vb 2023 LONDON

4 - 6 October, 2023 / London, United Kingdom

LAZARUS CAMPAIGNS AND BACKDOORS IN 2022-23

Peter Kálnai

ESET, Czechia

peter.kalnai@eset.com

ABSTRACT

The Lazarus group is an infamous North Korea-aligned threat actor that has been active since at least 2009. There are various types of campaigns attributed to Lazarus, based on toolset similarities, shared infrastructure, telemetry, or the cui bono principle, and we have been discovering many of these campaigns for years. In this paper we will discuss the group's recent attacks, or attempts, from the years 2022 and 2023, several of which have not previously been publicly disclosed. These include: decoy programming challenges against a Spanish aerospace company delivering a highly sophisticated RAT; Coinbase-themed decoys with both *Windows* and *macOS* payloads targeting individuals in South America; attempts to compromise banking entities in the United States and Tanzania via fake Signature Bank- and MUFG-themed job offers; an OpenSSL-based backdoor discovered in an agriculture-related entity in South Korea; a lure designed for *Linux* desktop users, submitted to *VirusTotal* from Georgia and related to the recent *3CX* supply-chain attack, and more.

In our study we will outline the relationship between recent Lazarus campaigns and their payloads delivered onto targeted systems. Usually there are many tools deployed in chains by these attackers: in the initial stages, various droppers, loaders, and simple downloaders are used to establish a stable foothold in the system, while in the later stages, full-featured RATs and more complex downloaders are involved. We will focus on these payloads, which are usually more sophisticated and harder to acquire. We dive deep into the attackers' custom implementation of the client-server model – the method of parsing server commands and the multi-step network authentication – as understanding and identification of these aspects may boost confidence in the attribution verdict.

Finally, we will note the diverse geographical distribution of the group's recent campaigns, and the fact that Lazarus operators have prepared native payloads to cover all the major platforms: *Windows*, *Linux* and *macOS*.

INTRODUCTION

The Lazarus group (also known as HIDDEN COBRA) is a threat actor that has been active since at least 2009. It is responsible for high-profile incidents such as the *Sony Pictures Entertainment* hack and tens-of-millions-of-dollar cyber heists in 2016, the WannaCryptor (aka WannaCry) outbreak in 2017, and a long history of disruptive attacks against South Korean public and critical infrastructure since at least 2011. The diversity, number and eccentricity in implementation of Lazarus campaigns define this group, alongside the fact that it performs all three pillars of cybercriminal activities: cyber espionage, cyber sabotage, and pursuit of financial gain.

In 2021, the US Department of Justice indicted North Korean citizens in connection with cyber-enabled financial (and cryptocurrency) crimes spread across 2014–2020 [1]. The North Korean regime itself considers cyberwarfare as an 'all-purpose sword' that guarantees relentless offensive capabilities [2].

The North Korean regime is reportedly involved in illicit economic activities. There is a study published by the Committee for Human Rights in North Korea on the country's criminal behaviour, including trade in amphetamine-type stimulants, production and distribution of high-quality counterfeit currency, trafficking in endangered species, and the manufacture of counterfeit cigarettes [3], [4]. Moreover, North Korea seems to orchestrate forced labour of its migrant workers on a global scale [5].

Despite the severe United Nations sanctions imposed on North Korea since 2006, the country still enjoys access to defence markets, and is involved in the sale of indigenous arms, materiel, and services to various state and non-state actors [6]. The US Defense Intelligence Agency (DIA), an intelligence agency under the Department of Defense, published an unclassified report on North Korea's defence and military strategy and tactics in 2021 [7]. It sheds some light on North Korea's nuclear weapons program, and missile development and testing, including of road-mobile short-range (SRBM), medium-range (MRBM), intercontinental (ICBM), and submarine-launched ballistic missiles (SLBM), both liquid- and solid-propelled.

When performing attribution, we don't have the intelligence means to identify the perpetrator of a cyber attack and that's not our aim either. To build a theory on Lazarus activity, our goal is to classify the group's new cyber attacks as significantly similar with those already attributed, using strong indicators expressed in terms of the malware toolset used, shared network infrastructure, and telemetry [8]. Additionally, we try to cluster various activity subtypes if we see borderlines between them. We provide the results of these efforts in the 'Campaigns' section. After recording many Lazarus attacks, we dare to say that their cyber activity reflects the North Korean state's motivations in the physical world, mostly due to the targeting of aerospace and defence sectors, and pursuing financial and cryptocurrency gain.

In [9] we summarized various aspects connected with the Lazarus group: the most notorious cases from the 2014-2018 period, the major toolset characteristics, and six previously unreported, Lazarus-linked cases that happened in the time frame of 2016-2018. Since then, the group's tactics, techniques, and procedures have evolved significantly. The 'TTPs' section in this paper summarizes various patterns that we extracted from the attacks: trojanizing of open-source projects like plug-ins for *Notepad++* or PDF viewers, the use of valid code-signing certificates, the Rich Headers analysis of the development environment that produces their payloads, and an overview of encryption present in their malware.

The group's toolset has been developed constantly. In the 'Backdoors' section, we provide a catalogue of the most common pre-final payloads deployed by the attackers. Notable is the use of strongly encrypted, HTTP(S)-based network protocols for client-server communication by their downloaders and full-featured RATs.

CAMPAIGNS

According to *ESET* telemetry, Lazarus is active globally, to the point where the targeted companies and individuals come from all over the world; see Figure 1 in which the turquoise colour represents countries with targets, light for geographically larger countries and dark for smaller ones (victims of the widespread supply-chain attacks are excluded).

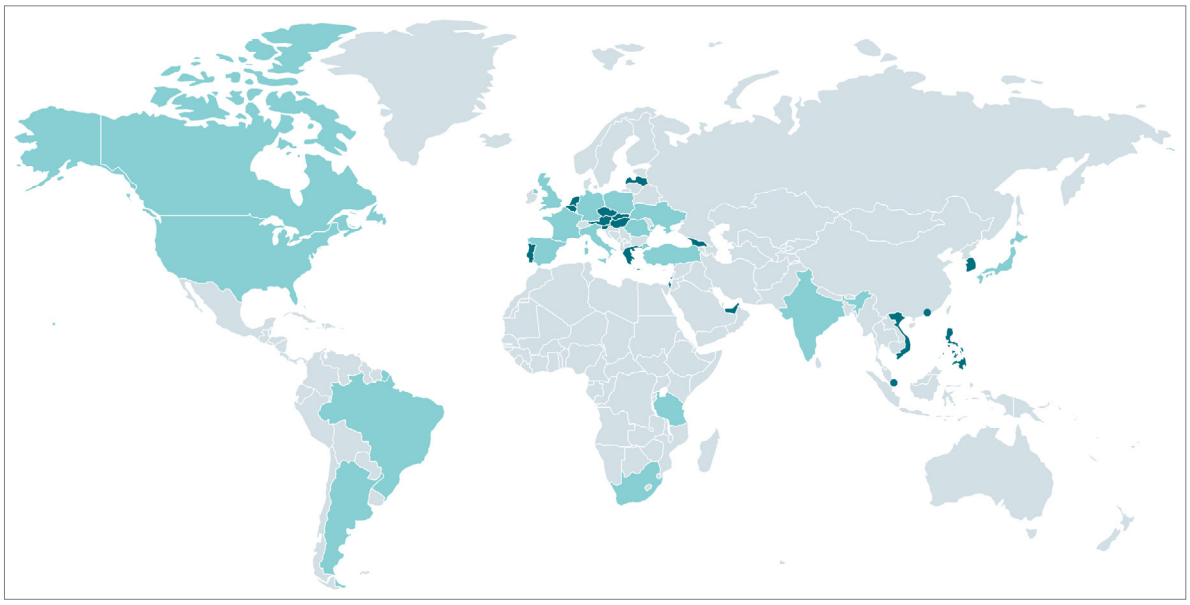

Figure 1: Targets recorded by ESET telemetry and VirusTotal submissions since 2020 are all over the globe.

Regular social engineering campaigns

There are known cases of Lazarus operators exploiting unknown vulnerabilities to gain initial access to their targets' networks [10]. However, in the majority of attacks they rely on compromising targeted systems via unsuspecting users and employees. This is typically achieved through a convincing story masquerading as an attractive but bogus job offer, important news from the crypto world, or a lucrative investment agreement, and delivered in varying forms, such as malicious *Word* documents, ZIP archives, optical disc images (ISO), and virtual hard disks (VHD).

Operation In(ter)ception

The initial report on this operation was published by *ESET* in June 2020 [11], but summarized attacks that had happened since September 2019. The initial compromise was established by luring the target with fictitious job offers from HR managers of leading organizations in aerospace and defence, such as *Collins Aerospace* and *General Dynamics*, followed by sending malicious files directly via *LinkedIn* messaging, or via a *OneDrive* link in an email.

Date	Country	Theme	Industry	C&C domain	Payloads
2022-01	Turkey	BAE Systems		`lm-career[.]com`	In(ter)ception downloaders
2022-01	Italy	Lockheed Martin	Pharmaceutical	`markettrendingcenter[.]com`	In(ter)ception backdoor
2022-01	Spain, India, Slovenia*	Lockheed Martin		`markettrendingcenter[.]com`	

2022-02	Ukraine	BAE Systems	Defence	`shopapppro[.]com`
				`techdesignshop[.]com`
				`designautocad[.]org`
				`shopwebstudio[.]com`
					In(ter)ception backdoor

2022-03	Turkey	Northrop Grumman	Defence	`topnewsagent[.]com`
				`designlabshop[.]com`
				`dailynewsagent[.]com`
				`freewaremail[.]com`
					In(ter)ception backdoor, BackbitingTea

2022-03	Brazil*	Solana		`webhosttech[.]org`	
2022-07	Argentina, Brazil	Coinbase	Cryptotrading	`concrecapital[.]com`	In(ter)ception downloaders
2022-10	Vietnam*	Crypto.com			
2022-12	USA		Finance	`cloudfly[.]org` `timecashlive[.]com`	In(ter)ception downloader BackbitingTea

Table 1: Operation In(ter)ception summary. The asterisks indicate a VirusTotal submission.

