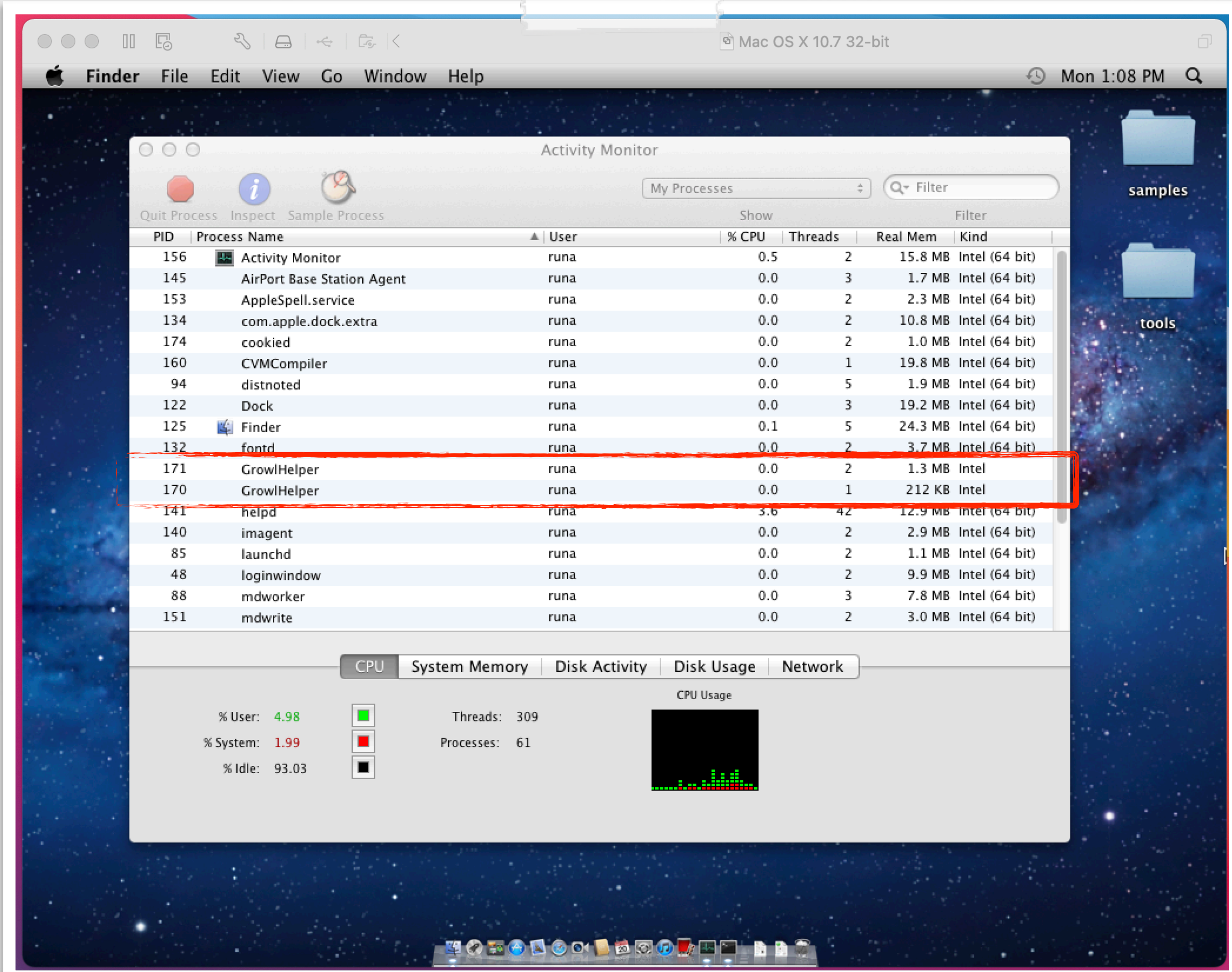
DEV/USE TIMELINE

2007 - 2013 (??)



Matches timeline of activity from Kaspersky

DOES IT RUN?



As of June 2021, OS X Lion is available for free from Apple

PERSISTENCE

Via LaunchAgent

```
% ls ~/Library/LaunchAgents  
com.apple.GrowlHelper.plist
```

Implant also self-deletes 🙄

```
% cat ~/Library/LaunchAgents/com.apple.GrowlHelper.plist  
  
<dict>  
  <key>Label</key>  
  <string>com.apple.GrowlHelper</string>  
  <key>ProgramArguments</key>  
  <array>  
<string>/Users/user/Library/Caches/  
com.apple.Growl.GrowlHelper/5d0d/GrowlHelper</string>  
  <string>-f</string>  
</array>  
  <key>RunAtLoad</key>  
  <true/>  
  <key>OnDemand</key>  
  <false/>  
</dict>
```

FILESYSTEM USAGE

```
fs_usage -w -f filesys
```

```
execve    /Users/user/GrowlHelper 0.015273 W bash.2848
execve    /Users/user/GrowlHelper 0.000383  GrowlHelper.2851

open     /Users/user/.profile 0.000018  GrowlHelper.2851
open     /Users/user/.bash_profile 0.000015  GrowlHelper.2851
open     /Users/user/.bash_login 0.000015  GrowlHelper.2851
open     /Users/user/.bashrc 0.000014  GrowlHelper.2851
open     /Users/user/.cshrc 0.000014  GrowlHelper.2851
open     /Users/user/.login 0.000014  GrowlHelper.2851
open     /Users/user/.tcshrc 0.000014  GrowlHelper.2851
open     /Users/user/.xsession 0.000007  GrowlHelper.2851
open     /Users/user/.xinitrc 0.000006  GrowlHelper.2851
```

MORE PERSISTENCE

Via `.profile`

```
% cat ~/.profile
```

```
GrowlHelper=`/path/to/5d0d/GrowlHelper 2>&1` # Automatic GrowlHelper. Do not remove
```

SELF-UPDATE

Online v. Offline

```
% file /Users/online/Library/Caches/com.apple.Growl.GrowlHelper/5d0d/*  
GrowlHelper: Mach-O executable i386  
Software Update Check: Mach-O executable i386  
db:          Berkeley DB 1.85 (Hash, version 2, native byte-order)  
fifo:        socket  
queue:       directory
```

```
% file /Users/offline/Library/Caches/com.apple.Growl.GrowlHelper/5d0d/*  
GrowlHelper: Mach-O executable i386  
db:          Berkeley DB 1.85 (Hash, version 2, native byte-order)  
fifo:        socket  
queue:       directory
```

NEW BINARY?!

Software Update Check

```
% file /Users/online/Library/Caches/com.apple.Growl.GrowlHelper/5d0d/*  
GrowlHelper: Mach-O executable i386  
Software Update Check: Mach-O executable i386  
db:          Berkeley DB 1.85 (Hash, version 2, native byte-order)  
fifo:        socket  
queue:       directory
```

3fcdbd3c5fa34fb8e8d58038fa1d1f13d37e8a4b GrowlHelper

3fcdbd3c5fa34fb8e8d58038fa1d1f13d37e8a4b Software Update Check

Theory: GrowlHelper drops a copy of itself to check for updates

FINDING CMDLINE ARGS

With try/fail*

Value	Meaning	Action
c	??	"** Commands will be processed immediately **"
d	??	If installed, drops copy "Software Update Check"
f	Default	LaunchAgent, creates: GrowlHelper, db, fifo, queue
p:	??	"GrowlHelper: option requires an argument -- p"
s	??	Runs without persisting, creates: db, fifo, queue
L	??	Runs without persisting, does not create files
N	??	LaunchAgent, creates: GrowlHelper, Software.., db

*** in an isolated, offline virtual machine**

FINDING CMDLINE ARGS

With Hopper

```
loc_956f:
0000956f    mov     dword [esp+0xf8+var_F0], esi           ; argument "optstring" for method imp__jump_table__getopt, CO
00009573    mov     ecx, dword [ebp+var_A8]
00009579    mov     dword [esp+0xf8+var_F4], ecx           ; argument "argv" for method imp__jump_table__getopt
0000957d    mov     eax, dword [ebp+var_A4]
00009583    mov     dword [esp+0xf8+var_F8], eax           ; argument "argc" for method imp__jump_table__getopt
00009586    call   imp__jump_table__getopt                 ; getopt
0000958b    cmp     eax, 0xffffffff
0000958e    jne    loc_9423
```

Look for argc, argv, getopt

```
loc_956f:
    eax = getopt(var_A4, var_A8, "cdefLnNp:rRs");
    if (eax != 0xffffffff) goto loc_9423;
```

Pseudo-code mode

ENTRY POINTS

From QI-ANXIN

Function name	Function
InitFunc_0	Get version information
InitFunc_1	Write ConfigInitdFile through /etc/init.d and /etc/rc.d to maintain persistence
InitFunc_2	Maintain persistence by writing configuration files of multiple shells
InitFunc_3	Maintain persistence by writing to XSession related configuration files
InitFunc_4	Parse network proxy from proxy URL
InitFunc_5	URL related resolution
InitFunc_6	Constant assignment
InitFunc_7	Generate UUID
InitFunc_8	Get proxy configuration from system environment variables
InitFunc_9	HTTP communication function initialization
InitFunc_10	HTTP communication interface function
InitFunc_11	HTTP proxy function initialization
InitFunc_12	Local loopback interface processing
InitFunc_13	TCP communication function initialization
InitFunc_14	Key chain access to realize login access of HTTP protocol
InitFunc_15	API to obtain system proxy configuration
InitFunc_16	Use LoginItem to maintain persistence
InitFunc_17	Use StartupItems to maintain persistence
InitFunc_18	Use LaunchAgent to maintain persistence
InitFunc_19	Get the configuration file in the home path to get the proxy configuration
InitFunc_20	SSL communication function initialization

FILESYSTEM USAGE

Keep what's needed

```
mkdir    /Users/user/.DS_Info          0.000083  GrowlHelper.2851
mkdir    /Users/user/.DS_Info/5d0d     0.000044  GrowlHelper.2851
mkdir    /Users/User/Library/Caches/com.apple.advanced 0.000066  GrowlHelper.2851

rmdir    /Users/user/.DS_Info/5d0d     0.000109  W GrowlHelper.2851
rmdir    /Users/user/.DS_Info          0.000240  W GrowlHelper.2851
rmdir    /Users/User/Library/Caches/com.apple.advanced 0.000068  GrowlHelper.2851
```

DECRYPTING A STRING

With Hopper + lldb

```
loc_15478:
00015478    call    sub_5f05                ; sub_5f05, CO
0001547d    mov     esi, eax
0001547f    test   eax, eax
00015481    lea    eax, dword [ebx-0x12534+aNxb3x9bx87xe0z+15] ; 0x2d313
00015487    cmove  esi, eax
0001548a    mov     ecx, 0x1
0001548f    lea    edx, dword [ebx-0x12534+dword_31e6c+20]    ; 0x31e80
00015495    lea    eax, dword [ebx-0x12534+aTx07rtxd9x927x+14] ; 0x2d487
0001549b    call   sub_f43a                ; sub_f43a
000154a0    mov     dword [esp+0x4c8+var_4B8], eax
```

Hopper has done the heavy lifting and figured out ecx, edx, eax for us 🙏

```
Current executable set to 'GrowlHelper' (i386).
(lldb) process launch --stop-at-entry
Process 173 launched: '/Users/runa/Desktop/samples/GrowlHelper' (i386)
Process 173 stopped
* thread #1: tid = 0x1f03, 0x8fe01030 dyld`_dyld_start, stop reason = signal SIGSTOP
   frame #0: 0x8fe01030 dyld`_dyld_start
dyld`_dyld_start:
-> 0x8fe01030:  pushl  $0
   0x8fe01032:  movl   %esp, %ebp
   0x8fe01034:  andl   $-16, %esp
   0x8fe01037:  subl   $12, %esp
(lldb) reg write pc 0x1549b
(lldb) reg write eax 0x2d487
(lldb) reg write edx 0x31e80
(lldb) reg write ecx 0x1
(lldb) b 0x154a0
breakpoint set --address 0x154a0
Breakpoint created: 1: address = 0x000154a0, locations = 1, resolved = 1
(lldb) c
Process 173 resuming
Process 173 stopped
* thread #1: tid = 0x1f03, 0x000154a0, stop reason = breakpoint 1.1
   frame #0: 0x000154a0
-> 0x154a0:  movl   %eax, 16(%esp)
   0x154a4:  movl   %esi, 12(%esp)
   0x154a8:  leal   108214(%ebx), %eax
   0x154ae:  movl   %eax, 8(%esp)
(lldb) reg read eax
eax = 0x00031e80
(lldb) mem read 0x31e80
0x00031e80:  68 76 65 72 73 69 6f 6e 2e 74 78 74 00 00 00 00  hversion.txt....
0x00031e90:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
(lldb) □
```

DECRYPTING MORE STRINGS!

