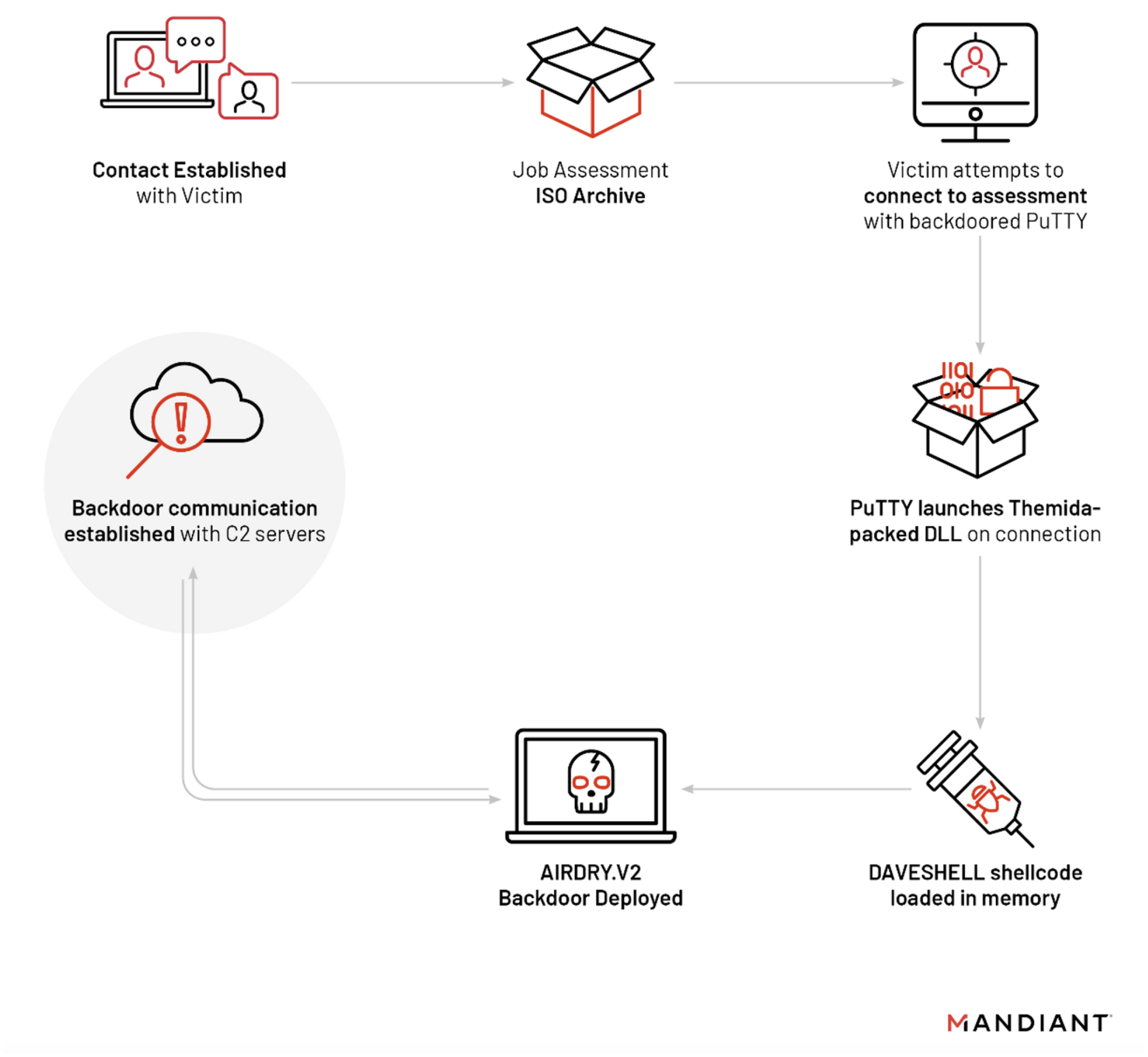
It's Time to PuTTY! DPRK Job Opportunity Phishing via WhatsApp

mandiant.com/resources/blog/dprk-whatsapp-phishing

In July 2022, during proactive threat hunting activities at a company in the media industry, Mandiant Managed Defense identified a novel spear phish methodology employed by the threat cluster tracked as UNC4034. Mandiant has identified several overlaps between this group and those we suspect have a North Korea nexus.