The main goals of the attackers were cyberespionage and exfiltration of the target's sensitive data. When the primary goal was successfully completed, the attackers additionally tried to monetize the network access through business email compromise [12]. Table 1 summarizes Operation In(ter)ception attacks, as observed by *ESET* telemetry and in *VirusTotal* submissions since 2022.

DangerousPassword campaigns

The name of these attacks was coined by *ThreatBook* in a November 2019 report [13], based on a malicious LNK downloader that was delivered to the target, containing a password that was required to unlock a malicious document with content of interest. The inception of the attacks goes back even further, as shown in a report by JPCERT/CC from July 2019 [14]. Since then, many blog and *Twitter* posts have discussed these campaigns, e.g. by *Kaspersky*, which denotes them as the BlueNoroff cluster of activity [15].

The typical trait of this activity is the use of various non-native file formats at the initial stages of the attack: LNK files, VBScripts, PowerShell, or JavaScript. In later stages, the attacker typically deploys native payloads like the WebbyTea downloader and the BackbitingTea RAT (also called SnatchCrypto).

As the majority of the targets are from the financial sector and the world of cryptocurrencies and blockchain technology, we conclude that the main goal is the theft of financial and crypto assets. See Table 2, which shows a summary of DangerousPassword attacks as observed by *ESET* telemetry and in *VirusTotal* submissions since 2022. The theme contains either a filename of the lure or a lucrative brand that the attackers falsely represented to attract the target.

Date	Country	Theme	Industry	C&C domain	Payloads
2022-02	United Kingdom*	New Salary Adjustments		`datacentre[.]center`	
2022-02	Ukraine*	зарплата_2022020708129312 (translation: salary)		`shopappppro[.]com`	
2022-02	Ukraine	JP Morgan Chase	Digital investment	`datacentre[.]center`	
2022-02	Poland	Pensja Adiustacja 202202101019 (translation: Salary Adjustment)	Defence	`shopappppro[.]com`	
2022-02	France	Lettre de veille international GICAN n°37 – février 2022 (translation: International watch letter GICAN n°37 - February 2022)	Defence	`shopappppro[.]com`	
2022-02	Georgia*	Digital Asset Fund (DACM)		`doc.filesaves[.]cloud`	
2022-03	Canada		Finance	`word.azure-company[.]net`	WebbyTea, BackbitingTea
2022-07	Israel	New Salary Adjustment	Blockchain	`www.googlesheet[.]info`	
2022-07	India*	Ledger Nano S&X		`dps.shconstmarket[.]com`	
2022-09	Latvia		Blockchain	`docs.azurehosting[.]co`	BackbitingTea, SecondHandTea
2022-09	Tanzania	MUFG	Finance	`verify.azure-protect[.]online`	
2022-12	Poland	Signature Bank	Finance	`doc.gdocshare[.]one`	BackbitingTea
2023-02	Netherlands	DocuSign – Executed Version (Secured)	Cryptotrading	`cloud.mekongcapital[.]net`	SecondHandTea
2023-02	United Arab Emirates*	PDF Reader for Winforms + DOJ Report on Bizlato Investigation		`safe.doc-share[.]cloud`	WebbyTea
2023-05	United States*	Internal PDF viewer for macOS 3.0 + Jump Crypto Investment Agreement		`cryptyk.ddns[.]net`	RustBucket for macOS [16]

Table 2: Summary of DangerousPassword attacks since 2022. The asterisks indicate a VirusTotal submission.

Operation DreamJob

The name of these campaigns was introduced in a blog post by *ClearSky* in August 2020 [17]. But the very first signs of them can be traced back to January 2019 [18]. Since then, many additional cases have been reported publicly: [19], [20], [21], [22], [23], [24], [25], [26] (as DeathNote).

The most common trait is the use of trojanized software that purports to assist the attackers' target with the hiring process, namely PDF viewers, remote access tools, and coding challenges. BlindingCan is the flagship RAT deployed to compromised victims, but there are additional new and sophisticated malware projects at the attackers' disposal, namely ScoringMathTea and LightlessCan (we discuss these specific payloads in the 'Backdoors' section).

The attackers' goals depend on the targeted sector. In particular for aerospace, our telemetry and the malware features indicate that the exfiltration of large amounts of sensitive data, like the company's internal knowledge base, is of the highest priority for the malware operators, and we conclude that the same applies to the defence sector. On the other hand, when the targeted network belongs to a technology or data company, then the attackers are after huge financial gain, which is performed via some form of business email compromise, or meddling with internal accounting systems. Finally, we don't yet have any conclusion as to why in this operation Lazarus attacks western media (note the *Comcast* and *Disney* themes), but these types of targets are less frequent.

Table 3 shows a summary of Operation DreamJob attacks, as observed by *ESET* telemetry and in VirusTotal submissions since late 2021. The theme contains the type of GUI delivered to the target (if present), together with the brand that the attackers falsely represented.

Date	Country	Theme	Industry	Payloads
2021-09	Belgium	Amazon	Media	OfficeCertTea
2021-10	Netherlands	Amazon	Aerospace	BlindingCan, FudModule, HTTP(S) uploader
2022-03	Spain	Coding challenges + Meta	Aerospace	BlindingCan, miniBlindingCan, LightlessCan, NickelLoader
2022-03	South Africa	SecurePDF + Airbus		ImprudentCook
2022-06	Italy*	MµPDF + Disney		
2022-10	Portugal*, Germany*	SecurePDF + Airbus		ScoringMathTea
2022-11	Netherlands*	TightVNC		WinInetLoader, NickelLoader
2023-01	India	SecurePDF + UltraVNC + Accenture	Tech & Data	miniBlindingCan, BlindingCan, LightlessCan
2023-02	Poland	SecurePDF + Boeing	Defence	ScoringMathTea, ImprudentCook
2023-02	*	Comcast		miniBlindingCan
2023-03	Georgia*	HSBC		SimpleTea for Linux
2023-05	Hungary*	TightVNC + Rosatom		WinInetLoader

Table 3: Instances of Operation DreamJob since late 2021. The asterisks indicate VirusTotal submissions.

Other attacks

Supply-chain attacks

A software supply-chain attack occurs when an attacker infiltrates a software vendor's network and employs malicious code to compromise the software before the vendor distributes it to their customers (definition by [27]). Lazarus has proved its capability in performing such advanced attacks several times in the past. The first publicly known case was reported in 2020 [28], when a legitimate download verification tool called *WIZVERA Veraport*, used mostly in South Korea, prompted its users to install malware via a browser plug-in after visiting compromised web servers. The deployed malware was validly signed by the code-signing certificate mentioned in Table 5. More recently, the following are the publicly known supply-chain attacks attributed to the Lazarus group.

Trading Technologies

Around February 2022, a public website of *Trading Technologies*, a company with around 31,000 followers on *LinkedIn*, was allegedly compromised via an unknown method, which was reported in March 2022 by *Google TAG* [29] (note the website `www.tradingtechnologies[.]com` is mentioned among the IoCs). Interestingly, despite the publicly disclosed fact that something strange was happening with *Trading Technologies*' website, it didn't prevent the chain of events that followed.

In April 2023, *Mandiant* published its report on the *3CX* supply-chain incident [30]. This report extended information about the *3CX* compromise, and discussed a preceding supply chain attack that originated from *Trading Technologies*' software installer package called *X_TRADER*, specifically version `r7.17.90p608`. Even though the package's metadata states March 2020 as the creation date, it contains a tainted data file and a main executable, both with timestamps from early November 2021, which is several months before *Google*'s initial warning from March 2022.

The technical analysis of another instance of the same version of the malicious installer was provided by *BroadCom* in a blog post on 21 April 2023 [31]. This installer is malicious too, but the tainted data file and the main executable have the same timestamps as the rest of the files: March 2020.

From *ESET*'s telemetry, we saw two occurrences of the malicious installer reported by *Mandiant*: first in August 2022, in a corporate network in the UK, and second in December 2022, at a small US consulting/trading firm.

In addition to these publicly known installers, we discovered in our telemetry an additional trojanized executable in an execution chain started by a *Trading Technologies* installer binary. The trojanized executable was a validly signed `setup.exe` that was executed in the network of a construction company in Malaysia in August 2022. Although we were not able to find out what specific *X_TRADER* installer build was trojanized in this case, the filename of the next-stage payload differed from the one used in the *Mandiant* and *BroadCom* reports (`TT User Setup-ja.mst` was used instead of `X_TRADER-ja.mst`).

3CX

We reported our account of the events related to the *3CX* incident in a blog post on 20 April 2023 [32]. As far back as September 2018, *Trading Technologies* had announced its plan to decommission the *X_TRADER* software in early 2020 and replace it with a new HTML5-based trading platform [33]. Unfortunately, the end-of-life state of the software did not prevent a *3CX* employee having it installed on a corporate machine, as we learned from the official incident response by *Mandiant* [30]. This was the first-ever recorded case of two linked supply-chain attacks, one enabling the other.