This is where I got lost

/tmp	ConfigInitdFile	index.html
upload_dir	.xinitrc	hps.txt
upload_key	.xsession	hversion.txt
upload_header	ConfigPersistXsession	/bin/sh -c
	ConfigPersistXInitRC	
download		NODELETE
InternetOpen	proxy_type	DELETE
	proxy_url	SECDELETE
login.php	ProxyUser	WAIT
getconf.php	<u>http://www.google.com</u>	WAIT_FOREVER
show.php		



Appears to handle encrypted strings in 10+ different ways

LISTENING POST?

CIA + NSA words

"No LP Configured"

-----> Listening Post?



Military term for SIGINT/reconnaissance, also used by NSA



A Vault 7 document covers *Listening Post (LP) Creation*

CONFIG FILES?

html, php, txt

```
login.php  
getconf.php  
show.php  
index.html  
hps.txt  
hversion.txt
```

--- ► Can't access the files 🙄



Kaspersky: BL and GL samples have "two C&C servers hardcoded in their configuration block: a hostname and an IP address"



QI-ANXIN: Talks to Listening Post through login.php and getconf.php, downloads follow-up code through getfile.php

CONFIG? SURVEY?

A string equals...

```
loc_132a0:
000132a0      mov     ecx, 0x1                ; CODE XREF=dword_12d4c+601
000132a5      lea    edx, dword [ebx-0x12534+dword_31e6c+20] ; 0x31e80
000132ab      lea    eax, dword [ebx-0x12534+aX04fxe2fkx81xa+9] ; 0x2d42a
                                ; Version
000132b1      call   decrypt_string_sub_f43a ; decrypt_string_sub_f43a
000132b6      mov     ecx, eax
000132b8      lea    esi, dword [ebx-0x12534+asc_2e2f0] ; "="
000132be      test   eax, eax
000132c0      jne    loc_132cc
```

References to 0x2e2f0

Q Search

Address	Value
0x12593 (sub_12523 + 0x70)	lea eax, dword [ebx-0x12534+asc_2e2f0]
0x12913 (dword_1260c + 0x307)	lea eax, dword [ebx-0x12534+asc_2e2f0]
0x12ce6 (dword_1297c + 0x36a)	lea eax, dword [ebx-0x12534+asc_2e2f0]
0x12fd8 (dword_12d4c + 0x28c)	lea ecx, dword [ebx-0x12534+asc_2e2f0]
0x132b8 (dword_13068 + 0x250)	lea esi, dword [ebx-0x12534+asc_2e2f0]
0x13626 (dword_13358 + 0x2ce)	lea edx, dword [ebx-0x12534+asc_2e2f0]
0x1ee36 (sub_1edb0 + 0x86)	lea ecx, dword [ebx-0x1edbe+asc_2e2f0]

Cancel Go

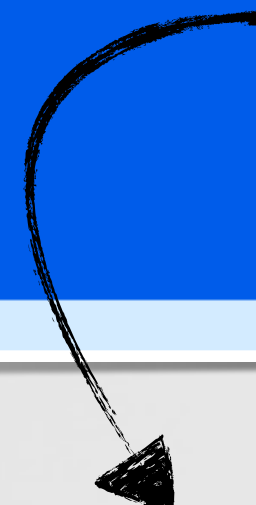


uname=
Time=%Y\%m\%d %H:%M:%S Z
Uptime=
Version=1.2.0
PID=

NETWORK TRAFFIC

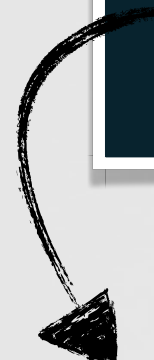
tcpdump + Wireshark

```
DNS 82 Standard query 0x7bd8 A notify.growlupdate.com
DNS 150 Standard query response 0x7bd8 No such name A notify.growlupdate.com SOA ns59.domaincontrol.com
DNS 87 Standard query 0x1e03 A notify.growlupdate.com.home
DNS 126 Standard query response 0x1e03 No such name A notify.growlupdate.com.home SOA home
DNS 76 Standard query 0xad14 A swscan.apple.com
```



Looks like a hostname! 🐱

Destination	Protocol	Length	Info
94.242.252.68	TCP	78	49307 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=405308294 TSecr=0 SACK_PERM=1
94.242.252.68	TCP	78	[TCP Retransmission] 49307 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=405309368 TSecr=0 SACK_PERM=1
94.242.252.68	TCP	78	[TCP Retransmission] 49307 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=405310466 TSecr=0 SACK_PERM=1
94.242.252.68	TCP	78	[TCP Retransmission] 49307 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=405311470 TSecr=0 SACK_PERM=1
94.242.252.68	TCP	78	[TCP Retransmission] 49307 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=405312471 TSecr=0 SACK_PERM=1
94.242.252.68	TCP	78	[TCP Retransmission] 49307 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=405313474 TSecr=0 SACK_PERM=1



And the IP address 🎉

LP: HOSTNAME

What can we learn?

"notify.growlupdate.com"

1 result (0.22 seconds)

It looks like there aren't many great matches for your search

Tip: Try using words that might appear on the page you're looking for. For example, "cake recipes" instead of "how to make a cake."



Good suggestion during a pandemic

INTERNET ARCHIVE Explore more than 616 billion web pages

WayBackMachine

notify.growlupdate.com

Hrm.

Wayback Machine has not archived that URL. Click here to search for all archived pages under <http://notify.growlupdate.com/>

DETECTION DETAILS COMMUNITY

Categories (i)

Websense ThreatSeeker unategorized

History (i)

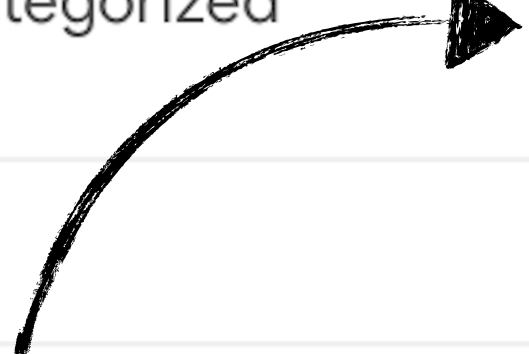
First Submission	2016-10-27 19:11:22
Last Submission	2017-03-09 02:13:03
Last Analysis	2017-03-09 02:13:03

HTTP Response (i)

Final URL

<http://notify.growlupdate.com/>

Prob KL :)