UNC4034 established communication with the victim over WhatsApp and lured them to download a malicious ISO package regarding a fake job offering that led to the deployment of the AIRDRY.V2 backdoor through a trojanized instance of the PuTTY utility.


Contact Established with Victim
Job Assessment ISO Archive
Victim attempts to connect to assessment with backdoored PuTTY
PuTTY launches Themida-packed DLL on connection
DAVESHELL shellcode loaded in memory
AIRDRY.V2 Backdoor Deployed
Backdoor communication established with C2 servers
MANDIANT

The Managed Defense Threat Hunting Mindset

One of the cornerstones of the Mandiant Managed Defense service offering is its proactive threat hunting program that protects our customers from advanced threat actor's tools, tactics and techniques that bypass traditional detection mechanisms. Managed Defense threat hunters leverage Mandiant's deep adversary research and exposure to threat actor behaviors to continually enhance and expand our threat hunting capabilities.

This activity was identified by our Mandiant Intelligence: Staging Directories mission, which searches for anomalous files written to directories commonly used by threat actors.

The techniques used by UNC4034 in this compromise, along with the techniques used in countless intrusions investigated by Mandiant, are used to continuously develop and refine threat hunting hypotheses within Managed Defense. These provide high fidelity and actionable leads that are informed by evolving threat actor tradecraft.

Initial Lead

The initial lead was a file downloaded to the host named `amazon_assessment.iso`. ISO and IMG archives have become attractive to threat actors because, from Windows 10 onwards, double-clicking these files automatically mounts them as a virtual disk drive and makes their content easily accessible. This reduces the effort needed to view the embedded files compared to other formats such as

RAR archives. Detecting malicious IMG and ISO archives served via phishing attachments is routine for Mandiant Managed Defense. The payloads contained within such archives range from commodity malware to advanced backdoors like the sample analyzed in this blog post.

Phishing Lure

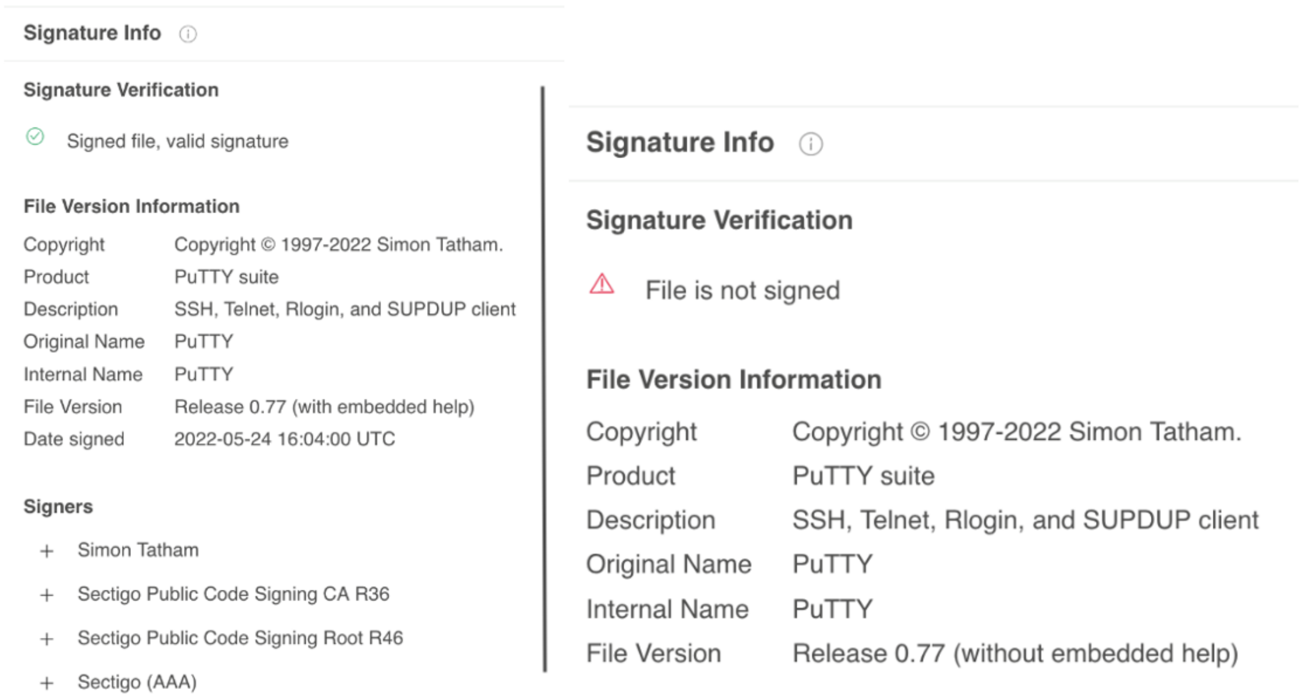
Mandiant Managed Defense performed an investigation on the host to determine how the file `amazon_assessment.iso` was created. Based on the available data, Mandiant assesses that UNC4034 initiated communication with the victim by offering them a job opportunity at Amazon via email. Subsequently, UNC4034 communicated with them over *WhatsApp* and shared the file `amazon_assessment.iso` , which the user downloaded using the web version of *WhatsApp*.