Exploitations

Initech INISAFE CrossWeb EX V3

This vulnerability, which has not been assigned a CVE ID, was reported by *AhnLab* in April 2022 [34]. In the IoCs section, they revealed multiple samples associated with the attacks, but we identified only a dropper of wAgentTea and the Racket downloader [35]. According to *AhnLab*'s report, a legitimate INISAFE CrossWeb EX V3 process was injected by a malicious DLL called `SCSKAppLink.dll` that originated from a malicious HTM website (see also [36] for an additional case). The filename is a disguise for a component of the legitimate *SoftCamp*'s *Secure KeyStroke* application, which is required to access certain government and banking websites in South Korea.

ESET telemetry recorded a malicious `SCSKAppLink.dll` deployed against South Korean targets in June 2021. The initial compromise was made via what appears to be a watering-hole attack on the website of a local golf club not far from Seoul. An early version of ThreatNeedleTea, using an RC4-based network protocol, was delivered to the targets as well.

CVE-2021-26606 (MagicLine4NX) [37]

In late October 2022, researchers from *AhnLab* reported [38] that the Lazarus group had started exploiting the CVE-2021-26606 vulnerability to gain additional access to the internal network of its South Korean victims.

This vulnerability is a buffer overflow that affects *Dream Security*'s *MagicLine4NX* software (a South Korean endpoint security solution) versions 1.0.0.17 and earlier. Exploitation allows attackers to remotely execute arbitrary code on targeted systems. In the reported attacks, the *MagicLine4NX* process was used to inject a malicious thread into `ftp.exe`. Again, plenty of samples are mentioned in the IoCs, from which our team was able to identify FudModule [24] and its loaders as the payloads among them.

In our telemetry from 2022, we identified several `ftp.exe` processes running shellcode, present exclusively among South Korean targets. The delivered payloads were wAgentTea dropped by a trojanized 64-bit liblzma v5.2.5 data compression library, PostNapTea dropped by a trojanized lecui library, and multiple instances of ThreatNeedleTea.

Unknown initial access

There are also attacks targeting South Korea where we were unable to identify the intrusion method:

- Unknown entity in January 2022. Deployed malware: wAgentTea.
- Newspaper in February 2022. Deployed malware: PostNapTea.
- Unknown entity in July 2022. Deployed malware: VMProtect-ed FudModule.

- An agriculture-related entity targeted in February 2023. Deployed malware: PostNapTea.
- Semiconductor industry in April 2023. Deployed malware: wAgentTea.

TTPs

Since there are many attacks attributed to the group, it's possible to see high-level patterns in its behaviour. In this section, we provide a summary of which open-source projects were chosen to be trojanized, what encryption methods are implemented in the payloads, and which valid code-signing certificates were observed.

Execution chain

To deliver malware and its configurations securely, the group usually arranges multiple stages of execution. This is a natural evolution as their most used payloads became exposed in previous attacks, and are now identified and detected. If they want to reuse them, a viable option is to store them in an encrypted state on the file system and precede their execution with a series of droppers and loaders. See Figure 2 for an example of a complex execution chain (note that four additional malicious helper executables are involved to load a RAT, the last step of the chain).

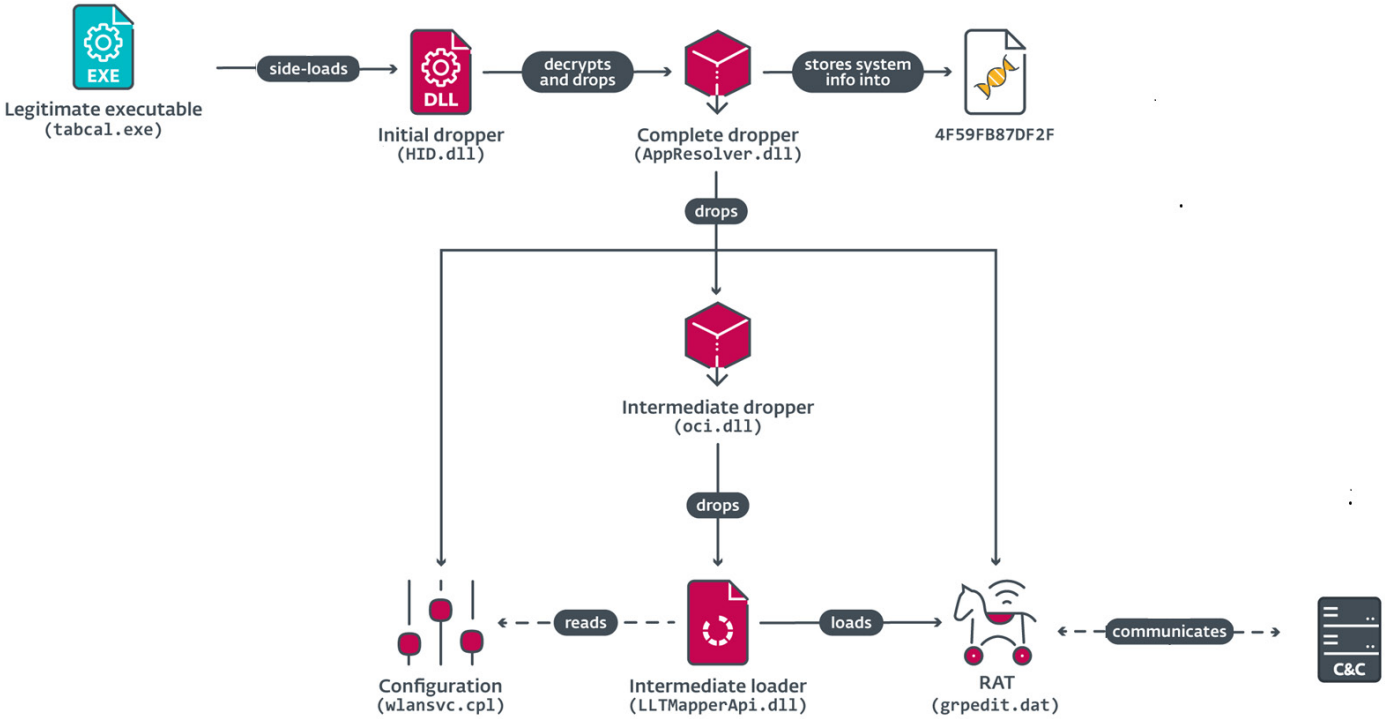

Figure 2: A complex execution chain used by Lazarus in recent Operation DreamJob campaigns.

Trojanized open-source projects

In recent years, Lazarus has started to incorporate its malicious code into open-source projects available online in hosting services like *GitHub*. The purpose is twofold: first, to trojanize a legitimate project into a dropper containing an embedded, encrypted next stage, possibly evading detection (note that only a fraction of the new executable is malicious); second, to have a seemingly legitimate GUI that serves various scenarios in social engineering attacks.

Plug-ins for Notepad++

We observed Lazarus extensively trojanizing plug-in projects for *Notepad++*, exclusively to use them as droppers for the next stage. All plug-ins are DLLs and a common feature is the presence of the following exported functions: `beNotified`, `getFuncsArray`, `getName`, `isUnicode`, `messageProc`, and `setInfo`. Interestingly, the attackers use these droppers for side-loading via a legitimate application, so these plug-ins also contain exports unrelated to *Notepad++*, but required by the loading executable. For example, the MZC8051 plug-in from the Spanish case in Q2 2022 [25] also contains the export `HidP_GetSpecificValueCaps` that is imported by `tabcal.exe`, its parent process. The attackers had to craft the source code of these DLLs carefully, as they didn't want to break this side-loading process. The list of *Notepad++* plug-ins that were used by Lazarus in campaigns since 2021 is shown in Table 4.

Plug-in	Developer	Description	Attack
NppExport 0.3.0	chcg	Export plug-in for Notepad++	[39]
NppAStyle 0.2.7	YWX	Artistic Style plug-in for Notepad++	VT [35]
ComparePlus 1.0.0	Pavel Nedev	File comparison plug-in for Notepad++	[26], [35]
FingerText 0.5.60, 0.5.61	erinata	Snippet plug-in for Notepad++	[40] Operation DreamJob in India Q1 2023
GOnpp 1.2.0.0	tike	Go programming language plug-in for Notepad++	[40]
Flashing-Tip	Tipikin Aleksandr	Notepad++ plug-in	[25]
MZC8051 0.1.1.0	Don HO	MZC8051 C compiler plug-in for Notepad++	[25]
LuaUtils 1.4.0.0	Charsi82	Lua plug-in for Notepad++	[25]
NppyPlugin	Jari Pennanen	General Python plug-in for Notepad++	[25]
FWDataViz 2.6.1.0	Shridhar Kumar	Fixed-width data visualizer plug-in for Notepad++	[22] [41]
GotoLineCol 2.4.2.0	Shridhar Kumar	Go to line, column plug-in for Notepad++	Operation DreamJob in India Q1 2023
Hex Editor Plugin 0.9.12	Jens Lorenz	Hex editor plug-in for Notepad++	Operation DreamJob April 2023

Table 4: A list of Notepad++ open-source plug-ins used by Lazarus in campaigns since 2021.

PDF readers

In this scenario, the attackers offer their target (at least) two files: a PDF viewer/reader and a PDF file. The target is instructed to open the provided PDF file with the provided PDF software. Because the PDF file is either incomplete or does not even have the proper format of a PDF document, opening it in any other PDF software leads to displaying an incomplete content or error message that the file is corrupt. In this way, the target may feel forced to follow the attackers' instructions and open the PDF in the provided viewer. This action triggers the malicious branch within the trojanized application (that is, by checking the MD5 hash of the PDF file against a hard-coded value). The proper content is displayed (decrypted from the PDF or downloaded from a C&C server), but, at the same time, the target's system is compromised, and the attackers are free to deliver additional payloads. Figure 3 shows an example of a full job description displayed to the target.