LP: HOSTNAME

crt.sh

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2121810481130736 (0x789c680022cf0)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: (CA ID: 24)
      serialNumber          = 07969287
      commonName            = Go Daddy Secure Certification Authority
      organizationalUnitName = http://certificates.godaddy.com/repository
      organizationName      = GoDaddy.com, Inc.
      localityName          = Scottsdale
      stateOrProvinceName  = Arizona
      countryName           = US
    Validity (Expired)
      Not Before: Oct 29 14:03:03 2013 GMT
      Not After : Oct 29 14:03:03 2014 GMT
    Subject:
      commonName            = notify.growlupdate.com
      organizationalUnitName = Domain Control Validated
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:c0:05:20:e5:de:ce:d8:e2:80:93:3e:92:82:e0:
        0d:76:49:1c:4a:df:9e:ce:18:85:aa:d6:bf:08:23:
        81:fb:25:ac:f6:fa:4a:a1:31:a5:bc:d2:60:70:3b:

```

Matches KL timeline :)

Created on Oct 29 2013

SINKHOLE

 to Kaspersky

This is a research system operated by **Kaspersky Lab**

For more information please visit:

www.kaspersky.com

www.securelist.com

For abuse please contact: theflame [AT] kaspersky [dot] com



Sinkhole: 95.211.172.143

2016-10-01 to 2017-10-02

DEVELOPMENT TRADECRAFT

DOs and DON'Ts



A screenshot of a Twitter post by Costin Raiu (@craiu). The post is a reply to @runasand and @dragon199421. The text discusses malware development guidelines: C2 jitter, secure erase / uninstall, SSL/TLS+extra crypto, size below 150K, encrypt logs and local collection, decrypt strings on the fly in mem... simply following these guidelines immediately makes the malware ("tools") more interesting and recognizable by a skilled analyst. The post is timestamped 2:11 AM · Aug 10, 2021 · Twitter Web App.

Costin Raiu ✓
@craiu

Replying to [@runasand](#) and [@dragon199421](#)

C2 jitter, secure erase / uninstall, SSL/TLS+extra crypto, size below 150K, encrypt logs and local collection, decrypt strings on the fly in mem... simply following these guidelines immediately makes the malware ("tools") more interesting and, recognizable by a skilled analyst.

2:11 AM · Aug 10, 2021 · Twitter Web App

File size over "ideal binary file size"

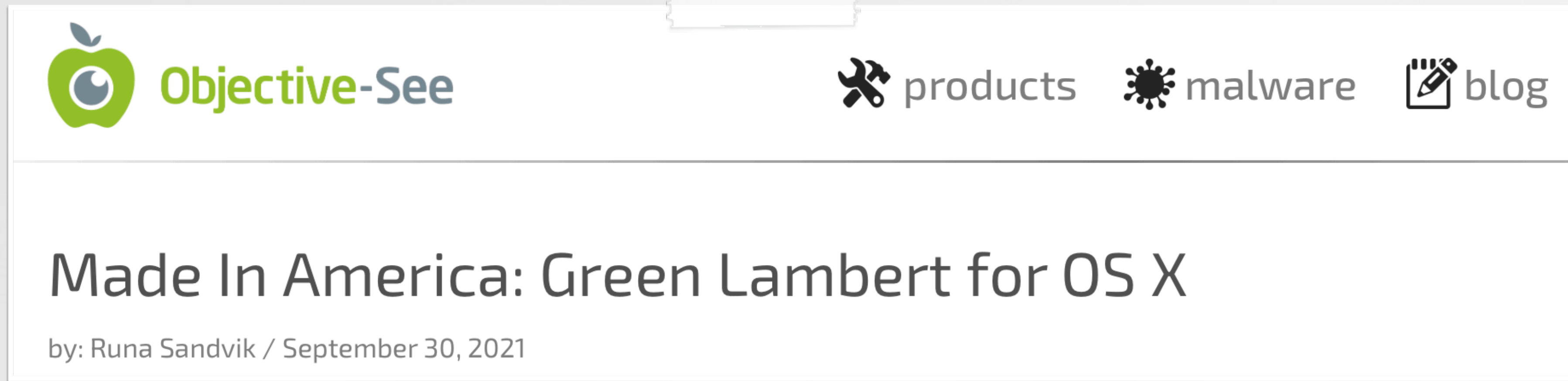
LP may be a CIA / USG specific term

Use of mtwhfsu / MTWHFSU

Use of libevent before it was cool

AND MORE !

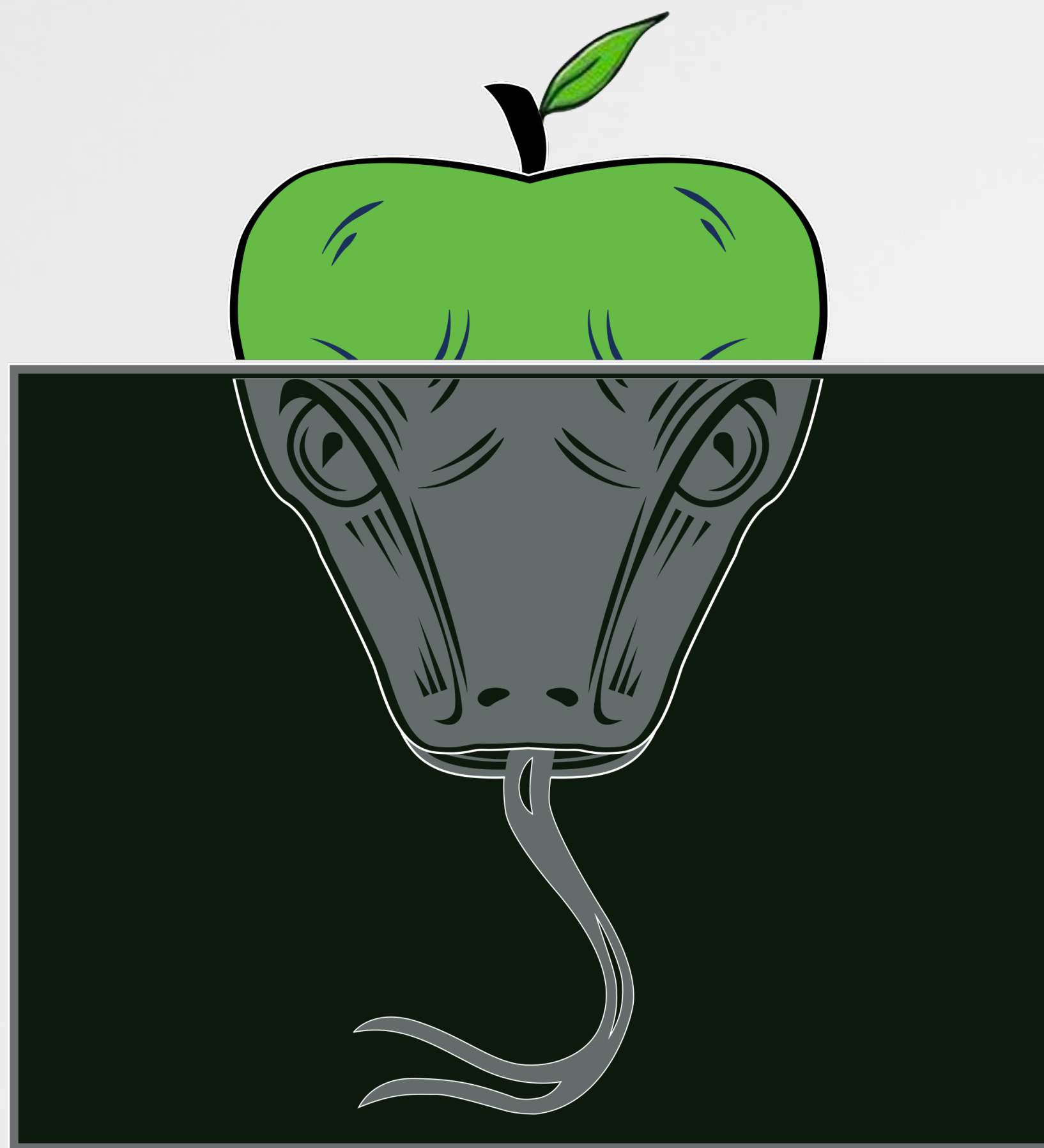
...blog on Objective-See.com



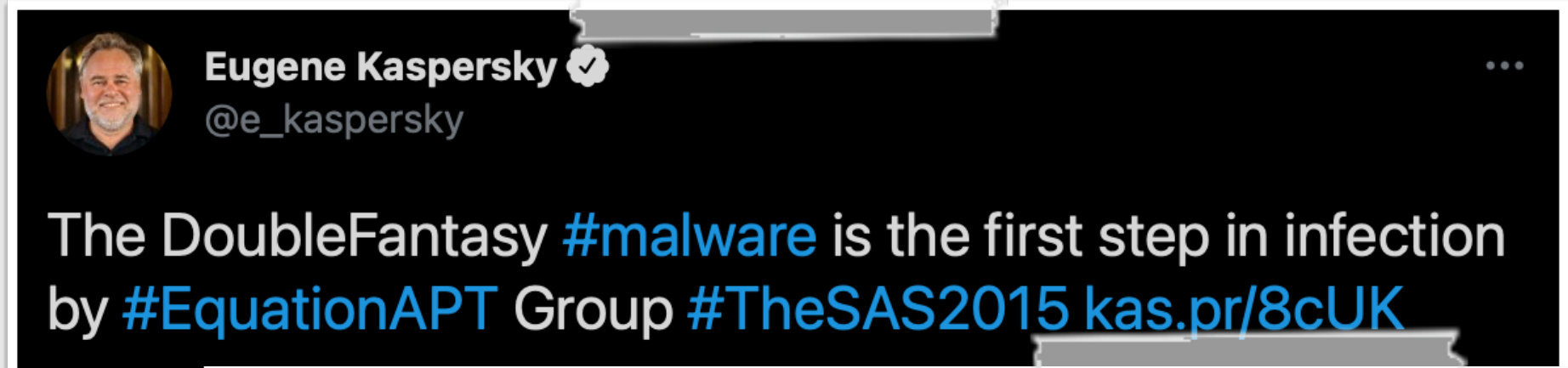
`objective-see.com/blog/blog_0x68.html`

DoubleFantasy

an NSA "first-stage" implant



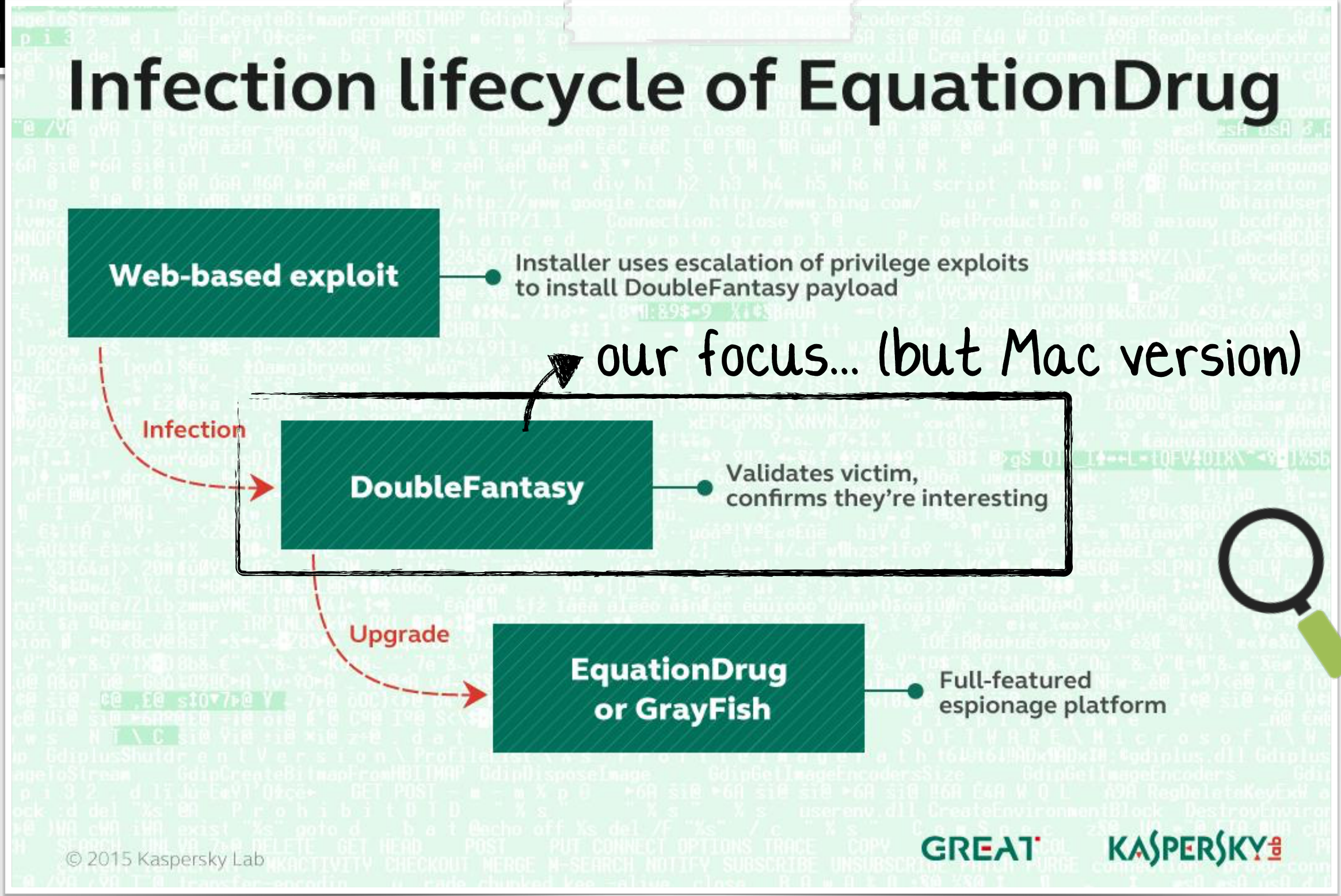
INITIALLY DISCLOSED BY KASPERSKY (*WINDOWS ONLY)



internal (proj.) name?



VALIDATOR is an implant used to gain first access on the target device, collect some preliminary information and enable the subsequent deployment of a larger and more sophisticated malware framework. It is



VALIDATOR

VALIDATOR is a part of a backdoor access system under the FOXACID project. The **VALIDATOR** is a client/server-based system that provides unique backdoor access to personal computers of targets of national interest, including but not limited to terrorist targets. **VALIDATOR** is a small Trojan implant used as a back door against a variety of targeted Windows systems, which can be deployed remotely or via hands on access to any Windows box from Windows 98 through Windows Server 2003. The LP is on-line 24/7 and tasking is 'queued', that is, jobs sit in a queue waiting for the target to 'call home', then the job(s) are sent one at a time to the target for it to process them. Commands are Put a file, get a file, Put, then execute a file, get system information, change **VALIDATOR** ID, and Remove itself. **VALIDATOR**'s are deployed to targeted systems and contact their Listening Post (LP) (each **VALIDATOR** is given a specific unique ID, specific IP address to call home to it's LP); SEPI analysts validate the target's identity and location (USSID-18 check), then provide a deployment list to Olympus operators to load a more sophisticated Trojan implant (currently OLYMPUS, future UNITEDRAKE). An OLYMPUS operator then queue up commands for the specific **VALIDATOR** ID's given by SEPI. Process repeats itself. Once target is hooked with the more sophisticated implant, **VALIDATOR** operators tend to cease. On occasion, operators are instructed by SEPI or the SWO to have VAIDATOR delete itself.

i DoubleFantasy: "the 1st step in infection ... validates the victim, confirms they're interesting" -Kaspersky

WHAT ABOUT A MAC VERSION?

...seemed likely!

13. Have you seen any non-Windows malware from the Equation group?

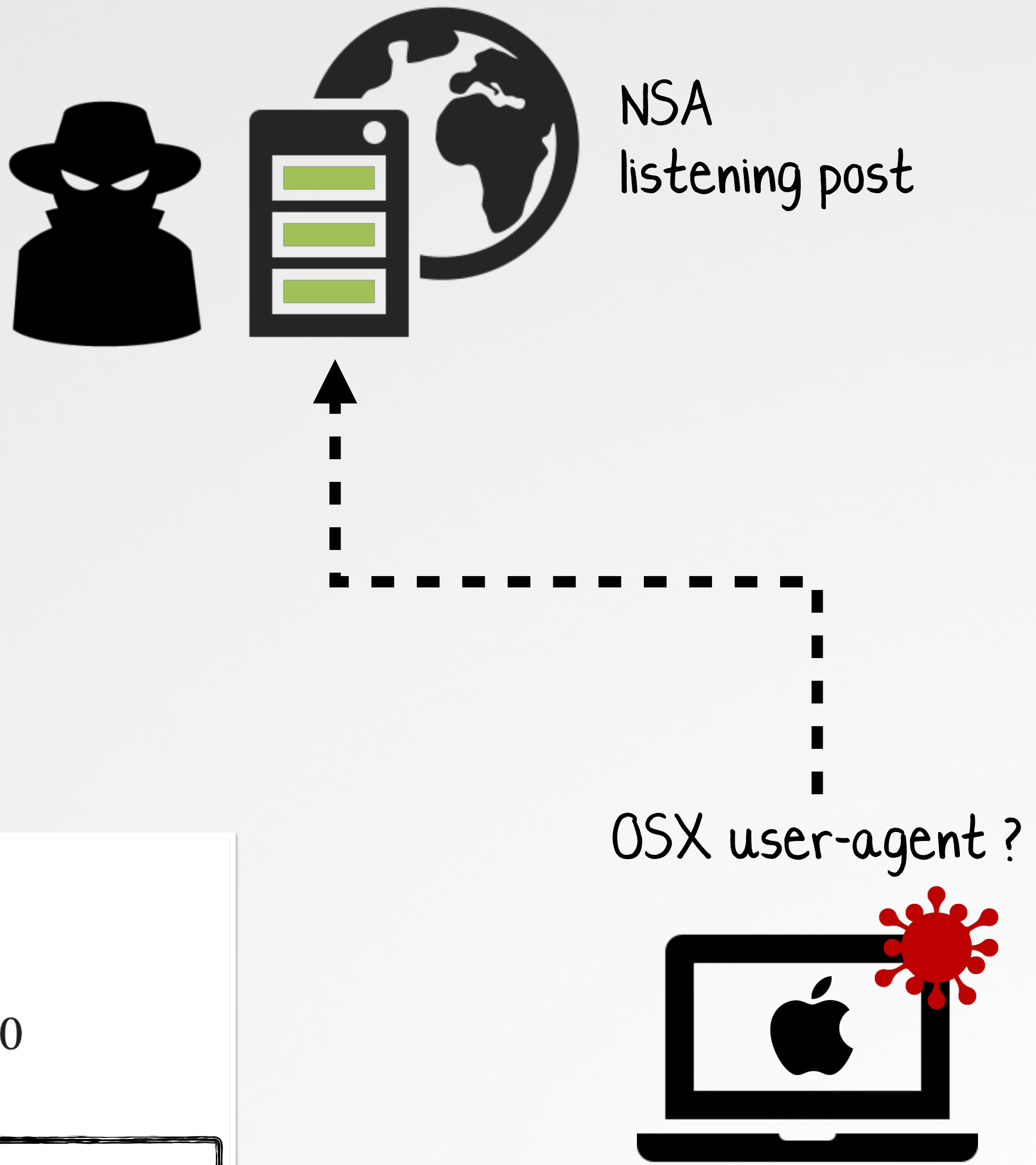
All the malware we have collected so far is designed to work on Microsoft's Windows operating system. However, there are signs that non-Windows malware does exist. For instance, one of the sinkholed C&C domains is currently receiving connections from a large pool of victims in China that appear to be Mac OS X computers (based on the user-agent).

OSX victims ...in China?

- Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:21.0) Gecko/20100101 Firefox/21.0
- Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/536.28.10 (KHTML, like Gecko) Version/6.0.3 Safari/536.28.10

This leads us to believe that a Mac OS X version of DOUBLEFANTASY also exists.

from: "Equation Group: Q&A" (Kaspersky)



ON VIRUSTOTAL

...since 2014?! 🤔

Date	Name	Source	Country
2014-08-05 05:02:57	mdworker	00719389 - web	CN

submitted from China

2014: 'mdworker'
(SHA-1: 1cb054c7186d52cf0c5db42a2ecb4a57b605b74f)

Date	Analyses
2014-08-05T05:02:57	0 / 54
2015-07-31T00:54:28	0 / 56
2020-03-30T16:36:10	2 / 61

SUMMARY	DETECTION	DETAILS	CONTENT	SUBMISSIONS
Security vendors' analysis on 2020-03-30T16:36:10				
Kaspersky			⚠️ HEUR:Trojan.Win32.EquationDrug.gen	
ZoneAlarm by Check Point			⚠️ HEUR:Trojan.Win32.EquationDrug.gen	

...but no detections until 2020!?

TRIAGING 'MDWORKER'

a (rather small), unsigned 32bit mach-0 binary

tool: "What's Your Sign"