The `amazon_assessment.iso` archive held two files: an executable and a text file. The text file named `Readme.txt` had connection details for use with the second file: `PuTTY.exe` .

Figure 1: Contents of `Readme.txt`

```
Server:      137.184.15[.]189
User: test
Pass: [Redacted by Mandiant]
```

PuTTY is an open source SSH and Telnet client. Legitimate, compiled versions of the tool downloaded from the publisher's website will have a valid digital signature. However, as shown in Figure 2, the `PuTTY.exe` binary in the malicious archive does not have a digital signature



Signature Info

Signature Verification

✓ Signed file, valid signature

File Version Information

Copyright	Copyright © 1997-2022 Simon Tatham.
Product	PuTTY suite
Description	SSH, Telnet, Rlogin, and SUPDUP client
Original Name	PuTTY
Internal Name	PuTTY
File Version	Release 0.77 (with embedded help)
Date signed	2022-05-24 16:04:00 UTC

Signers

+ Simon Tatham
+ Sectigo Public Code Signing CA R36
+ Sectigo Public Code Signing Root R46
+ Sectigo (AAA)

Signature Info

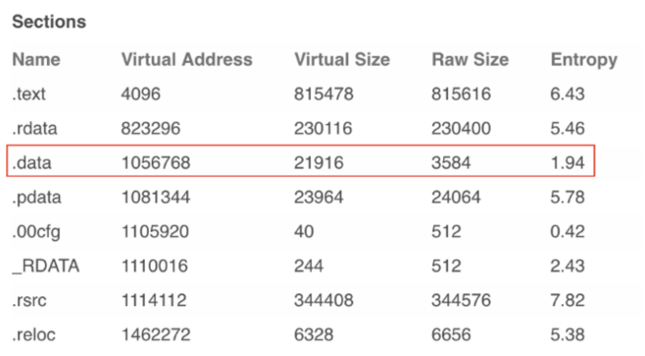
Signature Verification

⚠ File is not signed

File Version Information

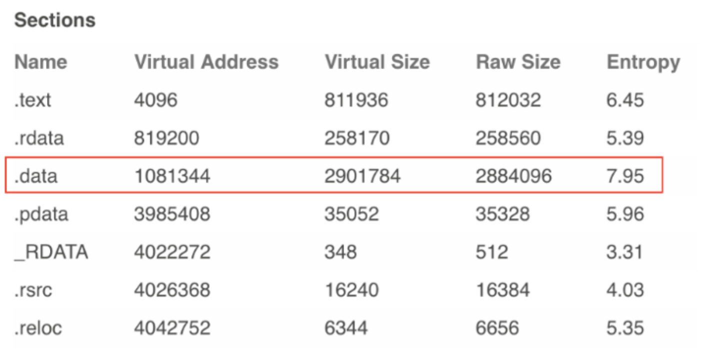
Copyright	Copyright © 1997-2022 Simon Tatham.
Product	PuTTY suite
Description	SSH, Telnet, Rlogin, and SUPDUP client
Original Name	PuTTY
Internal Name	PuTTY
File Version	Release 0.77 (without embedded help)