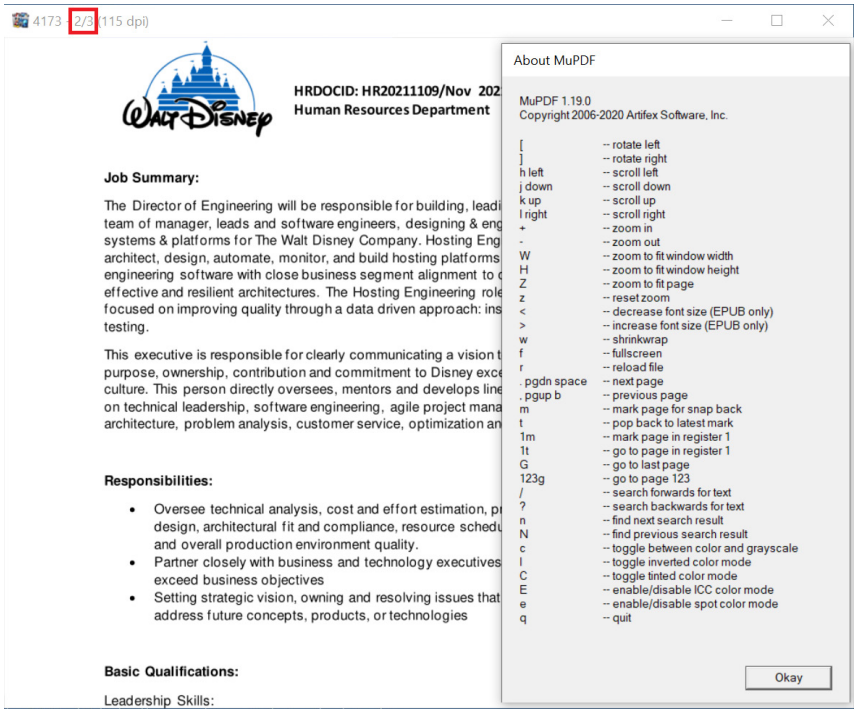

Figure 3: Full job description, starting on page 2, displayed in a trojanized MµPDF reader. The page shows no content in any other PDF software.

We have recorded several types of PDF software used by Lazarus. These are usually based on open-source apps available online:

- SecurePDF/SumatraPDF [42] versions 3.2 and 3.3, originally by Krystof Kowalczyk.
- MµPDF [43] version 1.19.0, originally by Artifex.
- Aloha PDF Reader, which is based on the open-source Tinker and UXReader PDF libraries [44] for Windows, both originally by Julius Oklamcak; the latter is in turn based on the PDFium library [45].
- PDF Viewer for WinForms, originally by DevExpress.
- Internal PDF Viewer 3.0 for macOS, based on [46] *Apple*'s built-in Quartz.PDFKit.

Remote access tools

In this scenario, the attackers require their target to connect to a remote computer – an action that's purportedly essential to continue with the hiring process. The victim receives a trojanized remote access tool together with connection details like an IP address or a username, and a password. The malicious branch is triggered when the target picks the corresponding connection in the provided GUI.

These network tools have been observed:

- PuTTY ([41])
- KiTTY ([23])
- UltraVNC (see Figure 4)
- TightVNC ([41]; also see Figure 4)
- LoginPortal, based on the ImGUI project [47]

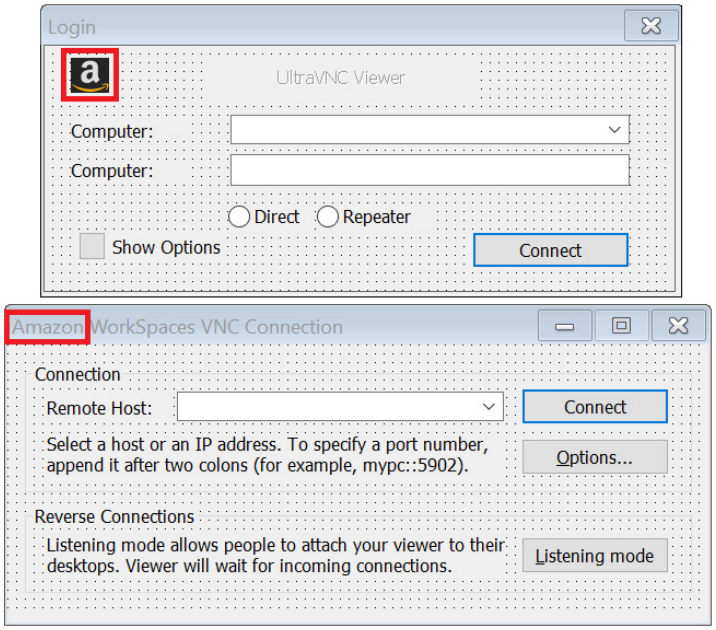

Figure 4: Amazon-themed dialogs of trojanized UltraVNC and TightVNC remote access tools.

Code-signing certificates

The group often uses code-signing certificates to sign native payloads. The main motivation is to make their operations look legitimate, as users and security solutions tend to trust signed software more than unsigned. In the following section, we will look more closely at how Lazarus has been using these in its operations.

Pre-2022

In this period, the attackers seemed to follow the same pattern regarding the code-signing certificates they used. There were multiple Lazarus operations with different, but similar, certificates. The repeating pattern was that they were issued to relatively unknown American or British companies, with no apparent sort of business that requires digital certificates, and with no online presence in general. Table 5 lists certificates used by Lazarus in various attacks and campaigns known before 2022. In the last column we state the type of campaign, together with a reference to a public report mentioning the certificate.

Regarding *macOS* systems, the attackers used valid code-signing certificates several times. A signed *Mac* executable disguised as a job description for *Coinbase* was uploaded to *VirusTotal* from Brazil on 11 August 2022. According to the timestamp, the bundle was signed on 21 July using a certificate issued in February 2022 to a developer named Shankey Nohria and team identifier `264HFWQH63`. The application is not notarized, and *Apple* revoked the certificate on 12 August. Even earlier, a signed *Mac* executable disguised as a job description for *Bitazu Capital* was submitted to *VirusTotal* from Singapore on 31 March 2021. It was signed using a certificate issued in late March 2021 to an organization presumably named 'Golden Book' with team identifier `H5YL5668C7` and using the Apple ID `goldenbook2021@icloud[.]com`. *Apple* revoked this certificate on 10 November 2021. In June 2023, certificates issued to two additional organizations were observed in the DangerousPassword attacks: BBQ BAZAAR PRIVATE LIMITED (team identifier `7L2UQTVP6F`) and DPS EXPRESS COMPANY LIMITED (team identifier `PN6L92RH7B`).

Subject name	Country	Email	Type of activity
16:20 Software, LLC	Pennsylvania, US	N/A	Operation In(ter)ception [11]
2 TOY GUYS	Florida, US	N/A	Operation DreamJob [20]
726 Lucile Development LLC	New Mexico, US	harryifrost@yahoo[.]com	Operation In(ter)ception [11]
BRAIN Technology INC	Oklahoma, US	lucasvcastillo.x@gmail[.]com	Operation In(ter)ception [11]
Alexis Security Group LLC	Arizona, US	RaymondJBurkett@protonmail[.]com	WIZVERA supply chain [28]
DREAM SECURITY USA INC	California, US	N/A	WIZVERA supply chain [28]
"A" MEDICAL OFFICE, PLLC,	New York, US	N/A	Operation DreamJob [26], [40]
DOCTER USA, INC.	Florida, US	bernardmking@tutanota.com	Exploited INITECH software [35]
MATCH CONSULTANTS LTD	Tackley, GB	N/A	From VirusTotal, unreported
SAMOYAJ	West Yorkshire, GB	N/A	Operation In(ter)ception [48]

Table 5: Code-signing certificates used by Lazarus mostly before 2022 (the latest certificate, SAMOYAJ, is from January 2022).

2022–2023

Since February 2022, the attackers have abandoned the previous pattern. The most notable change is the frequent use of *ProtonMail* as the email provider; see Table 6.

Subject name	Country	Email	Type of activity
Baltkot	Saint Petersburg, RU	baltkod@yandex.ru	Operation In(ter)ception
Dmitry Raykhman	New York, US	alexander132@protonmail.com	DangerousPassword attacks
Damion Spencer	Merseyside, UK	damions112@proton.me	DangerousPassword attacks
Scan-trader ApS	Midtjylland, DK	scan-trader@mail.ee	Operation In(ter)ception
Cold Air Systems	Ontario, CA	rezulbrown@protonmail.com	DangerousPassword attacks

Table 6: Recent certificates used by Lazarus.

Encryption methods

It's apparent that the attackers have strong technical skills at their disposal. In order to protect their operations – especially the payloads of later stages, the configuration files, and the network protocol – they incorporate strong ciphers in their code (using either a low-level implementation or a high-level one via *Windows Cryptographic Providers*). Table 7 shows a summary of encryption algorithms observed in payloads from their attacks in the time frame 2022-2023. From the length of keys, it's obvious that security is high and brute-force attacks are out of the question (we didn't analyse whether reductions in the encryption complexity are possible in these cases).