```
% file mdworker
mdworker: Mach-0 executable i386

% codesign -dvv mdworker
mdworker: code object is not signed at all

% du -h mdworker
116K
```

file & code-signing info

note: few dependences

```
% otool -hv mdworker
Mach header
      magic      cputype      filetype
      MH_MAGIC    I386         EXECUTE

% otool -L mdworker
/System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation
/System/Library/Frameworks/SystemConfiguration.framework/Versions/A/SystemConfiguration
```

(mach-o) header & dependencies

EMBEDDED STRINGS

...but mostly encrypted?

```
% strings - mdworker

Unknown error %d
http
%%%02x
0x00
[IP address]
Basic
Digest
Missing argument for '-x'.

9??]??
z??$?w??
I??
??>?b?g?
X3?? (?u?jw??
```

embedded strings

```
0x00018302      db      "\xB9\xCF\xFE\xF4\x03\x1A\xA5v\xFD", 0
              aUx03x0exf5x025:
0x0001830c      db      "u\x03\x0E\xF5\x025\xF3", 0
              a3ex0exfaxf9x1e:
0x00018314      db      "3E\x0E\xFA\xF9\x1E\xA5v\xFD", 0
              aXdcxcexb5x1b:
0x0001831e      db      "\xDC\xCE\xB5\x1B", 0
              aX1fixfexfbxf8x:
0x00018323      db      "\x1Fi\xFE\xFB\xF8\x1E\xA5v\xFD", 0
              aXadxdbxf9xf5x0:
0x0001832d      db      "\xAD\xDB\xF9\xF5\x0F\xE2U\xFC", 0
              aX1alxf7x08xfcx:
0x00018336      db      "\x1A\xF7\x08\xFC\xE6U\xFC", 0
              aX8fx9xf9xf7x0:
0x0001833f      db      "\x8F\xF9\xF9\xF7\x01\x1E\xA5v\xFD", 0
```

...many appear encrypted 😞




Malware authors encrypt strings to protect sensitive information and to hinder analysis.

...decryption, is a must!

Decryption via a Disassembler Script

...in four easy(ish) steps

- 1 Find start & end of `__cstring` segment  contains all encrypted strings
- 2 Extract each encrypted string
- 3 Run each string thru decryption algorithm
- 4 Annotate disassembly with (now) decrypted string

start: `"\xC7\xAED\x90z\x81"` -----
end: `/tmp`

0x0017eb7

db

`"\xC7\xAED\x90z\x81", 0`

`; /tmp`

1 FIND START & END OF CSTRING SEGMENT

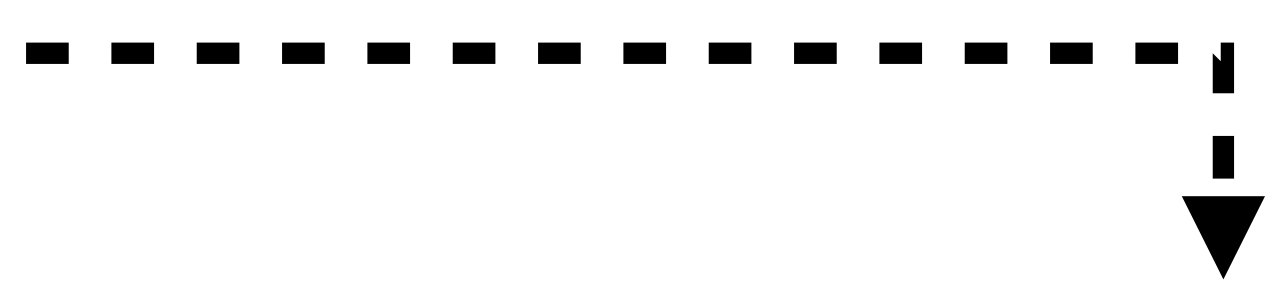
...contains all the encrypted strings

```
01 doc = Document.getCurrentDocument()
02
03 #iterate over each segment & section
04 for i in range(doc.getSegmentCount()):
05     seg = doc.getSegment(i)
06     print('segment: ' + seg.getName())
07
08     for sect in seg.getSectionsList():
09         print(' section: ' + sect.getName())
```

iterate over all segments & their sections

```
01 #find cstring section/info
02 if '__cstring' == sect.getName():
03
04     cSegment = segment
05     cSection = section
06     cSectionStart = section.getStartingAddress()
07     cSectionEnd = cSectionStart + section.getLength()
```

...looking for "__cstring"



```
segment: __TEXT
section: __text
section: __cstring
start: 0x17e50 -> end 0x187fb
```

result: output

2 EXTRACT ENCRYPTED STRINGS

...and then pass into decryption routine

```
0x00018302 db "\xB9\xCF\xFE\xF4\x03\x1A\xA5v\xFD", 0
           aUx03x0exf5x025:
0x0001830c db "u\x03\x0E\xF5\x025\xF3", 0
           a3ex0exfaxf9x1e:
0x00018314 db "3E\x0E\xFA\xF9\x1E\xA5v\xFD", 0
           aXdcxcexb5x1b:
0x0001831e db "\xDC\xCE\xB5\x1B", 0
```



full code, contains other "is really an encrypted string" checks (e.g. has cross-refs).

