100DaysofYARA - SpectralBlur

g-les.github.io/yara/2024/01/03/100DaysofYARA_SpectralBlur.html

A Clever Blog Name by Greg Lesnewich January 3, 2024

Jan 3, 2024

Today will be a quick post on a TA444 (aka Sapphire Sleet, BLUENOROFF, STARDUST CHOLLIMA) Macho family tracked as SpectralBlur we found in August, and how finding it led us to stumble upon an early iteration of KANDYKORN (aka SockRacket). Please read Elastic's EXCELLENT piece on that family.

Long story short, internet scan data from Censys tipped us a suspicious domain `pxaltonet[.]org` that we believed had tentative links to the Interception cluster (which we have since merged with TA444 in our dataset). We monitored this domain on VirusTotal (Netloc rules will make this easier now!) and observed a file called `.macshare` being downloaded from the `auth` subdomain.

Triage

In checking out this file, we don't see a lot of initial toolmarks from the developers in the metadata (no codesigning, no leftover RPath, etc.) but a decent amount in the strings - mostly function names! I've trimmed the strings output to only show us interesting ones.

```
{
    "FileType": "THIN",
    "Slices": [
        {
            "Header": {
                "Type": "mach_header_64",
                "Magic": 4277009103,
                "CPUType": "X86_64",
                "CPUSubType": "ALL",
                "FileType": "EXECUTE",
                "LoadCount": 16,
                "LoadSize": 1368,
                "Flags": [
                    "NOUNDEFS",
                    "DYLDLINK",
                    "TWOLEVEL",
                    "PIE"
                ],
                "Reserved": 0
            },
            "LinkedImages": {
                "LOAD_DYLIB": [
                    "/usr/lib/libSystem.B.dylib"
                ]
            },
            "Version": {
                "BuildVersion": {
                    "Platform": "MACOS",
                    "MinOS": "13.0.0",
                    "SDK": "13.3.0",
                    "Ntools": 1
                },
                "SourceVersion": 0
            },
            "UUID": "833902ac1aba3cee87dc52ac9f045f26",
            "BaseName": "",
            "InstallName": ""
        }
    ]
}
```


```
emit macshare | carve printable --min=6

/usr/lib/dyld
/usr/lib/libSystem.B.dylib
fffff.
AWAVATSH
[A\A^A_]
%s
%s
%s
There is NO WARRANTY, to the extent permitted by law.
/dev/null
/bin/sh
```

```
_mh_execute_header
authchannel
hangout
load_config
openchannel
read_packet
xcrypt
lose_fcontext
_fcontext
process
thread
ownload
_content
getcfg
hibernate
testconn
upload
estart
_content
ave_config
igchild
ocket_
onnect
ait_read
rite_packet
_value
__mh_execute_header
_authchannel
_close_fcontext
_hangout
_init_fcontext
_load_config
_mainprocess
_mainthread
_openchannel
_proc_die
_proc_dir
_proc_download
_proc_download_content
_proc_getcfg
_proc_hibernate
_proc_none
_proc_restart
_proc_rmfile
_proc_setcfg
_proc_shell
_proc_sleep
_proc_stop
_proc_testconn
_proc_upload
_proc_upload_content
_procs
_read_packet
_save_config
_sigchild
```