Figure 2: Digital signatures of the officially distributed PuTTY utility (left) and the malicious version (right)
The size of the PuTTY binary downloaded by the victim is also substantially larger than the legitimate version. Upon closer inspection, it has a large, high entropy `.data` section in comparison to the officially distributed version (Figure 3). Sections like these are typically indicative of packed or encrypted data.



Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy
.text	4096	815478	815616	6.43
.rdata	823296	230116	230400	5.46
.data	1056768	21916	3584	1.94
.pdata	1081344	23964	24064	5.78
.00cfg	1105920	40	512	0.42
_RDATA	1110016	244	512	2.43
.rsrc	1114112	344408	344576	7.82
.reloc	1462272	6328	6656	5.38



Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy
.text	4096	811936	812032	6.45
.rdata	819200	258170	258560	5.39
.data	1081344	2901784	2884096	7.95
.pdata	3985408	35052	35328	5.96
_RDATA	4022272	348	512	3.31
.rsrc	4026368	16240	16384	4.03
.reloc	4042752	6344	6656	5.35

Figure 3: Comparing the .data section in the officially distributed PuTTY utility (left) and the malicious sample (right)
The suspicious nature of the `PuTTY.exe` embedded in the ISO file prompted Managed Defense to perform a deeper investigation on the host and the file itself.

The execution of the malicious PuTTY binary resulted in the deployment of a backdoor to the host. The deployed backdoor is an evolution of the malware family Mandiant tracks as AIRDRY. Mandiant Managed Defense successfully investigated the compromise and contained the host before follow-on activity resulting from the deployed backdoor could occur. While Mandiant detected and responded to the compromise on 2022-07-05, the same PuTTY executable was seen on VirusTotal as early as 2022-06-27.

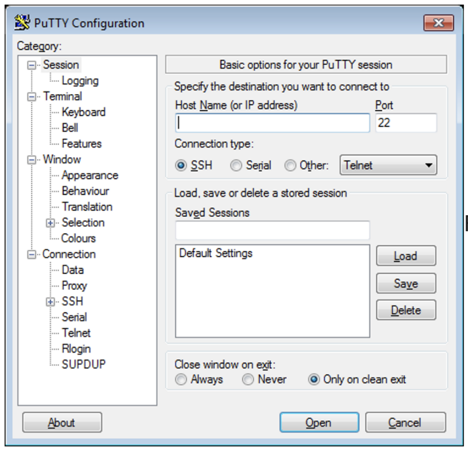
In addition, Mandiant discovered a second ISO archive named `amazon_test.iso` uploaded to VirusTotal on 2022-06-17. Mandiant found the second archive by pivoting from the IP address contained in the `Readme.txt` file. This ISO file also had a `Readme.txt` with the same IP address and a similar trojanized PuTTY executable.

Both ISO files found by Mandiant appear to use the same lure theme by posing as a recruitment assessment for Amazon. They also contained similarly constructed malware that resulted in a final AIRDRY.V2 backdoor payload.

Malware Analysis

Trojanized PuTTY

The executable embedded in each ISO file is a fully functional PuTTY application (Figure 4) compiled using publicly available PuTTY version 0.77 source code.



Port: 22
Connection type: [✓ SSH] Serial Other: Telnet
Close window on exit: Always Never [✓ Only on clean exit]

Figure 4: PuTTY interface displayed when executing the malicious samples

Each sample contains malicious code that writes an embedded payload to disk and launches it. However, the malicious code was inserted at different locations in each of the trojanized PuTTY samples. In the PuTTY sample discovered by Mandiant, the code resides in the `connect_to_host` function, which is in the source file `putty-0.77\ssh\ssh.c`. To trigger the code, the user must attempt an SSH connection to the host IP address provided in the `Readme.txt` file (Figure 1).