```
01 i = cSectionStart
02
03 #extract each string
04 # then pass it do the decryption function
05 while i < cSectionEnd:
06     encryptedStr = []
07
08     while(0 != segment.readByte(i)):
09         encryptedStr.append(segment.readByte(i))
10         i += 1
11
12     decryptedStr = decrypt(encryptedStr)
```

iterate over `__cstrings` section
extracting all strings



decrypt

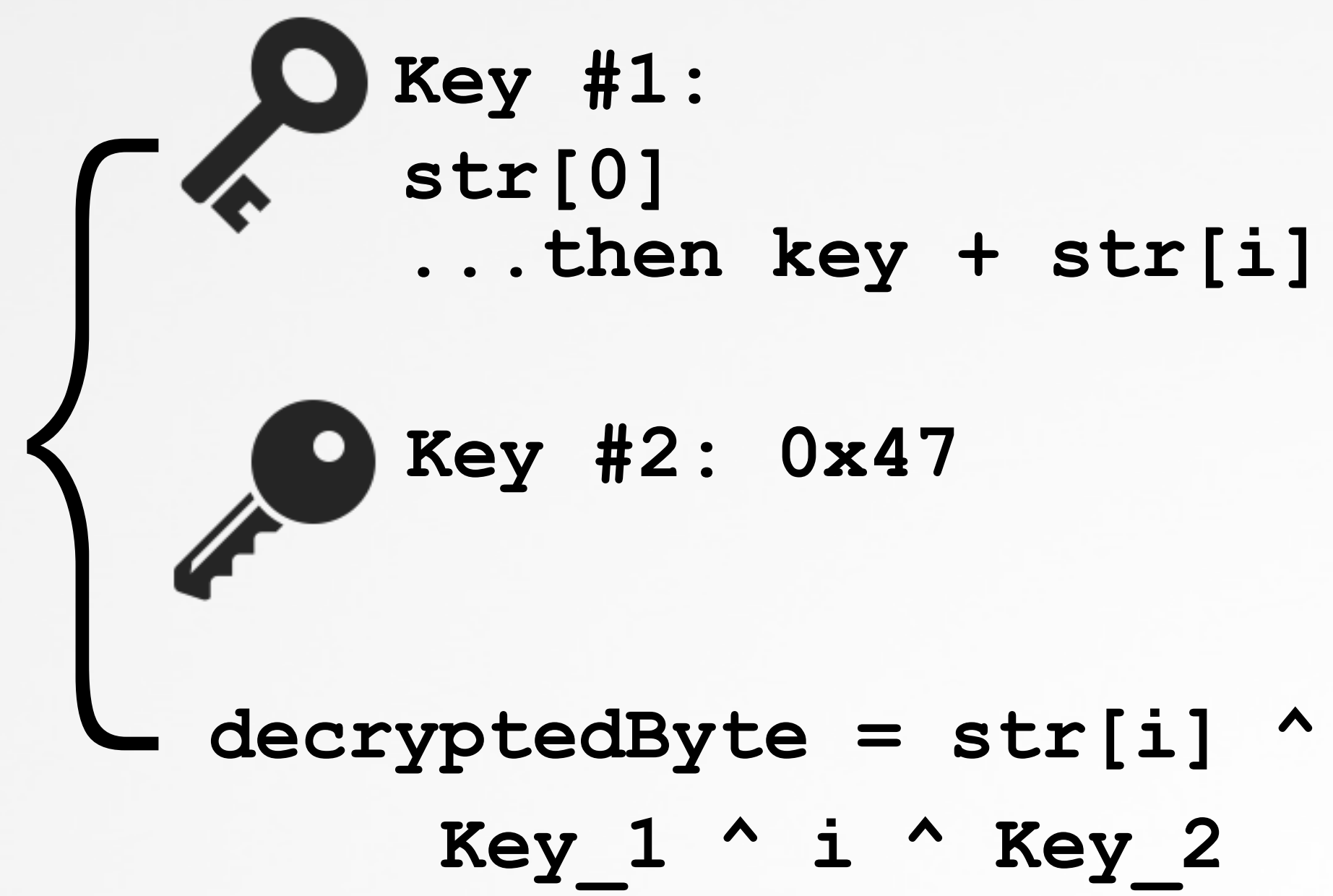
THE STRING DECRYPTION ALGORITHM

...located via static analysis (x-refs)

```
01 main:
02 ...
03 0x0000248b lea  eax, dword [ebx+0x15a3e] ;0x00017eb7 ("\xC7\xAED\x90z\x81")
04 0x00002495 call sub_d900
```



```
01 sub_d900:
02 ...
03 0x0000d930 movzx  edx, byte [esi]
04 0x0000d933 inc    esi
05 0x0000d934 mov    byte [ebp+var_D], dl
06 0x0000d937 mov    eax, edx
07 0x0000d939 mov    edx, dword [ebp+arg_0]
08 0x0000d93c xor    eax, edi
09 0x0000d93e xor    eax, ecx
10 0x0000d940 xor    eax, 0x47
11 0x0000d943 mov    byte [edx+ecx-1], al
12 0x0000d947 movzx  eax, byte [ebp+var_D]
13 0x0000d94b inc    ecx
14 0x0000d94c add    edi, eax
15 0x0000d94e cmp    ecx, dword [ebp+var_C]
16 0x0000d951 jne   loc_d930
```



string decryption algorithm (#1)

3 DECRYPT STRINGS

...via a reimplemented decryption algorithm



```
01 def decrypt(encryptedStr):
02     result = ""
03     decryptedStr = []
04
05     #init both XOR keys
06     xorKey_1 = encryptedStr[0]
07     xorKey_2 = 0x47
08
09     #decrypt each byte (and update key)
10     for i in range(1, len(encryptedStr)):
11         byte = (encryptedStr[i] ^ xorKey_1 ^ i ^ xorKey_2) & 0xFF
12         decryptedStr.append(chr(byte))
13
14         xorKey_1 = encryptedStr[i] + xorKey_1
15
16     result = ''.join(decryptedStr)
```

- 1 init xor key(s)
- 2 decrypt byte by byte
- 3 update xor key #1

(re)implemented decryption algorithm

4 ADD DECRYPTED STRING TO DISASSEMBLY

both at string's location, and at any x-refs

```
01 #add decrypted string as inline comment
02 cSegment.setInlineCommentAtAddress(stringStart, decryptedString)
03
04 #for each reference
05 # add decrypted string as inline comment
06 for reference in segment.getReferencesOfAddress(stringStart):
07     segment.setInlineCommentAtAddress(reference, decryptedString)
```

Add as comment:

{ inline, at string
and, at each x-reference

"\xC7\xAED\x90z\x81"

↓
"/tmp"

aXc7xaedx90zx81:

db "\xC7\xAED\x90z\x81", 0

; /tmp, DATA XREF=main+27

↓ + at x-reference(s)

```
lea    eax, dword [ebx-0x2479+aXc7xaedx90zx81]
mov    dword [esp+0x38+var_34], eax
call   sub_d900
```

```
; /tmp, "\\xC7\\xAED\\x90z\\x81"
; argument #2 for method sub_d900
; sub_d900
```


STRINGS DECRYPTION ALGORITHM #2

...a simple multiplication scheme

```

01 0x00010019 lea    eax, dword [ebx+0x86dd] ;0x00018514
02 0x0001001f mov    dword [esp+0x8], 0xb
03 ...
04 0x0001002d mov    dword [esp+0x4], esi
05 ...
06 0x00010034 mov    dword [esp], eax ; argument #1 (encrypted string)
07 0x00010037 call   sub_14030
  
```

```

0x00018514 db "'\xE9\xE90Q\x15\xBDn\xE9", 0
  
```

```

01 sub_14030:
02 ...
03
04 loop:
05 0x00014080 mov    ecx, dword [ebp+arg_0]
06 0x00014083 movsx  eax, byte [ecx+edx]
07 0x00014087 imul  eax, eax, 0x1d
08 0x0001408a mov    byte [esi+edx], al
09 0x0001408d inc    edx
10 0x0001408e cmp    edi, edx
11 0x00014090 jne    loop
  
```

$$\text{decryptedByte} = \text{encryptedStr}[i] * 0x1D$$

string decryption algorithm (#2)

DECRYPTED STRINGS

cmdline options, configurations, and more?

```
"cdi:l:s:p:"  
"lp"  
"ICD"  
"KEY"  
  
"/Library/Caches/  
com.apple.LaunchServices-02300.csstore"  
  
"/Default.aspx?%s"  
"Accept-Language"  
"SESSID="0d1975bf%11x9c:eac:%u:%u"  
  
"Proxy-Authenticate"  
"Proxy-Authentication-Info"
```

cmdline options?

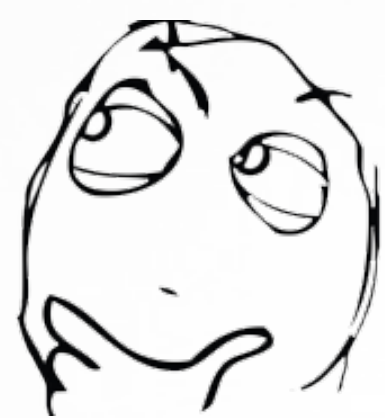
config keys?

??

lp comms?

proxy aware?

decrypted strings



(MORE) DECRYPTED STRINGS

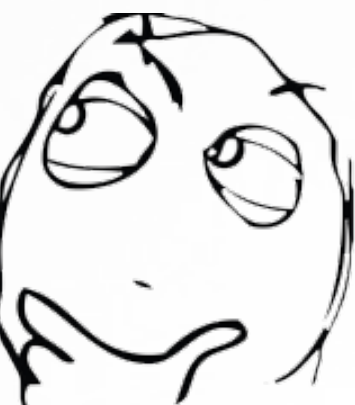
...survey template, and more?