AES	NickelLoader (128-bit), WebbyTea (256-bit), ImprudentCook (128-bit), PostNapTea, wAgentTea (128-bit), SecondHandTea (256-bit); BlindingCan (256-bit + LZ4); miniBlindingCan (256-bit + LZ4); various droppers (128-bit); IconicLoader (Galois/Counter Mode, GCM); SimpleTea for Linux (GCM)
HC	OfficeCertTea (128-bit); dropper of ComeBacker [10] (256-bit)
A5/1	SimpleTea (96-bit)
(custom) RC4	In(ter)ception backdoor, BackbitingTea, BlindingCan, miniBlindingCan
RC5	Racket downloader (256-bit)
RC6	LightlessCan (256-bit)
ChaCha20	In(ter)ception backdoor, new ThreatNeedleTea
Salsa20	Dropper of ScoringMathTea (256-bit)
IDEA	ScoringMathTea
Panama	WinInetLoader (256-bit)
VEST-32	Payload from early Operation DreamJob [49]
(custom) DES-based	LPEClientTea [26]

Table 7. Encryption ciphers used in Lazarus payloads.

Rich Headers analysis

Just as before [9], we looked at the Rich Headers metadata [50] from the PE headers of payloads involved in the Lazarus attacks. Based on our data, the majority of attacks target 64-bit systems, as we see four times more 64-bit executables than 32-bit ones. Unlike the results from 2018, in the time since H2 2020 there were no builds by Visual Studio 98.

Table 8 presents an overview of which Visual Studio linker was used to produce various types of Lazarus malware. It seems that the majority of projects are developed in Visual Studio 2010. Interestingly, there's homogeneity in the origin of samples associated with Operation In(ter)ception – produced mostly with Visual Studio 2015 (and mostly with the linker version 14.0.24215).

Linker version	**Developed malware**
Visual Studio 2010 (10.0.30319)	NickelLoader, WebbyTea, OfficeCertTea, BackbitingTea, SecondHandTea, LightlessCan, old ThreatNeedleTea
Visual Studio 2010 SP1 (10.0.40219)	BackbitingTea, ImprudentCook, ScoringMathTea, HTTP(S) uploader, BlindingCan, miniBlindingCan, FudModule, new ThreatNeedleTea
Visual Studio 2015 (14.0.24215 + 14.0.24210)	Operation In(ter)ception (including droppers, loaders, injectors, downloaders, and backdoors)
Visual Studio 2017 (14.11.25547)	wAgentTea
Visual Studio 2019 (14.16.27031, 14.21.27702, 14.29.30146)	PostNapTea, WinInetLoader

Table 8: A summary of Visual Studio linkers used to produce Lazarus malware.

BACKDOORS

While the toolset used by Lazarus is large and has evolved over time, in this section we focus only on the most representative payloads that were acquired in recent years. All payloads from the initial stages, which serve mostly the purpose of establishing a stable foothold on the compromised system like document macros, scripts, native droppers, and loaders, are omitted. Under the term 'backdoors' we include all payloads that bring significant abilities to the attackers like downloaders with multi-step authentication and full-featured RATs. We try to refrain from calling these final, as there may be additional, but rarely seen, malware delivered based on the attackers' goals. Examples of payloads that could be considered final are the HTTP(S) uploader [40], FudModule [24], a keylogger, and a screenshot taker [51].

In the figures in this section, the procedure names are our interpretation of their function, not original symbolic information by the authors. Even if the executable is not a C++ project, we prefer to use the convention class::method, where the class is a general description (like `Core`, `System`, `HTTP`, `Command`, etc.) and the method is a specific description (like `parse_commands`, `get_processes`, `write_read_exchange`, `read_file`, etc.).

Downloaders

NickelLoader

We observed this payload for the first time in the Spanish case of Operation DreamJob [25]. NickelLoader is an HTTP(S) downloader that recognizes four commands, all five letters long (`abcde`, `avdrq`, `gabnc` and `dcrqv`). The length of these command names invoked a loose association with a slang term for the US five-cent coin – a nickel – so we named this backdoor NickelLoader. The most important commands are `avdrq` and `gabnc`, each of which loads a received buffer as a DLL. For this purpose, the attackers likely used MemoryModule [52], a library that can be used to load a DLL completely from memory.

Another occurrence of NickelLoader was recorded on 6 June 2022. A user from Italy submitted a disk image called `DisneyPDF.iso` to *VirusTotal*. The image contained two files: a trojanized open-source PDF viewer, `DisneyPDF.exe`, (the original, legitimate code is available online as MµPDF 1.19.0) and a fake job offer claiming to be from the *Disney Corporation* called `JD.pdf`. However, the document `JD.pdf` does not contain any job description, just the front page with a *Disney* theme and an additional two pages with the note 'No Preview Available'. Pages 2 and 3 appear only if opened with the `DisneyPDF.exe` reader (as it contains another PDF file with full content embedded and encrypted in its body); see Figure 3 above.

Another case was reported by *Microsoft* in September 2022 [22]. A trojanized *TightVNC Viewer* disguised as `IBMTech-VNC.exe` dropped a variant of NickelLoader with updated names for the commands (`eknag`, `eacec`, `hjmwk`, `wohnp`) and slightly different implementation. Instead of loading a DLL via `MemoryModule`, it simply executes the attacker's shellcode provided from the C&C server.

A variant of NickelLoader was mentioned as LIDSHOT in the blog post by *Mandiant* in March 2023 [53].

WebbyTea

This malware is a downloader associated with DangerousPassword attacks. We named this based on the string `web_t`, which is present in the PDB information of its dropper (`Y:\E\Code\Developing\C\`**`web_t`**`\PdfViewer\x64\Release\DevExpress.XtraList.v19.2.pdb`). A variant of WebbyTea was reported as Persistent Backdoor #1 in [51]. In July 2023 we also saw a variant of native WebbyTea for *macOS* systems.

Initially, it expects a base26-encoded URL (see Figure 5) stored in the registry key `Desktop` located at `HKEY_CURRENT_USER\Software\Microsoft\Accessibility` (this artifact is deleted immediately after the malware successfully reads it).

```
 1 __int64 WinMainEx()
 2 {
67   ii = 0;
68   jj = 0;
69   while ( 1 )
70   {
71     v43 = jj;
72     v44 = 0i64;
73     if ( jj >= strlen(Encoded) )
74       break;
75     Decoded[ii] = 26 * (Encoded[jj] - 'A') + Encoded[jj + 1] - 'A';
76     jj += 2;
77     ++ii;
78   }
79   Decoded[ii] = 0;
80   First4Bytes = Decoded;
81   if ( &v6 != (__int64 *)-112i64 && !strncmp_2(First4Bytes, "http", 4ui64) )
```

Figure 5: Base26-encoded URL decoded at WebbyTea's startup.

During initialization, the malware generates a unique identifier for the victim, based on the user and computer names, and the current time. Then it collects the following data (for illustration, we provide a PowerShell command in parentheses that outputs the same value):

- Proxy settings (`Get-ItemProperty -Path 'HKCU:\Software\Microsoft\Windows\CurrentVersion\Internet Settings' | Select ProxyEnable, ProxyServer, AutoConfigUrl`)
- Installation date (`Get-CimInstance -ClassName Win32_OperatingSystem | Select-Object -Property InstallDate`)
- *Windows* product name and version (`Get-ComputerInfo | Select WindowsProductName, OsHardwareAbstractionLayer`)

- Manufacturer (`Get-CimInstance -ClassName Win32_ComputerSystem | Select-Object -Property Manufacturer`)
- Product name (`Get-CimInstance -ClassName Win32_ComputerSystemProduct | Select-Object -Property Name`)
- System boot time (`systeminfo | find "System Boot Time"`)
- Time zone (`Get-TimeZone | Select-Object -Property "DisplayName"`)
- Computer name\username (`$env:computername \ $env:username`)
- Current time (`Get-Date`)
- List of running processes formatted, including PID, session ID, PPID, image name, and bitness (`Get-CimInstance Win32_Process | Select ProcessId, SessionId, ParentProcessId, Name`)

The system data is then encrypted using the AES-256 ECB cipher and a hard-coded key. The encrypted data is sent to a C&C server and the response is expected to be a DLL that is loaded and executed as a remote thread in a newly created `explorer.exe` process.

OfficeCertTea

We observed both 32-bit and 64-bit variants of OfficeCertTea in the wild, for example in Q2 2021 in Belgium [40]. This payload was also mentioned by *Kaspersky* in April 2023 [26], under the generic name 'Downloader Loader', so we decided to name it after one of its in-the-wild filenames, `officecert.ocx`.

During initialization, the malware generates a unique identifier for the victim, based on the user and computer name, and the current time. Then it collects the following data (again, a PowerShell command is provided for illustration) and sends it to the C&C server:

- Computer name\username (`$env:computername \ $env:username`)
- *Windows* product name, version, and architecture (`Get-ComputerInfo | Select WindowsProductName, WindowsCurrentVersion, OSArchitecture`)
- Info about the local computer according to the DNS (`$env:userdomain` and `Get-CimInstance -ClassName Win32_ComputerSystem | Select DNSHostName`)
- Logical drives (`Get-CimInstance -ClassName Win32_LogicalDisk | Select DeviceID`)
- List of running processes formatted, including PID, PPID, and image name (`Get-CimInstance Win32_Process | Select ProcessId, ParentProcessId, Name`)
- Architecture: adds the same value as the bitness of the malware and is hard-coded.

In Figure 6, the four-step communication of OfficeCertTea is visible in the form of pseudocode. An encrypted DLL to load is received in the final, fourth step.

```
1   void __fastcall Core::MainConnect(unsigned int *a1)
2   {
18
19    Sleep(60000u);
65    System::get_user_computer_name((__int64)SystemInfo);
66    System::get_product_version_arch((__int64)SystemInfo);
67    System::get_server_DNS((__int64)SystemInfo);
68    System::get_logical_drives((__int64)SystemInfo);
69    System::get_processes((__int64)SystemInfo);
108   if ( (unsigned int)HTTP::connect((__int64)SystemInfo) )
109     exit(0);
110   if ( (unsigned int)HTTP::read_page1(SystemInfo) )// GET ?forumID=165140040&method=page1
111     exit(0);
112   if ( (unsigned int)HTTP::write_enRSA_page2((__int64)SystemInfo) )// POST ?forumID=165140040&method=page2
113     exit(0);
114   if ( (unsigned int)HTTP::write_enHC128_page3((__int64)SystemInfo) )// POST ?forumID=165140040&method=page3
115     exit(0);
116   if ( (unsigned int)HTTP::read_deHC128_page4(SystemInfo) )// GET ?forumID=165140040&method=page4
117     exit(0);
118   Sleep(0xFFFFFFFF);
119 }
```

Figure 6: The main communication steps of OfficeCertTea.