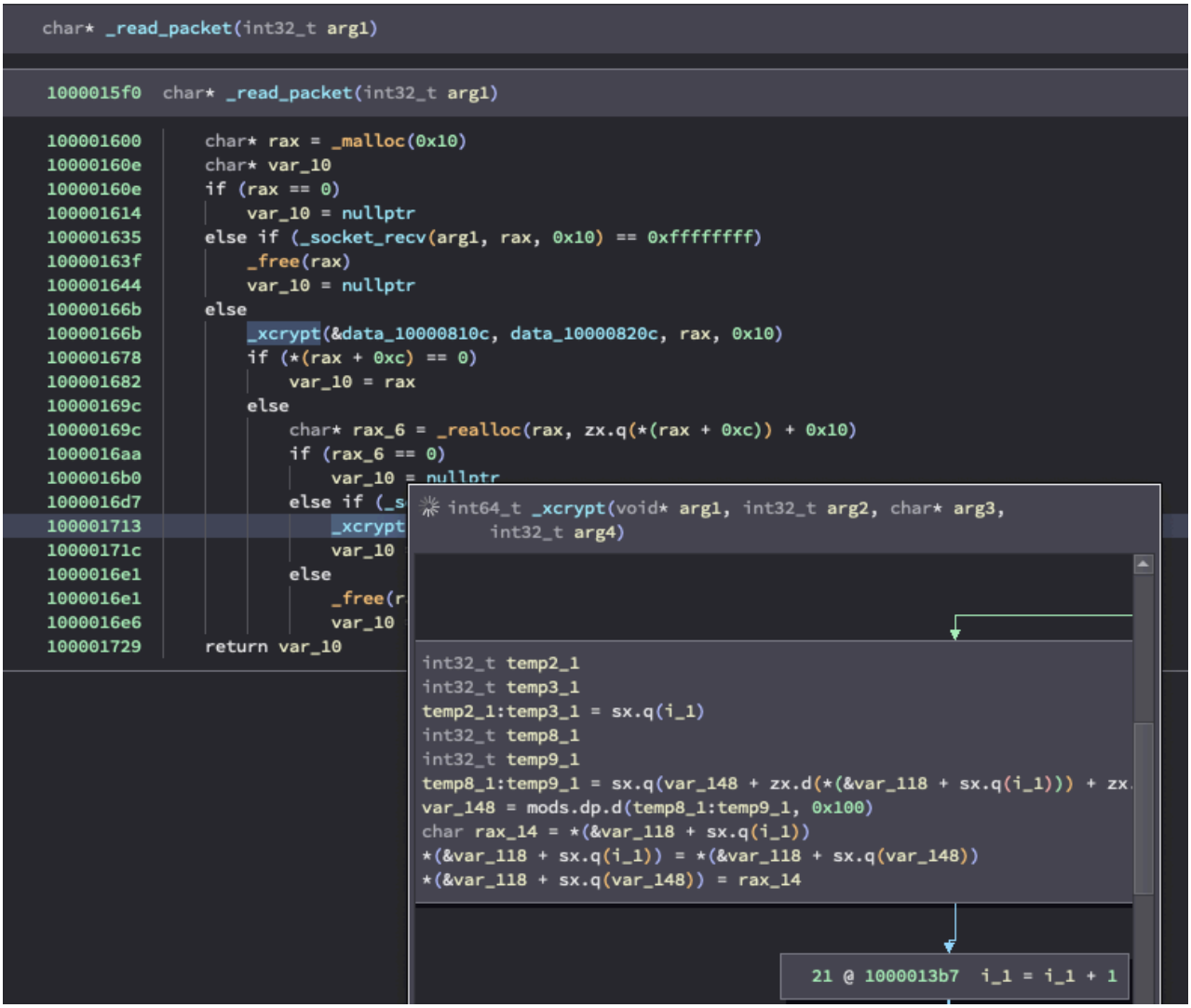
```
_socket_close
_socket_connect
_socket_recv
_socket_send
_wait_read
_write_packet
_write_packet_value
_xcrypt
```

So two things happened from this point for me - I load the binary up in Binary Ninja or Ghidra, and at the same time, I started running queries on some of the unique strings and kicked off a retrohunt with a rule very similar to the following. I ignored some super unique strings like `xcrypt`, `authchannel`, or `load_config` to do exactly what I accomplished - finding a potentially related family!