```
"001:%s"  
"002:%i.%i.%i.%i %llu"  
"003:%s"  
"004:NO PROXY HERE"  
...  
"MACHTYPE"  
"036:%s"  
...  
"LANG"  
"043:%s"  
...  
"045:%d Years %d  
    Days %d Hours %d Minutes"  
  
"DYLD_INSERT_LIBRARIES" - - - - -> ??
```

} survey?

recall implants supposed goal is
to validate targets of interest...

(more) decrypted strings



IN APPLE'S MALWARE REMOVAL TOOL (MRT)?

say hello to "OSX.ATG11.A"

```

% grep -R /Library/Caches/com.apple.LaunchServices-02300.csstore /Library/Apple/System/
Binary file /Library/Apple/System/Library/CoreServices/MRT.app/Contents/MacOS/MRT matches

$ strings - MRT.app/Contents/MacOS/MRT
OSX.ATG11.A
~/Library/LaunchAgents/com.apple.mdworker.plist
~/Library/Assistants/mdworker
/Library/Caches/com.apple.LaunchServices-02300.csstore

```

match (decrypted) string

```

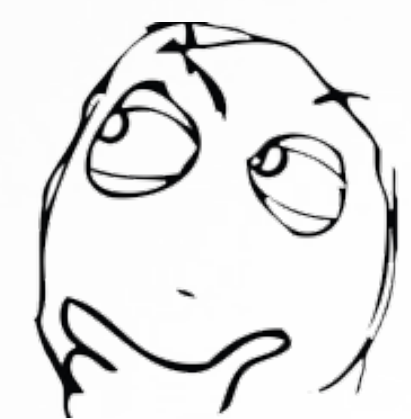
01 sub_1000e1ab0:
02 ...
03 adrp x11, #0x100147000 ; 0x100147457@PAGE
04 add x11, x11, #0x457 ; "OSX.ATG11.A"
05
06 adrp x10, #0x100147000 ; 0x100147470@PAGE
07 add x10, x10, #0x470 ; "~/Library/LaunchAgents/com.apple.mdworker.plist"
08
09 adrp x0, #0x100147000 ; 0x1001474a0@PAGE
10 add x0, x0, #0x4a0 ; "~/Library/Assistants/mdworker"
11
12 adrp x0, #0x100147000 ; 0x1001474c0@PAGE
13 add x0, x0, #0x4c0 ; "/Library/Caches/com.apple.LaunchServices-02300.csstore"

```

Apple's name

} persistence?

---> ??



MRT, disassembled

FINDING MAIN()

as this is where we'll start our analysis

```
% otool -l mdworker
...
Load command 9
cmd LC_UNIXTHREAD
flavor i386_THREAD_STATE
eax 0x00000000 ebx 0x00000000 ecx 0x00000000 edx 0x00000000
edi 0x00000000 esi 0x00000000 ebp 0x00000000 esp 0x00000000
ss 0x00000000 eflags 0x00000000 eip 0x00002140 cs 0x00000000
```

LC_UNIXTHREAD:
load command, with the
initial program state (deprecated).

```
01 EntryPoint:
02 0x00002140 push 0x0
03 0x00002142 mov ebp, esp
04 ...
05 0x00002164 call sub_216a
```

entry point
(by default: standard c-runtime)

```
01 sub_216a:
02 0x0000216a push ebp
03 0x0000216b mov ebp, esp
04 0x0000217c mov dword [_NXArgc], eax
05 0x00002181 mov dword [_NXArgv], edi
06 0x00002187 mov dword [_environ], ebx
07 ...
08
09 0x0000223d call sub_2470
10 0x00002242 mov dword [esp], eax
11 0x00002245 call imp___jump_table__exit
```

the call to main
(followed by a call to exit)

ANALYSIS OF MAIN

initialization, cmdline parsing, and then?

```
01 //0x00002470
02 int main(int argc, char* argv[]) {
03     eax = sub_d900(&var_19, "\xC7\xAED\x90z\x81", 0x6);
04     chdir(eax);
05
06     sub_2260(argc, argv);
07
08     sub_cd80();
09
10     if (sub_ce30() == 0x0) {
11         sub_2420();
12     }
13
14     return 0x0;
15 }
```

"/tmp"

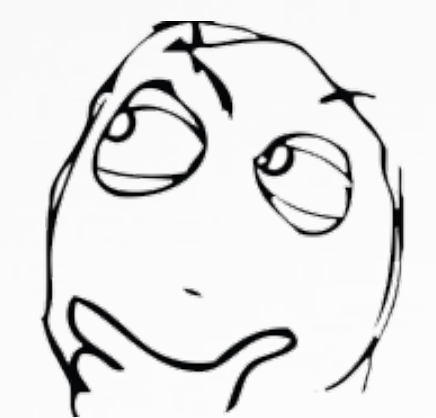
change dir. to /tmp

parse args

signal handlers

daemonize, ...and then ??

main()



PARSING CMDLINE ARGS

getopt: "cdi:l:s:p:" and the logic for "-c"

```
01 //parse args
02 int sub_2260(int argc, char* argv[]) {
03     ...
04 nextArg:
05     eax = sub_d900(eax, "6\x13h\x9C4\xAFKH\xB6G\xB7;", 0xc);
06     eax = getopt(var_4, var_8, eax, 0x0, 0x0);
07     if (eax == 0xffffffff) goto .leave;
08
09     if (eax == 'c') goto handleC;
10
11     if (eax == 'd') exit(0x0);
12
13     goto nextArg
```

"cdi:l:s:p:"

```
01 handleC:
02     eax = sub_7b20();
03     if (eax == 0x0)
04         printf(sub_d900(..., "\xAA\xA3C\xDE-", 0x5));
05
06     exit(0x0);
```

"OK"

```
01 int sub_7b20() {
02     eax = sub_d900(&var_49, "| \x15\x98\x04\x0C\tb\x96\x0C'c\xFD...\xC3\x88\x0C\x1C,\x14", 0x38);
03     return = sub_7a30(eax);
04 }
```

"/Library/Caches/com.apple.LaunchServices-02300.csstore"

```
01 int sub_7a30(char* path) {
02     if(0 == stat(path, &var_68)
03         unlink(path);
04 }
```

logic for "-c"

PARSING CMDLINE ARGS

`-c`, cleanup and then exit

```
$ ./doubleFantasy -c  
OK
```

```
$ touch /Library/Caches/com.apple.LaunchServices-02300.csstore  
  
# fs_usage -w -f filesystem  
stat /Library/Caches/com.apple.LaunchServices-02300.csstore doubleFantasy  
  
unlink /Library/Caches/com.apple.LaunchServices-02300.csstore doubleFantasy  
  
$ ls /Library/Caches/com.apple.LaunchServices-02300.csstore  
ls: /Library/Caches/com.apple.LaunchServices-02300.csstore: No such file or directory
```

file monitoring
(note: unlink)



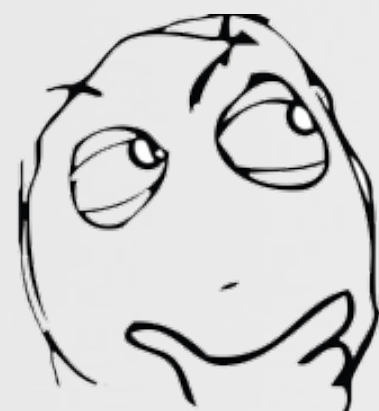
com.apple.LaunchServices
-02300.csstore



SUPPORTED CMDLINE ARGS

though most aren't implemented?

Value	Meaning	Action
c	cleanup	remove com.apple.LaunchServices-02300.csstore ...and then exit
d	die	just exit
i:	interactive??	...not implemented
l:	listening post??	...not implemented
s:	??	...not implemented
p:	persist??	...not implemented



this version,
...compiled without support for **i:l:s:p: ??**

DAEMONIZE

...twice, to prevent tty acquisition

```
01 int sub_ce30() {
02
03     eax = fork();
04
05     //child
06     if(0 == eax) {
07         setsid();
08
09         eax = fork();
10
11         //grandchild
12         if(0 == eax) {
13
14             //decrypts to: /dev/null
15             eax = decryptStr(&var_40, "\xD8\xB1...", 0xb);
16
17             //redirect handles
18             eax = open(eax, 0x2);
19             dup2(eax, 0x0);
20             dup2(eax, 0x1);
21             dup2(eax, 0x2);
22             close(eax);
23         }
24     }
25     ...
```

"The standard way to create a daemon is to simply do `p=fork()`; `if(p) exit()`; `setsid()`.

In this case, the parent also exits and the first child process is reparented.

The double-fork magic is only required to prevent the daemon from acquiring a tty." -parasietje (s.o)

THAT FILE?

`/Library/Caches/com.apple.LaunchServices-02300.csstore`

```
01 int sub_7a80() {  
02  
03     path = decryptStr("\\xDA\\xB3\\...\\x1C\\x14");  
04  
05     sub_7760(path, ...) - - - - -
```

`com.apple.LaunchServices-02300.csstore`

```
01 int sub_7760(char* path, ...) {  
02  
03     handle = open(path, 0x0);  
04     result = read(handle, var_5020, 0x2800);  
05  
06     //decrypt file's contents
```

```
# fs_usage -f filesystem
```

```
open F=3 (R____) /Library/Caches/com.apple.LaunchServices-02300.csstore  
RdData[A] B=0x1000 /Library/Caches/com.apple.LaunchServices-02300.csstore
```

```
doubleFantasy  
doubleFantasy
```

file monitoring

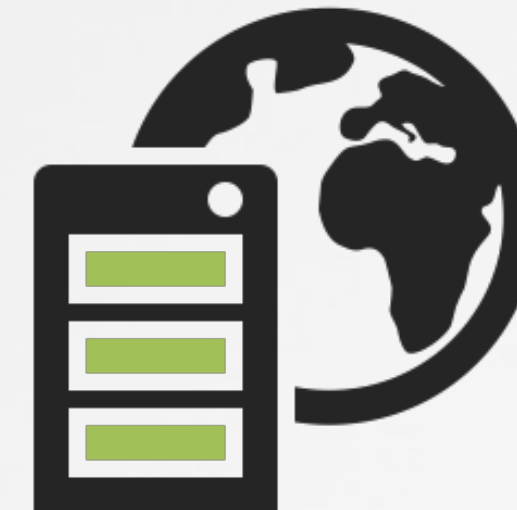
CONFIG FILE?

extract (config) parameters

```
01 int sub_3f80() {  
02     //load & decrypt config file  
03  
04  
05  
06     //decode/find string: "CIAE"  
07     //decode/find string: "lp"  
08     //decode/find string: "CI"  
09     //decode/find string: "ICD"  
10     //decode/find string: "SDI"  
11     //decode/find string: "MD"  
12     //decode/find string: "KIDX"  
13     //decode/find string: "KEY"  
14     //decode/find string: "ETC"  
15     //decode/find string: "ECSD"  
16     //decode/find string: "EDC"  
17     //decode/find string: "ICD"  
18     //decode/find string: "IL"  
19     //decode/find string: "IM"  
20     //decode/find string: "MDM"
```



no access to sample
config file...



listening post?



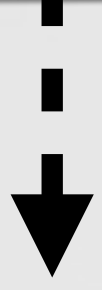
DoubleFantasy (v8.2.0.3/Windows) :

"C&C IPs or hostnames (specified in config ...)" -Kaspersky

COMMAND AND CONTROL COMMUNICATIONS

...via the open-source neon http library?