ImprudentCook

This payload was reported for the first time by *AhnLab* in June 2021 [39]. Most notable is that it's hidden, together with its configuration and an AES-128 key for its decryption, in an ADS stream of its dropper. We saw two in-the-wild cases, both of the Operation DreamJob type of activity. We decided to name it ImprudentCook because the module contains two arrays of strings that represent cookie names for services, including *Bing*, *Daum* and *GitHub*:

1. `iKc;__uid;OAX;DMP_UID;PCID;_gid;_gat;csrftoken;NID;1P_JAR;JSESSIONID;WLS;SNID;__utma;BID;SRCHD;GsCK_AC;spintop;eader;XSRF-TOKEN;_gat_gtag_UA;webid_enabled;EDGE_V;dtck_channel;dtmulti;UUID;XUID;ZIA;IUID;SSID;_gh_sess;_octo`
2. `channel;post_titles;xfw_exp;wiht_clkey;SGPCOUPLE;NRTK;fbp;uaid;SRCHUSR;GUC;HPVN;dtck_blog;dtck_media;MUIDB;SRCHHPGUSR;SiteMain`

This is a quite distinguishing feature from other Lazarus payloads; however, the array of cookies was not leveraged later in the code. It contains information that looks like the version and the value was 5.60 in both cases (the version of the payload from *AhnLab*'s report was 5.40).

For proper functioning, the malware requires a configuration stored in the parent process's executable, in an ADS stream called `:rsrc`. It contains multiple URLs and a UID of the client. The main functionality of ImprudentCook is to connect to a responding C&C server, authenticate with the server in three major steps, and receive and load additional stages in the form of DLLs. The exchanged content is Base64 encoded and AES-128 encrypted.

WinInetLoader

Lazarus trojanized various *Notepad++* projects (which we described in more detail in the 'Plugins for Notepad++' section) to drop and execute this malicious WinInetLoader HTTP(S) downloader. We decided on this nickname because the malicious part repeatedly resolves *Windows* APIs from `wininet.dll` using the library name 'WinInet' in camel case.

For an HTTP query to be accepted by the server, malware authors usually choose constant parameter names. To increase variety (likely for network detection evasion), the attackers decided to loosen the format of the query and allow multiple parameter names in their HTTP queries. The names are sorted in three arrays and formatted into the `%s=` part of the query; see Figure 6. The arrays are as follows:

1. `idxAp, idv, lid, ds_s_kwgid, gclid, cx, eqid, cvid, sv_pq, ylt`
2. `q, board, orderBy, rangeType, iscqry, pq, wd, kid, bdx, licu`
3. `gs_, ved, rsv_t, iylc, refig, w, ei, ds_e_adid, sig, list`

Each parameter is chosen randomly. Moreover, there's a hard-coded value of the second parameter within the client: see Figure 7.

```
1  bool __stdcall HTTP::write_read_exchange()
2  {
27   TickCount = GetTickCount();
28   srand(TickCount);
29   l_ParameterNames_2 = &ParameterNames_2[rand() % 10];
30   l_ParameterNames_1 = &ParameterNames_1[rand() % 10];
31   v3 = rand();
32   swprintf_s(
33     (wchar_t *const)Buffer,
34     0x824ui64,
35     (const wchar_t *const)"%s=%s&%s=QQ34ES899WEET17BGSAAA8&%s=",
36     ParameterNames_0[v3 % 10].ParameterName,
37     g_5Letter_ID,
38     l_ParameterNames_1->ParameterName,
39     l_ParameterNames_2->ParameterName);
```

```
<'gs_'>
<'ved'>
<'rsv_t'>
<'iylc'>
<'refig'>
<'w'>
<'ei'>
<'ds_e_adid'>
<'sig'>
<'list'>
```

Figure 7: HTTP(S) query of WinInetLoader with randomly chosen parameters from three groups of names.

The network communication consists of two HTTP exchanges. The first is Base64 encoded and serves as the authentication step. The second is both Base64 encoded and encrypted by Panama [54], which is a quite uncommon 256-bit stream cipher, and the result is a buffer with a command for what to do next. Based on the value of the command, one of the following is performed:

- `0x11173`: a third HTTP exchange is realized (Base64 encoded and Panama encrypted).
- `0x11174`: shellcode is included in the buffer and the malware executes it.
- `0x11176`: the client disconnects.

wAgentTea

In December 2020, *Kaspersky* [55] reported on Lazarus attacking entities related to COVID-19 research. A payload called wAgent was dropped and loaded by the trojanized legitimate compression utility *XZ Utils*. We recorded deployment of very similar wAgent variants against mostly South Korean targets. As the name of the malware collides (at least phonemically) with various legitimate software projects or their components, we added the usual suffix Tea for its designation.

As in the *Kaspersky* blog post, there is a list of hard-coded parameter names used in an HTTP request:

```
identy;tname;blogdata;content;thesis;method;bbs;level;maincode;tab;idx;tb;isbn;entry;doc;
category;articles;portal;notice;product;themes;manual;parent;slide;vacon;tag;tistory;
property;course;plugin
```

The network communication between the client and the server involves several exchanges of HTTP request/response pairs, always protected by the combination of Base64 encoding and AES-128 encryption. The final step is receiving a DLL that is loaded and executed using the `MemoryModule` method, just as in the case of NickelLoader. This classifies wAgentTea as a sophisticated downloader.

miniBlindingCan

In [25], we spotted malware sharing the starting command ID `0x2009` with BlindingCan, but with only a small subset of the remaining command IDs, so it does not support BlindingCan's full set of features (see Figure 8). Instead, this malware has the ability to download new shellcode and exchange its configuration with the C&C server. Because of this significant reduction, we decided just to add the prefix 'mini-' to the name. Note that both pieces of malware mostly likely share the same implementation of the C&C server part.

```
1  __int64 __fastcall Core::parse_commands(LPVOID lpThreadParameter)
2  {
24   while ( 1 )
25   {
26     if ( CommandId[1] == 0x2009 )
27     {
28       SystemInfo = g_Argc;
29       memmove(a4, &SystemInfo, TargetSystemInfo_Size);
30       if ( !(unsigned int)HTTP::report_home(a1, (char *)g_u64Unknown, 0x1990,
31         goto _exit_code_5;
32       goto LABEL_31;
33     }
61     switch ( CommandId[1] )
62     {
63       case 0x2031:
64         l_Configuration = LocalAlloc(LMEM_ZEROINIT, MalwareConfig_Size);
65         memmove(l_Configuration, &g_Configuration, MalwareConfig_Size);
79       case 0x2032:
80         memmove(&g_Configuration, CommandId + 8, MalwareConfig_Size);
81         Command::encrypt_write_file();
82         goto LABEL_22;
83       case 0x2037:
84 LABEL_22:
85         if ( !(unsigned int)HTTP::report_home_wrp((char *)g_u64Unknown, 0x1990,
86           goto _exit_code_5;
```

Figure 8: Command parsing by miniBlindingCan with command IDs highlighted. Only the value 0x2009 is shared with BlindingCan.

Full-featured RATs

BadCall/SimpleTea

These payloads are compiled from a common code base for all major platforms: *Windows*, *Linux* and *macOS*. As the samples were collected over various attacks in time and have overlapping functionality, the naming ambiguities were introduced.

In April 2023 [32], in relation to the *3CX* incident, we reported on two *Linux* payloads, named BadCall for Linux and SimplexTea (which we alternatively call SimpleTea for Linux). The first is supposed to run on *Linux* servers and decrypts its configuration from `/tmp/vgauthsvclog` using `0x5E` as the XOR key. The second is more complex. It's an object-oriented project, which does not run on *Linux* distributions without a graphical user interface, and decrypts its

configuration from `/home/%user%/.config/apdl.cf` using `0x7E` as the XOR key. From the class names implementing the supported commands, it looks like the latter project was based on the first one, and additionally expanded its features. We concluded that both of these *Linux* payloads come from the same attackers.

Still, we were looking for a connection to Lazarus. There is a report by CISA from September 2019 [57] that describes proxy server RATs for *Windows* that use the same fake TLS communication trick as the *Linux* payload for servers. That was the reason we decided to denote the *Linux* malware as BadCall. However, the code of these *Windows* tools does not contain the command parsing logic that is present in SimpleTea; see Table 9. Because of the fake TLS trick with the same list of masquerading domains (see [32], Figure 4), and the use of the A5 stream cipher (see [32], Figure 5), we conclude that this evidence is sufficient for the attribution verdict.

There is a *macOS* build of SimpleTea as well. In February 2021 [58], CISA reported on *macOS* samples involved in the cryptocurrency theft-motivated attacks from 2020. A payload named `prtspool` is included in the list. It loads its configuration from an XOR-encrypted configuration stored in `/private/etc/krb5d.conf` (a single-byte key, `0x5E`, is used for its decryption). Most importantly, it has an analogical implementation of command parsing; see Table 9.