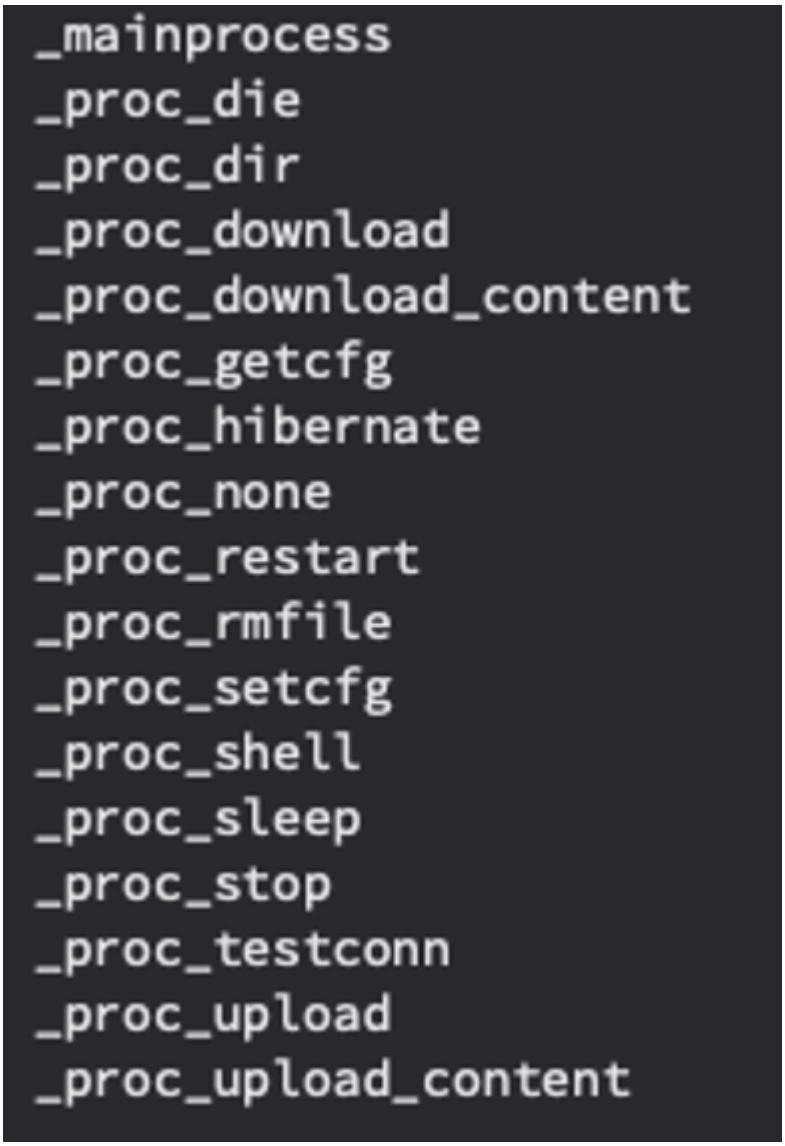
```
rule APT_NK_TA444_SpectralBlur_SockRacket_Overlap
{
        meta:
                author = "Greg Lesnewich"
                description = "track overlaps across KandyKorn/SockRacket and
SpectralBlur"
                date = "2023-08-21"
                version = "1.0"
                hash =
"0753859738620c7394f04220e273974982203a6ea1c2a30247149a9c8ff07037" //SockRacket
                hash =
"1d6cf7159c8dd98299798b0985f62dd15cb2e64550cd57a9e747dc3bee5f46d8" //SockRacket
                hash =
"c99729c39d197dd774e6febab5ec33abdf31f4404b4ffadad553efb3aa86192d" //SockRacket
                hash =
"d2d60f678d0b881b3e079b46bdb813f9f7d8802a227aea46926e4bbd1838f9e5" //SockRacket
                hash =
"d57a2e0c42c63659d6c09fc593fd5d272aec75b3629d9993b760142c731a191d" //SockRacket
                hash =
"f91801b458d875cfe61f927d16202b3a853d07e89a66ca4663989878e94242ad" //SockRacket
                hash =
"6f3e849ee0fe7a6453bd0408f0537fa894b17fc55bc9d1729ae035596f5c9220" //SpectralBlur

        strings:
                $s_dylib = "/usr/lib/libSystem.B.dylib" ascii wide
                $s_string1 = "/dev/null" ascii wide
                $s_string2 = "SHELL" ascii wide
                $s_string3 = "/bin/sh" ascii wide
                $s_import1 = "inet_addr" ascii wide
                $s_import2 = "inet_ntoa" ascii wide
                $s_import3 = "socket" ascii wide
                $s_import4 = "socket" ascii wide
                $s_import5 = "gethostbyname" ascii wide
                $s_import6 = "getpwuid" ascii wide
                $s_import7 = "kill" ascii wide
                $fp1 = "ftp" nocase ascii wide
                $fp2 = "kermit" nocase ascii wide
        condition:
                (
                uint32(0) == 0xfeedface or // Mach-O MH_MAGIC
                uint32(0) == 0xcefaedfe or // Mach-O MH_CIGAM
                uint32(0) == 0xfeedfacf or // Mach-O MH_MAGIC_64
                uint32(0) == 0xcffaedfe or // Mach-O MH_CIGAM_64
                uint32(0) == 0xcafebabe or // Mach-O FAT_MAGIC
                uint32(0) == 0xbebafeca    // Mach-O FAT_CIGAM
                )
                and filesize < 3MB and all of ($s_*) and none of ($fp*)
}
```

As the retro hunt was running, I started my analysis. And, thankfully, the operators kept function names intact when they compiled, so it made my life easy! SpectralBlur is a moderately capable backdoor, that can upload/download files, run a shell, update its configuration, delete files, hibernate or sleep, based on commands issued from the C2.