```
0x000158de    mov     dword [esp+0x38+var_30], 0x28    ; argument #3 for method sub_14030
0x000158e6    mov     dword [esp+0x38+var_34], eax    ; argument #2 for method sub_14030
0x000158ea    lea    eax, dword [esp+0x38+var_34]    ; http://webdav.org/neon/hooks/proxy-auth, "\\x88\\x04\\x040\\x02\\xBB
0x000158f0    mov     dword [esp+0x38+var_38], eax    ; argument #1 for method sub_14030
0x000158f3    call   sub_14030                        ; sub_14030
```



NSA in violation of GPL? 🤔

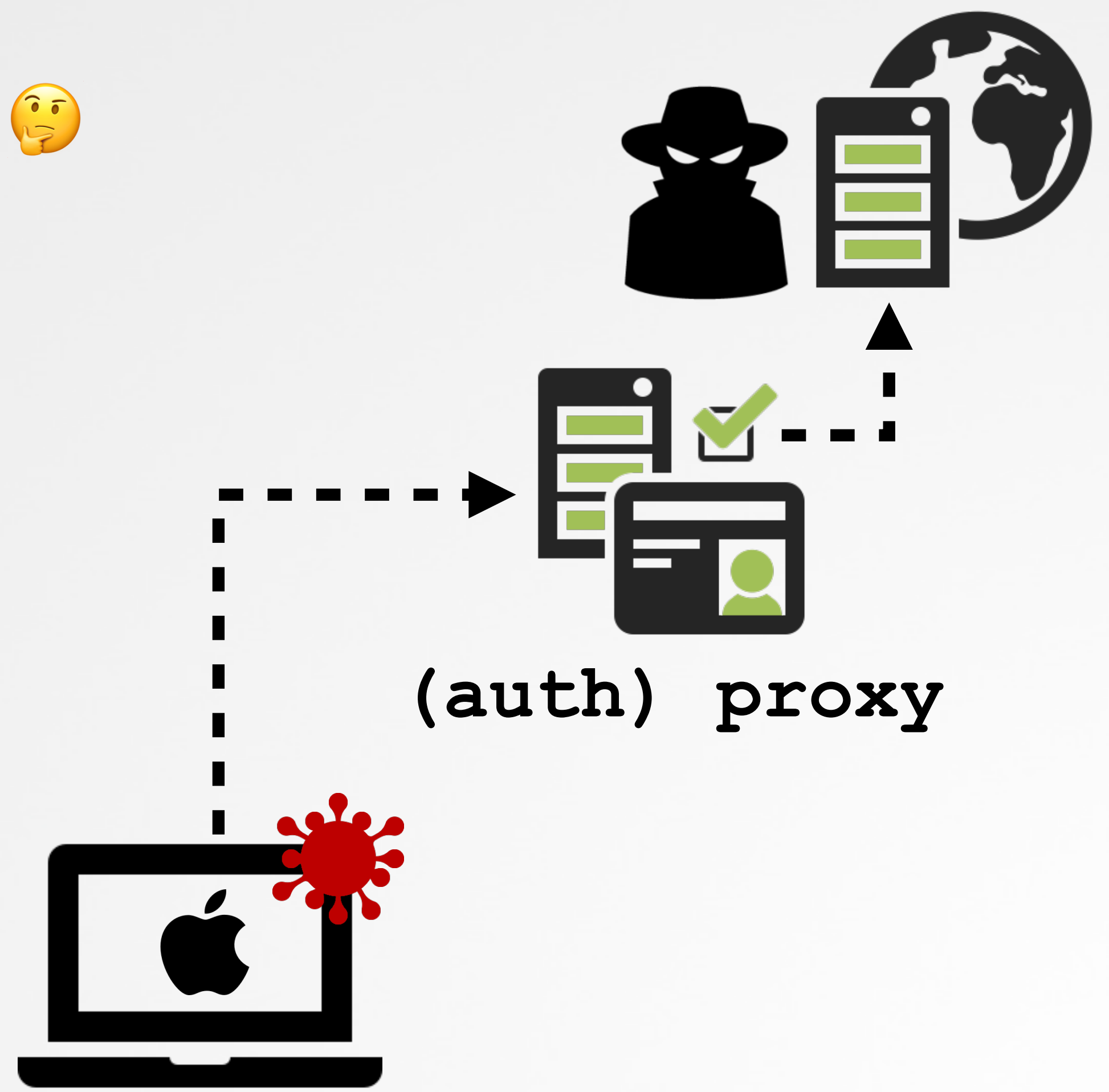
neon

neon is an HTTP and WebDAV client library, with a C language API.

Features:

- High-level interface to HTTP and WebDAV methods.
- Low-level interface to HTTP request handling, to allow implementing new methods easily.
- Persistent connection support (HTTP/1.1 and HTTP/1.0 aware)
- Basic and Digest authentication (RFC 7616/7617, including SHA-2, userhash)
- Kerberos (Negotiate) and SSPI/NTLM authentication (Unix and Windows)
- HTTP and SOCKS (v4/5) proxy support (including authentication)
- SSL/TLS support using OpenSSL or GnuTLS (client certs via files or PKCS#11)
- Generic WebDAV 207 XML response handling mechanism

neon
(github.com/notroj/neon)



SURVEY!

survey all the thingz



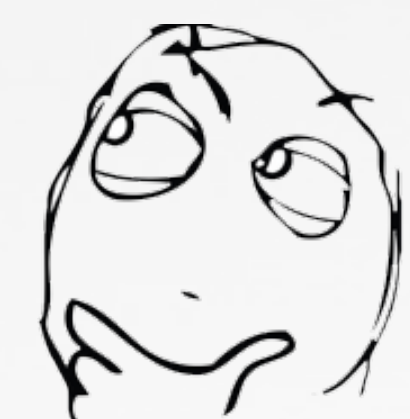
DoubleFantasy: "validates victim, confirms they're interesting" -Kaspersky

```
01 int sub_b1b0() {
02
03     //"001:%s"
04
05     //"002:%i.%i.%i.%i %llu"
06
07     //"003:%s"
08
09     //"004:NO PROXY HERE"
10
11     ...
12
13     //"045:%d Years %d Days %d Hours %d Minutes"
14
15     ...
16
17     //"048:%s"
```

embedded strings
(...now, decrypted)

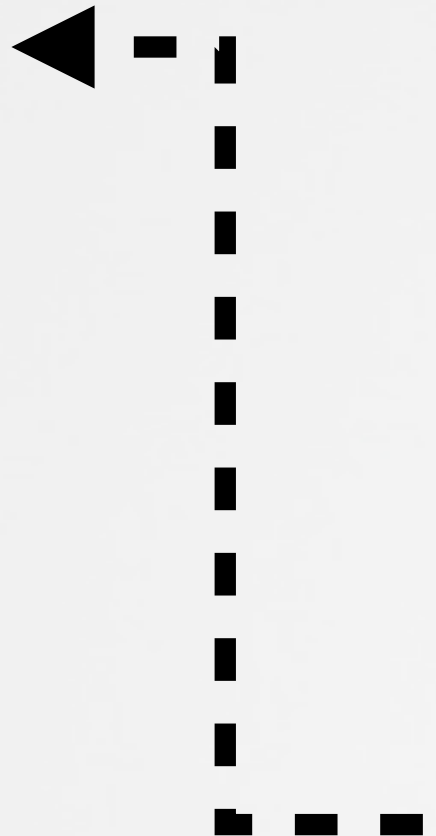
(in a debugger), can we just modify the instruction pointer to coerce the malware to survey our VM?

...of course! :)



```
(lldb) reg write $pc 0x4910
```

```
01 sub_4910:
02 ...
03 0x00004970 call sub_b1b0 ;survey?
```



SURVEY!

survey all the thingz


```
01 sub_4910:  
02 ...  
03 0x00004970 call sub_b1b0 ;survey?
```

once sub_b1b0 returns,
...a full survey can be found in the return (**EAX**) register :)

```
(lldb) x/s $eax  
001:127.0.0.1  
...  
030:user  
031:501:20  
  
035:Darwin Kernel Version 18.6.0: ... xnu-4903.261.4~2/RELEASE_X86_64  
  
038:HST  
040:Sat Sep 4 21:52:19 2021  
  
042:users-Mac.local  
043:en_US.UTF-8  
  
048:doubleFantasy
```

abridged survey
(macOS 10.14 VM)

 network  user

 system info

 lang/date/time

(REMOTE) COMMANDS

logic flow, via a jump table

```

01 0x00005adc call $+5
02 0x00005ae1 pop ebx ;ebx = 0x5ae1
03
04 0x00005b69 call sub_4cb0 ;get tasking
05
06 0x00005b8d movzx eax, dl ;cmd #
07 0x00005b90 sub eax, 0x42
08 0x00005b93 cmp eax, 0x53
09 0x00005b96 ja loc_5d00 ;invalid cmd
10
11 0x00005b9c mov eax, dword [ebx+eax*4+0xc7] ;0x5ba8+cmd*4
12 0x00005ba3 add eax, ebx
13 0x00005ba5 jmp eax ;execute command
  
```

command lookup/execution

$$cmd = *(jump\ table + (cmd\ \# * 4)) + ebx$$

ex: cmd 0x1E = 0x3AD + 0x5AE1 = 0x5E8E

```

01 loc_5e8e: ←
02 ...
03 call survey!
  
```

```

01 ;jump table
02 0x00005ba8 dd 0x0000022f ;cmd 0x00
03 0x00005bac dd 0x0000021f ;cmd 0x01
04 0x00005bb0 dd 0x0000021f ;cmd 0x02
05 0x00005bb4 dd 0x0000021f ;cmd 0x03
06 ...
07 0x00005bc8 dd 0x000002c6 ;cmd 0x08
08 ...
09 0x00005bcc dd 0x000003ce ;cmd 0x09
10 ...
11 0x00005c20 dd 0x000003ad ;cmd 0x1E
12 ...
13 0x00005c60 dd 0x0000038c ;cmd 0x2E
14 ...
15 0x00005c74 dd 0x0000036b ;cmd 0x33
16 0x00005c78 dd 0x0000034a ;cmd 0x34
17 ...
18 0x00005c80 dd 0x00000329 ;cmd 0x36
19 0x00005c84 dd 0x00000308 ;cmd 0x37
20 ...
21 0x00005ca0 dd 0x000002e7 ;cmd 0x3E
22 ...
23 0x00005ce8 dd 0x000002c6 ;cmd 0x50
24 ...
25 0x00005cf0 dd 0x000002a8 ;cmd 0x52
26 0x00005cf4 dd 0x0000028a ;cmd 0x53
  
```

jmp table (0x00005ba8)

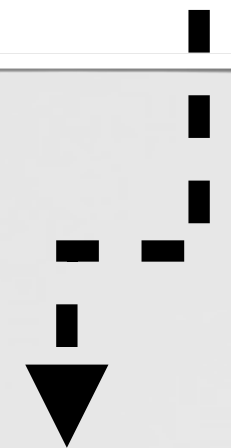
loc_5d00:
"not implemented"

survey

CMD 0x9 (0x00005EAF)

read file to exfiltrate to server?