In the PuTTY sample discovered on VirusTotal, the malicious code was inserted into the `ssh2_userauth_process_queue` function (source file: `putty-0.77\ssh\userauth2-client.c`). The code resides in the part of the function responsible for performing password authentication, as opposed to other methods such as public key or keyboard-interactive authentication. Once the user establishes a connection and enters their username and password, the malicious code is executed regardless of the authentication result. These execution guardrails are likely an attempt by UNC4034 to avoid unnecessarily dropping the next stage of their malware.

The part of the malicious code that drops and executes a payload is nearly identical between the two samples. The legitimate Windows executable `C:\Windows\System32\colorcpl.exe` is copied to the new directory `C:\ProgramData\PackageColor` and the embedded payload is written to `C:\ProgramData\PackageColor\colorui.dll`. The PuTTY binary observed in the compromise launches `colorui.dll` via DLL search order hijacking using the command shown in Figure 5.

Figure 5: Observed `colorcpl.exe` execution

```
C:\Windows\System32\cmd.exe /c start /b C:\ProgramData\PackageColor\colorcpl.exe
0CE1241A44557AA438F27BC6D4ACA246
```

In the VirusTotal sample, `cmd.exe` is not used to launch `colorcpl.exe`. Instead, the Windows API function `WinExec` launches the process shown in Figure 6.

Figure 6: VirusTotal sample execution

```
C:\ProgramData\PackageColor\colorcpl.exe C8E71F4613ABFCA10B6330C9
```

In both instances, the command-line argument passed to `colorcpl.exe` is not related to the legitimate function of the Windows executable. Instead, each argument is utilized by the malicious DLL as described below.

Persistence is established for `C:\ProgramData\PackageColor\colorcpl.exe` via `schtasks.exe`. A scheduled task named `PackageColor` executes `colorcpl.exe` at 10:30AM local time every day.

Malicious DLL

The `colorui.dll` samples were packed using the commercial software protector Themida. The unpacked samples contain file paths that reveal their purpose. An example path is shown in Figure 7.

Figure 7: Embedded file path that has the string "`ShellCodeLoader`"

```
W:\Develop\aTool\ShellCodeLoader\App\libressl-2.6.5\crypto\cryptlib.c
```

Both samples contain an identical shellcode payload named DAVESHELL. The payload is decrypted using a custom XOR-based algorithm with a dynamically generated key. The key is the result of concatenating the following strings:

1. Parent process name (`COLORCPL.EXE`)
2. Malicious DLL filename (`COLORUI.DLL`)
3. Command-line argument passed to `colorcpl.exe` by the PuTTY executable

The necessary decryption key for the VirusTotal sample is shown in Figure 8.

Figure 8: Decryption key for the PuTTY sample found on VirusTotal

```
COLORCPL.EXECOLORUI.DLLC8E71F4613ABFCA10B6330C96CA3D3B1
```

The inclusion of this key also serves as an anti-analysis mechanism: without the correct key, nothing of significance happens when the DLL is executed.

The command-line argument passed to `colorcpl.exe` also dictates how the decrypted shellcode is executed. Based on this argument, `colorui.dll` may execute the shellcode from within `colorcpl.exe` or inject it into a new instance of a legitimate Windows process. In the case of process injection, the injection target is chosen randomly between `credwiz.exe` or `iexpress.exe`.

The injected shellcode payload is DAVESHELL, a publicly available dropper in the form of shellcode that executes an embedded payload in memory. The embedded payload is a VMProtect-packed evolution of the AIRDRY backdoor.

AIRDRY.V2

The AIRDRY backdoor, also known as BLINDINGCAN, has been thoroughly documented in reports published by CISA and JPCERT. These earlier versions of AIRDRY supported numerous backdoor commands including file transfer, file management, and command execution. In the most recent version, however, the traditional backdoor commands have been removed in favor of a plugin-based approach that supports multiple communication modes. The details that follow address the evolution to AIRDRY.V2 and highlight similarities with previous versions.