Thanks to the report by *Kaspersky* [59] on the *3CX* incident and their description of a Gopuram loader, we finally discovered also the *Windows* counterpart of the SimpleTea malware in *ESET*'s telemetry. A payload displays the required similarities like the loading of the configuration file (Figure 9), the A5 cipher, and the supported commands (Table 9). The attackers managed to deploy this malware as a memory-only payload against an individual from Germany in January 2022. At the same time, a variant of Gopuram loader was dropped on the victim's file system as `C:\Windows\System32\wbem\ncobjapi.dll`. This shows a relationship between SimpleTea for *Windows* and the Gopuram loader, and also adds up to the attribution of all three cases: the *HSBC*-themed malware targeting *Linux* users; the *3CX* incident, and the individual from Germany.

```
__int64 LoadConfig()
{
  memset(FileName, 0, 0x105ui64);
  memset(SystemDirectory, 0, 0x105ui64);
  NumberOfBytesRead[0] = 0;
  fn_GetSystemDirectoryA(SystemDirectory, 260i64);
  FormatString(FileName, "%s\\%s", SystemDirectory, "mfds.ax");
  hConfigFile = (void *)fn_CreateFileA(FileName, GENERIC_READ, 1i64);
  len = 14i64;
  do
  {
    *i++ ^= 0x5Eu;
    --len;
  }
  while ( len );
  fn_CloseHandle(hFile);
  return 1i64;
}
```

Figure 9: A routine in SimpleTea for Windows showing how the configuration file mfds.ax is decrypted using the 0x5E XOR key.

Command	Windows	Linux (Server/Desktop)	macOS
MSG_ReadConfig	0x3521	0x5252/0x27CA	0x3521
MSG_WriteConfig	0x3522	0x5253/0x27CB	0x3522
MSG_Del/MSG_SecureDel	0x3523	0x5244/0x27CC	0x3523
MSG_Up	0x3524	0x523F/0x27CD	0x3524
MSG_Down	0x3525	0x5240/0x27CE	0x3525
MSG_Cmd	0x3529	0x5246/0x27D2	0x3529
MSG_Run	0x352A	0x5243/0x27D3	0x352A
MSG_Dir	0x352C	0x523E/0x27D5	0x352C
MSG_Test	0x352F	0x524B/0x27D8	0x352F
MSG_SetPath	0x3530	0x524A/0x27D9	0x3530
MSG_GetComInfo	N/A	0x5249/0x27D8	N/A
MSG_Sleep	N/A	0x5251/0x27C4	N/A
CMsgZip	0x3527	N/A /0x27D0	N/A

Table 9: Indices of SimpleTea commands for Windows, Linux and macOS.

BlindingCan

BlindingCan is a full-featured RAT whose name was coined by CISA in August 2020 [60]. It is one of the most notorious payloads associated with Lazarus and has been observed in many of their attacks and reported multiple times, e.g. [23] (called AIRDRY), [25], [40], [61].

LightlessCan

In [25], we discovered a new RAT that we decided to call LightlessCan (the name is inspired by BlindingCan). We consider it a major shift in comparison with its predecessor. The most notable is the malware developers' focus on mimicking the functionality of the usual red-teaming *Windows* commands, like `wmic`, `whoami`, `systeminfo`, `netstat`, `netsh advfirewall`, `tasklist`, `ipconfig`, `net`, `schstasks`, `reg`, `sc`, `ping`/`ping6`, etc. By executing the code within their malware, instead of on the command line, they significantly reduce the noisiness that behaviour-based security solutions like EDRs monitor. It seems that Lazarus takes detection evasion as a serious priority (note also the sophisticated EDR-blinding malware FudModule at their disposal). We didn't conclude whether they more likely reversed *Windows* system binaries or got inspired by the code available via the Wine project (or both). In the previously reported Lazarus attacks [19], these commands were executed multiple times after the attackers had gained a foothold in the target's system.

A variant of LightlessCan seems to be have been reported under the name SIDESHOW by *Mandiant* in March 2023 [53].

PostNapTea

This payload is a complex object-oriented project that has not yet been described in public. It stores its configuration in JSON format (an example of such a configuration is shown in Listing 1 below). The version `0.0.1` suggests that this is a new development.

It resolves the *Windows* APIs it requires during runtime, via the Fowler–Noll–Vo (FNV) hash function (note that ScoringMathTea does that similarly, but the algorithm is a modified Justin Sobel hash). There are five classes that represent command groups:

- `CCButton`: for file manipulation and screen capturing
- `CCBitmap`: for network commands
- `CCComboBox`: for file system management
- `CCList`: for process management
- `CCBrush`: for control of the malware itself

```
{
    "proxylist": [{
        "proxy": "https0x3A//www.takegawahelmet.com.tw",
        "url": "/wp-includes/SimplePie/Net/IPv4.php"
    }],
    "service": "",
    "process": "",
    "count": 10,
    "default_sleep": 10,
    "type": 3,
    "ckey": "WfClq6HxbSaOuJGaH5kWXr7dQgjYNSNg",
    "id": "Z5UAHijc",
    "ver": "0.0.1",
    "monitor_process": "false",
    "monitor_usb": "true",
    "monitor_session": "true",
    "monitor_hiber": "false",
    "dir": "C0x3A\Windows",
    "cmd": "C0x3A\Windows\system32\cmd.exe",
    "hibernate": 0
}
```

Listing 1: PostNapTea's configuration in JSON.

First, the group `CCButton` contains four commands for file manipulation and screen capturing. Next, the `CCBitmap` class implements functionality that mimics *Windows* commands often used by attackers after compromising a system. The supported commands are `sc`, `reg`, `arp`, `net`, `ver`, `wmic`, `ping`, `whoami`, `netstat`, `tracert`, `lookup`, `ipconfig`, `systeminfo`, and `netsh advfirewall`. Similar to the LightlessCan case, creating their own implementations of these

commands allows them to operate more stealthily and lowers the chance of being detected by behavioural security products like EDRs.

Next, the class `CCComboBox` contains commands related to file system manipulation: deleting a file securely, getting a directory's content, getting a list of drives, creating a new folder, setting the file times, getting properties of a folder, and renaming a file.

In the class `CCList`, various commands for process management are implemented: unloading a DLL, creating a new process, creating a new process as a user, injecting a DLL in a process, killing a process, and getting a process list.

Finally, the class `CCBrush` implements features for controlling the malware itself: getting information about the victim, updating the JSON configuration, sending the configuration to the C&C server, and testing a C&C connection.

BackbitingTea and SecondHandTea

These two payloads are full-featured RATs and seem like flagship backdoors used in the DangerousPassword campaigns. Both malware projects appear to be based on the same code base. They differ in several properties like the paths of the configuration files (see the bullet list below), the network library (openSSL-1.1.0f vs. wolfSSL vs. Winsock TCP/IP), the encryption (RC4 vs. AES-256), and the compression (LZ4 vs. ZIP). Overall, however, from Table 10 it's obvious that most of the functionality between variants of BackbitingTea (observed in Canada in March 2022, and in the United States in December 2022) and SecondHandTea is shared.

Functionality	**SecondHandTea**	**BackbitingTea Canada**	**BackbitingTea USA**
Get system info	`0x00`	`0x6F00620069006E`	`0xE2FCD92E`
Get disk info	`0x01`	`0x6F00620075006E`	`0xF5A367E2`
Get directory list (`dir`)	`0x02`	`0x75007300740065`	`0x24E430BD`
Get directory tree	`0x03`	`0x70007200640064`	`0xBA8F85C7`
Compress and upload a file	`0x04` (LZ4)	`0x64006500730063`	`0x987ADC57`
Download and decompress a file	`0x05` (LZ4)	`0x73007500620069`	`0xBF2979EEi64`
Compress a folder recursively (prefix of the temporary output and the algorithm in brackets)	`0x06` (`TPK`, LZ4)	`0x6400650073007A` (`TMP`, ZIP)	`0xC70C64BB` (`TMP`, ZIP)
Delete file securely	`0x07`	`0x6C0069006D0070`	`0xC0DC3219`
Get process info (`tasklist`)	`0x08`	`0x76006900700072`	`0x80A00A8F`
Terminate a process by PID	`0x09`	`0x7000720064006D`	`0xFCD96BAE`
Create process and collect output	`0x0A`	`0x6C00ED00640063`	`0xFAA62957`
Create/open an event	`0x0E`	`0x63006F006D0065`	`0x16104DA7`
Set file times as the source (in brackets)	`0xF` (`kernel32.dll`)	`0x74006900640061` (`wininet.dll`)	`0xFDF50A87` (`msconfig.exe`)
Set current directory	`0x10`	`0x63006100640069`	`0xD443E3DF`
Update the encrypted configuration on the file system (byte size and cipher in brackets)	`0x11` `0x12` (6398, AES-256)	`0x6500730063006F` `0x64006F0072006D` (6816, RC4)	`0x297B5E45` `0x9E0A387B` (7824, RC4)
Create a new process	`0x13`	`0x63006F00720072`	`0xE461424B`
Create a new process as a user	`0x14`	`0x63006F00630073`	`0x7DE16DD7`
Execute an MZ file in a console or as a remote thread	`0x15`	`0x69006E00640070`	`0xDD221747`
Inject and execute an MZ file into a process (by PID) as a new remote thread	`0x16`	`0x69006E00790069`	N/A
Decrypt, inject, and execute shellcode into a `explorer.exe` process as a new remote thread	`0x17`	`0x69006E00790065`	`0xFCD8FF01`
Open a TCP connection	N/A	`0x70007200640063`	`0xB10B015E`
Uninstall self	N/A	`0x6D006F00720069`	N/A
Download a file and write it to the file system	N/A	`0x73007500620069`	`0xBF2979EE`

Table 10: A comparison of commands among variants of BackbitingTea and SecondHandTea.