```
01 0x00005eaf: ;case 9  
02 ...  
03 0x00005ec6 call sub_2d70
```



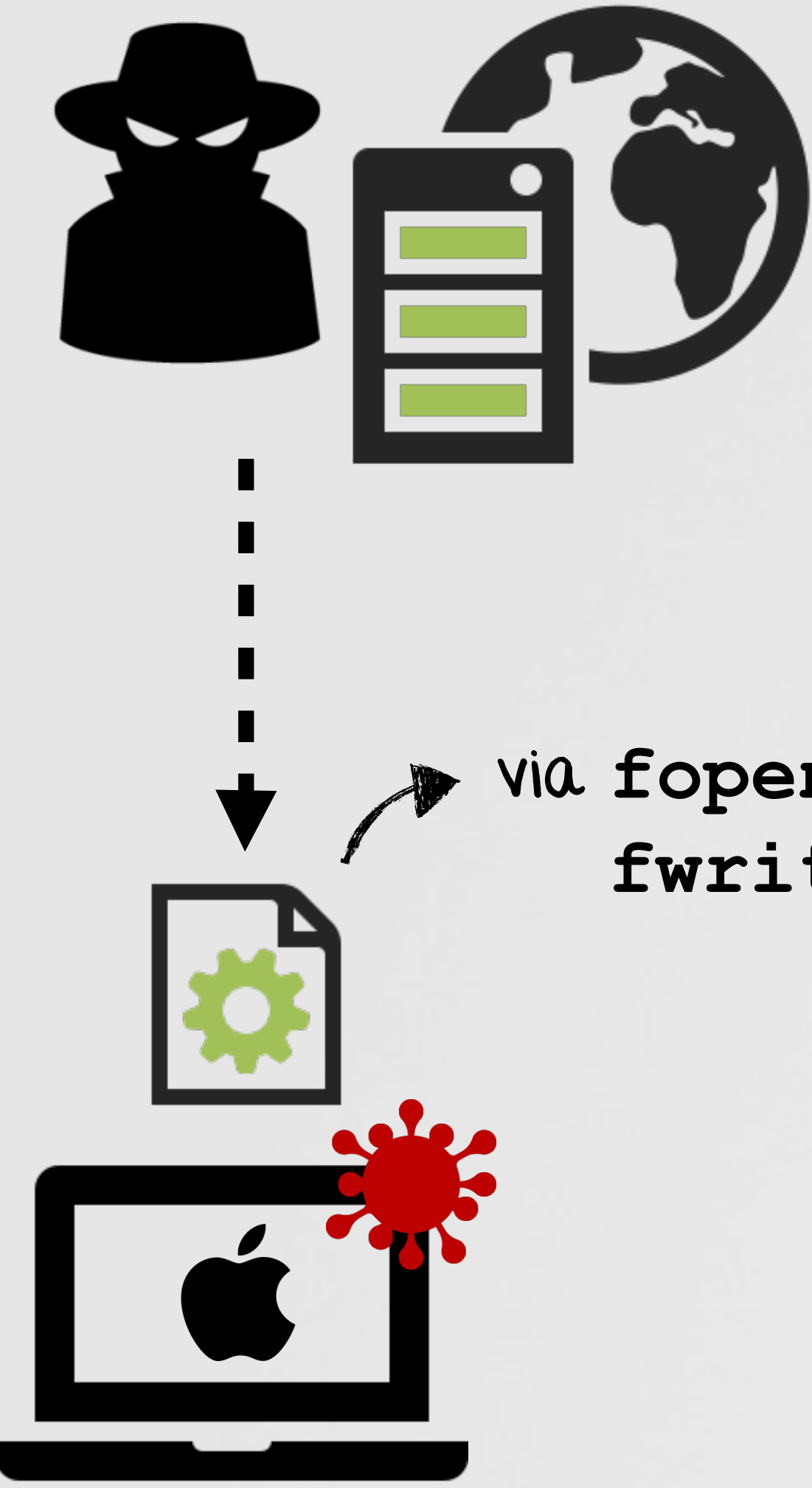
```
01 sub_2d70 (...)  
02  
03 0x00002eb5 lea eax, dword [ebx+0x15143] ; "r"  
04 ...  
05 0x00002ec2 call fopen  
06  
07 0x00002f55 call fseek  
08  
09 0x0000308d call fread  
10  
11 0x00002e1a call fclose
```

open & read a file

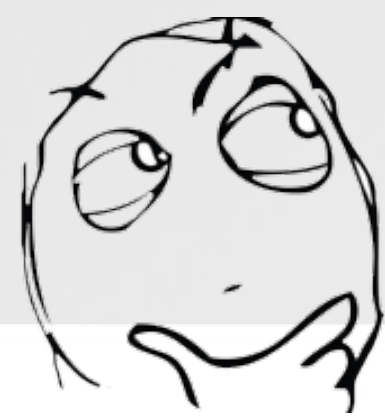


CMD 0x50 (0x00005DA7)

download & exec payload ...with a twist!



via `fopen(..., "w+")`
`fwrite(...)`



...there is an execution/persistence mechanism where the implant is spawned via `DYLD_INSERT_LIBRARIES` ??

```

01 sub_cfd0(char* path) {
02     chmod(path, 0700);
03
04     pid = fork()
05     if(0 == pid) ;child
06     {
07         //unset DYLD_INSERT_LIBRARIES
08
09         execl(...);
10

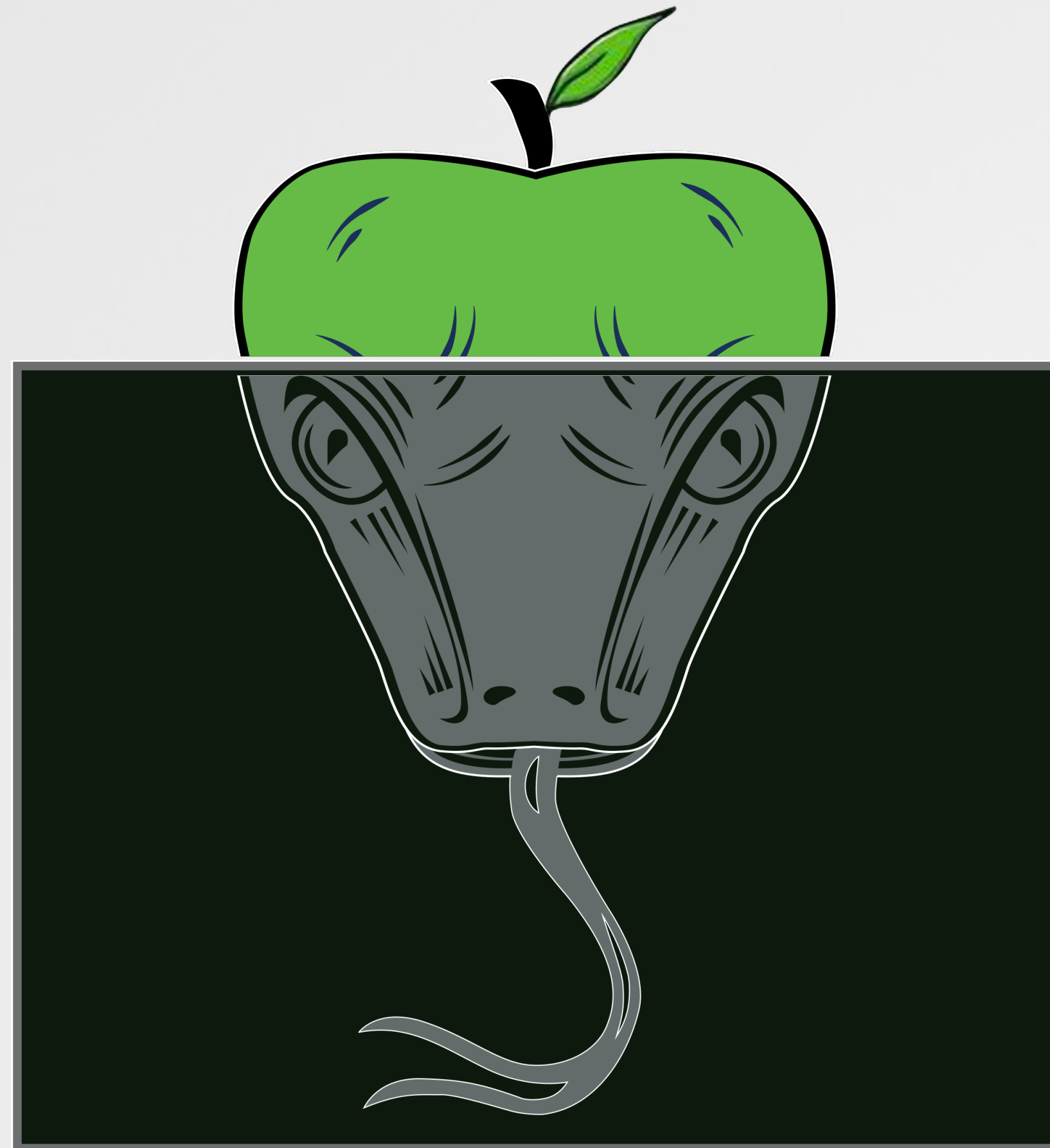
```

```

01 memcpy(*(env + envIndex_DYLD * 0x4),
02         "DYLD_INSERT_LIBRARIES", lengthOf_DYLD);
03
04 *(env + envIndex_DYLD * 0x4) + lengthOf_DYLD) = '=';
05 *(env + envIndex_DYLD * 0x4) + lengthOf_DYLD + 0x1) = '\x00';

```

Conclusions



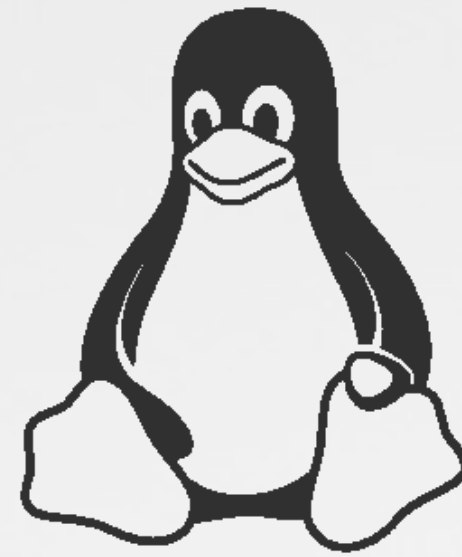
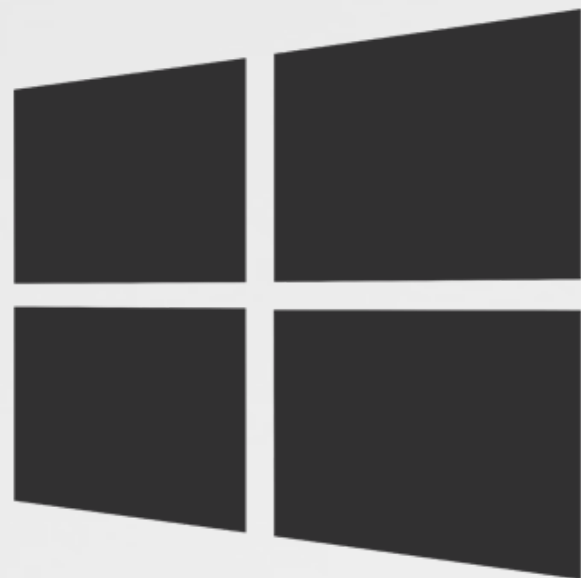
CONCLUSIONS



1

have macOS capabilities

well, obviously ;)



2

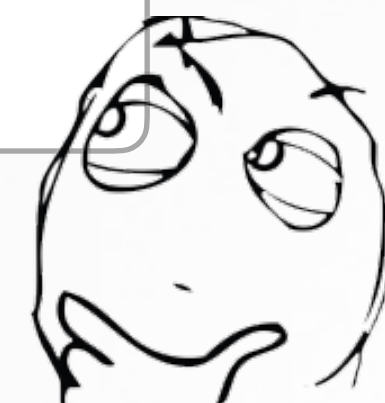
cross-platform functionality



3

well written, & effective...

What/where are their current capabilities?



Made In America

RESOURCES :

"美国中央情报局网络武器库分析与披露"

("Analysis and Disclosure of U.S. Central Intelligence Agency's Cyber Weapons")
<https://ti.qianxin.com/blog/articles/network-weapons-of-cia/>

"Vault 7: CIA Hacking Tools Revealed"

<https://wikileaks.org/ciav7p1/>

"Longhorn: Tools used by cyberespionage group linked to Vault 7"

<https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=7ca2e331-2209-46a8-9e60-4cb83f9602de&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments>

"Unraveling the Lamberts Toolkit"

<https://securelist.com/unraveling-the-lamberts-toolkit/77990/>

"Equation Group: Q&A"

https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064459/Equation_group_questions_and_answers.pdf

"Equation Group: From Houston with Love"

<https://securelist.com/equation-group-from-houston-with-love/68877/>