The backdoor's configuration is AES-128 encrypted in CBC mode with the hard-coded key `KAA5M8MNDKLJB8PI`. An integrity check is performed on both the encrypted and decrypted configuration. The backdoor exits if the check fails.

The decrypted configuration contains the backdoor's communication mode. Supported modes are listed in Table 1 along with their corresponding internal class names, where applicable. All three modes are centralized around a class named `CSinSocket`.

Table 1: AIRDRY.V2 communication modes

Mode	Class Name	Description
HTTP	`CHTTP_Protocol`	Communication via HTTP
File	`CFileRW`	Communication via a file
SMB	N/A	Communication via SMB over a named pipe

The structure of the embedded configuration is dependent on the communication mode. The observed sample is configured to use HTTP. The structure of its 0x24B0-byte configuration is outlined in Table 2. No SMB or file mode versions of AIRDRY.V2 have been identified and thus their configuration structure is unknown.

Table 2: HTTP-mode configuration structure

Offset	Description
`0x0000`	Operation mode
`0x0002`	C2 URL count
`0x0006`	C2 URL #1
`0x020E`	C2 URL #2
`0x0416`	C2 URL #3
`0x061E`	C2 URL #4
`0x0826`	C2 URL #5
`0x0A2E`	Proxy enabled flag
`0x0A32`	Proxy server
`0x0C3A`	Proxy port
`0x0C3C`	Proxy credentials available
`0x0C40`	Proxy username
`0x0E48`	Proxy password
`0x1050`	Unknown
`0x1054`	Maximum beacon count
`0x1056`	Report if disk space is available

`0x105A`	Report if there is an active RDP session
`0x105E`	Beacon interval in seconds
`0x1060`	Start date and time (empty)
`0x1068`	System ID
`0x106C`	Unknown
`0x18A0`	`c:\windows\system32\cmd.exe`
`0x1AA8`	`%temp%`
`0x1CB0`	Unknown

The configuration layout above is like the one outlined in Appendix A of the JPCERT report. The maximum beacon count stored at offset `0x1054` reflects the number of times the backdoor attempts to connect to a command and control (C2) server before waiting the amount of time stored at offset `0x105E`. The current sample is configured to beacon up to five times before sleeping for 60 seconds.

By default, the backdoor is not configured to use a proxy server. It is also not configured to report the status of available disk space (offset `0x1056`) or the presence of an active RDP session (offset `0x105E`). However, these features could be enabled via a configuration update issued by a C2 server.

The configuration strings at offsets `0x18A0` and `0x1AA8` are not referenced in the code but have been seen in previous AIRDRY configurations. It's possible these strings remain in the configuration for use by a downloaded plugin.

The configuration value at offset `0x1060` reflects the date and time after which the backdoor should begin communicating with its C2 servers. This value is empty in the current configuration, which results in communication beginning immediately. However, the value can be updated via a command from a C2 server, which could result in the backdoor being inactive for a period of time.

The backdoor's C2 URL count is five; however, the configuration only has three distinct URLs (Figure 9).

Figure 9: Configured C2 URLs

```
hxxps://hurricanepub[.]com/include/include.php
hxxps://turnscor[.]com/wp-includes/contacts.php
hxxps://www.elite4print[.]com/support/support.asp
```

The backdoor issues an HTTP POST request to a randomly selected C2 URL. An example request is shown in Figure 10.

Figure 10: Example HTTP POST request

```
bbs=VkxXU1lPvBGI7BJ40K4=shD5nhF+2amakV9H&article=EAAAABIAAAAgAAAAAAAAAAAAAAAAAAAAgu
PGZC0NvqYJ/yy4qGzW98G5M6Ab5UDNTt3lna8k/O8=
```

The `bbs` and `article` field names in the request body are hard-coded. The decoded Base64 data assigned to the bbs field aligns closely with the format outlined in the JPCERT report, including the same custom RC4 implementation. Where AIRDRY.V2's request structure differs from its predecessor is the second field, which is used to send command-related data. Prior to being Base64 encoded, the `article` data is compressed using LZ4 and then encrypted with AES. The AES-256 key is derived using the SHA256 hash of a hard-coded 32-byte sequence.