It communicates via sockets wrapped in RC4



The available commands listed under the term `proc`



And once the retro came back, we can see some similarities from sound familiar to the KandyKorn article you read from Elastic?? But these feel like families developed by different folks with the same sort of requirements.

I haven’t the time or skill to fully reverse SpectralBlur so if anyone in the community is keen on it, go for it!! But that XCrypt function looks unique enough to sig on along with some of the function names.

```
rule APT_NK_TA444_SpectralBlur
{
        meta:
                author = "Greg Lesnewich"
                description = "track the SpectralBlur backdoor"
                date = "2023-08-21"
                version = "1.0"
                hash =
"6f3e849ee0fe7a6453bd0408f0537fa894b17fc55bc9d1729ae035596f5c9220"
                DaysofYARA = "3/100"

        strings:
                $xcrypt1 = {
                        99                  // cdq
                        f7 [4-8]            // idiv    dword [rbp-0x11c {var_124}]
                        8b [4-8]            // mov     eax, dword [rbp-0x14c
{var_154_1}]
                        48 63 d2            // movsxd  rdx, edx
                        0f b6 0c 11         // movzx   ecx, byte [rcx+rdx]
                        01 c8               // add     eax, ecx
                        b9 00 01 00 00      // mov     ecx, 0x100
                        99                  // cdq
                        f7 f9               // idiv    ecx
                }

                $xcrypt2 = {
                        8b 85 c4 fe ff ff        // mov     eax, dword [rbp-0x13c
{var_144_2}]
                        83 c0 01                 // add     eax, 0x1
                        b9 00 01 00 00           // mov     ecx, 0x100
                        99                       // cdq
                        f7 f9                    // idiv    ecx
                        [20-40]
                        01 c8                    // add     eax, ecx
                        b9 00 01 00 00           // mov     ecx, 0x100
                        99                       // cdq
                        f7 f9                    // idiv    ecx
                }

                $symbol1 = "xcrypt" ascii wide
                $symbol2 = "_proc_die" ascii wide
                $symbol3 = "_proc_dir" ascii wide
                $symbol4 = "_proc_download" ascii wide
                $symbol5 = "_proc_download_content" ascii wide
                $symbol6 = "_proc_getcfg" ascii wide
                $symbol7 = "_proc_hibernate" ascii wide
                $symbol8 = "_proc_none" ascii wide
                $symbol9 = "_proc_restart" ascii wide
                $symbol10 = "_proc_rmfile" ascii wide
                $symbol11 = "_proc_setcfg" ascii wide
                $symbol12 = "_proc_shell" ascii wide
                $symbol13 = "_proc_sleep" ascii wide
                $symbol14 = "_proc_stop" ascii wide
                $symbol15 = "_proc_testconn" ascii wide
                $symbol16 = "_proc_upload" ascii wide
```

```
                $symbol17 = "_proc_upload_content" ascii wide
                $symbol18 = "_sigchild" ascii wide

                $string1 = "/dev/null" ascii wide
                $string2 = "SHELL" ascii wide
                $string3 = "/bin/sh" ascii wide
                $string4 =
{2573200a2573200a2573200a2573200a2573200a2573200a2573200a257320} // %s with
repeating new lines string
        condition:
                //(
                //uint32(0) == 0xfeedface or // Mach-O MH_MAGIC
                //uint32(0) == 0xcefaedfe or // Mach-O MH_CIGAM
                //uint32(0) == 0xfeedfacf or // Mach-O MH_MAGIC_64
                //uint32(0) == 0xcffaedfe or // Mach-O MH_CIGAM_64
                //uint32(0) == 0xcafebabe or // Mach-O FAT_MAGIC
                //uint32(0) == 0xbebafeca    // Mach-O FAT_CIGAM
                //) and
                (any of ($xcrypt*) or 4 of ($symbol*) or (all of ($string*)))
}
```

Wrapping Up

TA444 keeps running fast and furious with these new MacOS malware families. Looking for similar strings lead us to link SpectralBlur and KandyKorn (which were further linked to TA444 after more samples turned up, and eventually, a phishing campaign hit our visibility that pulled down KandyKorn). So knowing your Macho stuff will help track emerging DPRK capability if that is your interest!