AIRDRY.V2's file mode uses the same HTTP request format but writes each request to a file. The file path is built based on elements found in the backdoor's configuration. Because a file mode version of AIRDRY.V2 has not been identified, the purpose behind writing HTTP requests to a file has yet to be determined. In SMB mode, the backdoor communicates by sending an SMB2 WRITE request to a named pipe. The host name and pipe name are specified in the configuration.

Like previous AIRDRY backdoors, AIRDRY.V2 utilizes the value `0x2040` to request a command from a C2 server. Supported command IDs are listed in Table 3.

Table 3: Supported command IDs

Command ID	Description
`0x2009`	Upload basic system information
`0x2028`	Update the beacon interval based on a value provided by the C2 server
`0x2029`	Deactivate until new start date and time
`0x2031`	Upload the current configuration
`0x2032`	Update the configuration
`0x2037`	Keep-alive
`0x2038`	Update the beacon interval based on a value in the configuration
`0x2052`	Update the AES key used to encrypt C2 requests and configuration data
`0x2057`	Download and execute a plugin in memory

AIRDRY.V2 supports nine commands. Prior versions of AIRDRY supported nearly thirty commands. Of the nine commands, only two were not present in previous AIRDRY samples: `0x2052` and `0x2057`. Command `0x2052` can be used to update the 32-byte sequence used to derive the AES-256 key described above. Command ID `0x2057` represents the shift from a backdoor that supports numerous commands to the new plugin-based approach in AIRDRY.V2.

Downloaded plugins are executed in memory and provided with a structure that includes a copy of the decrypted configuration, the system ID, and an object that facilitates communication with the configured C2 servers. Notably the backdoor itself has the necessary logic to function as a plugin.

Threat Actor Spotlight: UNC4034

Mandiant identified several overlaps between UNC4034 and threat clusters we suspect have a North Korean nexus. The AIRDRY.V2 C2 URLs belong to compromised website infrastructure previously leveraged by these groups and reported in several OSINT sources. Of note is the prior use of this compromised infrastructure to deliver the AIRDRY backdoor via CUTELOOP downloaders embedded in malicious documents.

Based on the identified overlaps and the social engineering tactics used, Mandiant suspects this activity represents an extension of enduring “Operation Dream Job” campaigns that leverage a different attack chain with ISO files and trojanized binaries rather than weaponized documents. This is likely one of several malware delivery techniques being employed by North Korean actors after a target has responded to a fabricated job lure. Recent public reporting also details the usage of other social media platforms to pose as legitimate companies and post fake job advertisements that target cryptocurrency developers.

The use of ISO files has become increasingly popular in the delivery of both commodity and targeted malware. Mandiant has observed well-known actors, such as APT29, adopting the use of ISO files to deliver their malware.

Detection Opportunities

The investigation into this compromise revealed new leads and indicators to pivot from in future hunting efforts. An important caveat to these hunting leads is that they are not indicators of compromise. Instead, they are interesting activity that may call for further investigation by an analyst.

These include ISO and IMG archive files downloaded from sources such as *WhatsApp*, email providers, and cloud storage services. On *Windows 10* and later, the origin URL of a downloaded file is stored in the `Zone.Identifier` alternate data stream. Mandiant routinely observes archive files delivered as phishing attachments that originate from those sources.

The second hunting lead is the execution of `colorcpl.exe` from an unusual directory like `C:\ProgramData\PackageColor`. Most often, the executable runs from its standard location in `C:\Windows\System32` or `C:\Windows\SysWOW64`.

The third hunting lead found for future hunting efforts is the execution of `colorcpl.exe` with command-line arguments. By default, `colorcpl.exe` executes with no command line arguments. In this scenario, `colorcpl.exe` was executed with a command-line argument that consisted of hexadecimal characters. The argument was ultimately evaluated by a malicious DLL that was loaded into the process.

Mandiant consistently finds new and novel threat actor activity such as the one described in this blog post. To ensure our customers are protected, Mandiant's detection and threat hunting capabilities are continuously evolving and are fueled by our experiences on the frontlines.

Malware Family Definitions

DAVESHELL

DAVESHELL is shellcode that functions as an in-memory dropper. Its embedded payload is mapped into memory and executed.

AIRDRY

AIRDRY is a backdoor written in C++ that communicates via HTTP. Its capabilities include shell command execution, file transfer, secure file deletion, file management, process termination, and process enumeration.

AIRDRY.V2

AIRDRY.V2 is an evolution of the AIRDRY backdoor that supports three communication modes: HTTP, file-based, and SMB. AIRDRY.V2 does not support the extensive list of commands found in AIRDRY. Instead, its functionality is extended via downloaded plugins that are executed directly in memory. AIRDRY.V2 can also update its configuration.

CUTELOOP

CUTELOOP is a downloader written in C++ that retrieves payloads via HTTPS. Downloaded payloads are mapped into memory and executed. CUTELOOP expects a command-line argument that is used to decrypt the payload URL.

Technical Indicators

Host and Network-based Indicators

Malware Family	MD5	SHA256
ISO Attachment	90adcfdaead2fda42b9353d44f7a8ceb	8cc60b628bded497b11dbc04facc7b5d7160294cbe521764df1a9ccb219bba6b
ISO Attachment	6d1a88fefd03f20d4180414e199eb23a	e03da0530a961a784fbba93154e9258776160e1394555d0752ac787f0182d3c0
Trojanized PuTTY Dropper	8368bb5c714202b27d7c493c9c0306d7	1492fa04475b89484b5b0a02e6ba3e52544c264c294b57210404b96b65e63266
Trojanized PuTTY Dropper	18c873c498f5b90025a3c33b17031223	cf22964951352c62d553b228cf4d2d9efe1ccb51729418c45dc48801d36f69b4
Themida-Packed Dropper for DAVESHELL	c650b716f9eb0bd6b92b0784719081cd	aaad412aeb0f98c2c27bb817682f08673902a48b65213091534f96fe6f5494d9
Themida-Packed Dropper for DAVESHELL	4914bcbbe36dfa9d718d02f162de3da1	3ac82652cf969a890345db1862deff4ea8885fe72fb987904c0283a2d5e6aac4

Type	Value	Comment
IPv4 Address	137.184.15[.]189	IP address seen in `Readme.txt`
URL	https://hurricanepub[.]com/include/include.php	AIRDRY.V2 C2
URL	https://turnscor[.]com/wp-includes/contacts.php	AIRDRY.V2 C2
URL	https://www.elite4print[.]com/support/support.asp	AIRDRY.V2 C2
File	C:\ProgramData\PackageColor\colorcpl.exe	Microsoft binary used for DLL search order hijack
File	C:\ProgramData\PackageColor\colorui.dll	Themida packed dropper for DAVESHELL
Scheduled Task	Task Name: PackageColor	Persistence mechanism

MITRE ATT&CK Mapping

ATT&CK Tactic Category	Techniques
Initial Access	T1566.001: Phishing: Spearphishing Attachment T1566.003: Phishing: Spearphishing via Service
Execution	T1059.003: Command and Scripting Interpreter: Windows Command Shell T1053.005: Scheduled Task/Job: Scheduled Task
Persistence	T1574.001: Hijack Execution Flow: DLL Search Order Hijacking T1053.005: Scheduled Task/Job: Scheduled Task
Defense Evasion	T1574.001: Hijack Execution Flow: DLL Search Order Hijacking T1055.001: Process Injection: Dynamic-link Library Injection T1218: System Binary Proxy Execution T1620: Reflective Code Loading T1027.002: Obfuscated Files or Information: Software Packing
Command and Control	T1071.001: Application Layer Protocol: Web Protocols T1071.002: Application Layer Protocol: File Transfer Protocols T1132.001: Data Encoding: Standard Encoding T1573.001: Encrypted Channel: Symmetric Encryption T1573.002: Encrypted Channel: Asymmetric Encryption