The names of configuration files are hard-coded in the binaries. In the parentheses are either an in-the-wild case from our telemetry or the name under which it was publicly reported:

- `C:\Windows\AppPatch\msomain.sdb` (msoRAT [62])
- `C:\Users\Public\Videos\vid.list` (a development server in Cyprus targeted in July 2021)
- `C:\Users\Public\Videos\OfficeIntegrator.dat` (Backdoor [51])
- `C:\Windows\AppPatch\PublisherPolicy.tms` (Persistence Backdoor #2 [51])
- `C:\Windows\System32\normnlc.nls` (DangerousPassword in Canada in March 2022)
- `C:\Users\Public\Videos\fav.dat` (DangerousPassword in the USA in December 2022)
- `C:\Windows\assembly\pubvak16.dat` (SecondHandTea, DangerousPassword in the Netherlands in February 2023)

ThreatNeedleTea

The history of this payload starts as early as the beginning of 2021, when it was observed by *Google* [10] in attacks against security researchers, and against the defence sector [63] by *Kaspersky*, whose researchers came up with the name of this activity cluster – ThreatNeedle). To distinguish between the activity and the backdoor (a full-featured RAT), we added the preferred suffix -Tea.

We observed an updated version of ThreatNeedleTea deployed against South Korean targets in Q1 2022. It replaced RC4 for encryption of network data with either a single-byte XOR key or the ChaCha20 cipher. The RAT supports the usual set of commands that are sorted in a function table; see Figure 10.

```
__int64 Core::process_commands()
{
  if ( Core::decrypt_config() )
  {
    strcpy(g_Key128, "g9nr7xi5gh1P0DcY");
    Commands[1] = (__int64)Command::read_file;
    Commands[2] = (__int64)Command::download_file;
    Commands[3] = (__int64)Command::send_output_enChaCha20;
    Commands[4] = (__int64)Command::rename_file;
    Commands[5] = (__int64)Command::get_filetime;
    Commands[6] = (__int64)Command::dir;
    Commands[7] = (__int64)Command::free_disk_space;
    Commands[8] = (__int64)Command::execute_with_attributes;
    Commands[9] = (__int64)Command::get_token_from_SID;
    Commands[10] = (__int64)Command::list_processses_with_SIDs;
    Commands[11] = (__int64)Command::terminate_process_by_PID;
    Commands[12] = (__int64)Command::stub_0;
    Commands[13] = (__int64)Command::get_logical_drives;
    Commands[14] = (__int64)Command::disconnect;
    Commands[15] = (__int64)Command::download_config;
    Commands[16] = (__int64)Command::upload_configuration;
    Commands[17] = (__int64)Command::OK;
    Commands[18] = (__int64)Command::TCP_connection;
    Commands[19] = (__int64)Command::execute_collect_output;
    Commands[20] = (__int64)Command::set_current_dir;
    Commands[21] = (__int64)Command::get_current_dir;
    Commands[22] = (__int64)Command::get_system_info;
    Commands[23] = (__int64)Command::memload_DLL;
    Commands[24] = (__int64)Command::send_output_enXOR;
    Commands[25] = (__int64)Command::send_temp_file_enXOR;
```

```
_BOOL8 __fastcall Command::execute_collect_output(__int64 a1)
{
  printf_l_0(CommandLine, (const _locale_t)0, "/c %s >> %s 2>&1",
  StartupInfo.cb = 104;
  StartupInfo.wShowWindow = 0;
  StartupInfo.dwFlags = 1;
  if ( CreateProcessW(
         aCWindowsSystem,
         (LPWSTR)CommandLine,
         0i64,
         0i64,
         0,
         CREATE_NO_WINDOW,
         0i64,
         0i64,
         &StartupInfo,
         &ProcessInformation) )
```

```
__int64 Command::OK()
{
  return 1i64;
}
```

Figure 10: Commands supported by the new ThreatNeedleTea. Execution where the output is sent back to the C&C server is a typical supported command in many Lazarus payloads.

ThreatNeedleTea fingerprints the system for information like hostname, architecture, computer name, *Windows* product name, and physical address (the last can be printed by running `Get-NetAdapter -Name * -Physical | Select MacAddress` in PowerShell). The following is an example of data sent to the C&C server:

```
|x64|Wininet : 1_exe : Core|MSEDGEWIN10|Windows 10 Enterprise|2|00-0C-29-AC-74-05
```

Interestingly, an older version of ThreatNeedleTea contains a test command indexed by the number 13. It just sends a formatted message, 'Recv Msg : OK', back to the server (shown in Figure 11). It may be the same command with void functionality at index 17 in the new ThreatNeedleTea: see Figure 10. This particular string is similar to a function name used in SimpleTea for *macOS* and *Linux*; see [32], Table 1 (the `RecvMsg` command of BADCALL for *Linux*).

```
1  int __cdecl Core::parse_commands(LPCWSTR lpString)
2  {
288              if ( !lstrcmpW((LPCWSTR)&CommandID, L"0013") )
289              {
290                qmemcpy(&v79[1], L"\tRecv Msg : OK\r\n", 0x22u);
291                v21 = lstrlenW(&v79[1]);
292                results = operator new(2 * v21);
293                v22 = lstrlenW(&v79[1]);
294                memset(results, 0, 2 * v22);
295                v23 = lstrlenW(&v79[1]);
296                memcpy(results, &v79[1], 2 * v23);
297                v6 = 2 * lstrlenW(&v79[1]);
298                goto report_command_base64_RC4;
299              }
300              if ( lstrcmpW((LPCWSTR)&CommandID, L"0014") )
301              {
302                if ( lstrcmpW((LPCWSTR)&CommandID, L"0015") )
                   {
```

Figure 11: Command 0013 in the old ThreatNeedleTea with a familiar string observed in SimpleTea.

ScoringMathTea

ScoringMathTea is a complex RAT that supports 40 commands sorted in a function table, in the same style as ThreatNeedleTea. The implemented functionality is the usual required by Lazarus: manipulation of files and processes, exchanging the configuration, collecting the victim's system info, opening a TCP connection, and executing local commands or new payloads downloaded from the C&C server. Some of the hard-coded character strings of ScoringMathTea are shown in Figure 12. A format of a DLL export that is executed by one of the commands is highlighted.

It does hash-based dynamic *Windows* API resolution of the required functions, instead of the usual resolution of all the functions at the beginning of runtime. The algorithm used for the hash calculation is a modified Justin Sobel bitwise checksum (instead of `1315423911`, the constant `47954261` is used). For example, `0x61EC1D82` resolves to `kernel32!CreateFileW` and `0x58443E2C` to `ws2_32!socket`.

```
aFun02d         db 'fun%02d',0          ; DATA XREF: Command__exec_Export+82↑o
; const char a9p8mbeqs3ysnwd[]
a9p8mbeqs3ysnwd db '9P8mBEqs\3YsnwdpUz\UYz8E6P CM\1sFEoBDTW3Ko1E',0
                                        ; DATA XREF: System__ProductName+3D↑o
                align 8
; const char aTx4rphzw81u[]
aTx4rphzw81u    db 'tx4rPHzw81u',0      ; DATA XREF: System__ProductName:loc_1800158ED↑o

aSuccess:                               ; DATA XREF: FS__checkfiles+2E5↑o
                text "UTF-16LE", ' [Success]',0Dh,0Ah,0
                align 8
aFailed:                                ; DATA XREF: FS__checkfiles:loc_18001B7CB↑o
                text "UTF-16LE", ' [Failed]',0Dh,0Ah,0

aMozilla50Windo db 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHT'
                                        ; DATA XREF: HTTP__connect+C8↑o
                db 'ML, like Gecko) Chrome/107.0.0.0 Safari/537.36 Edg/107.0.1418.42',0
                align 8
aPost:                                  ; DATA XREF: HTTP__connect+356↑o
                text "UTF-16LE", 'POST',0
                align 8
aContentLength:                         ; DATA XREF: HTTP__write+37↑o
                text "UTF-16LE", 'Content-Length:',0
aSD_0:                                  ; DATA XREF: HTTP__write+2B↑o
                text "UTF-16LE", '%s %d',0
                align 20h
aContentTypeApp:                        ; DATA XREF: HTTP__write+6C↑o
                text "UTF-16LE", 'Content-Type: application/x-www-form-urlencoded',0
aDoctypeHtml    db '<!DOCTYPE html>',0  ; DATA XREF: HTTP__send_data_wrp+53D↑o
```

Figure 12: Some of the character strings of ScoringMathTea. The highlighted one is the format of an exported function of an additional stage.

CONCLUSION

The goal of this study is to provide a high-level perspective on Lazarus, mainly as seen by *ESET* telemetry during the last two years. The group is suitably considered as one of today's most active threat actors, spreading its attacks globally. From

the variety of the TTPs, campaigns and tools, it seems there must be a large number of cooperating people whose work manifests as the group's activity, including the development and testing of an enormous toolset of droppers, loaders, and client-server pairs; hacking weak webservers for C&C infrastructure; vulnerability research; completing complex tasks like acquiring valid code-signing certificates and robust evasion of security products; and related stealthy work, like monetizing illegal access and laundering the obtained funds and cryptocurrencies. Almost certainly there is additional hidden work that hasn't yet manifested, as inevitably only a limited part of 'the iceberg' has been revealed, and is revealable, by available means. Lazarus is undoubtedly very well organized and continues to be a significant threat in cyberspace.

ACKNOWLEDGEMENTS

We thank Dominik Breitenbacher for participating in this research. We appreciate Martin Smolár's efforts for carefully reading the text in the role of a reviewer. Also, thanks go to Slavomír Labský, Martin Rusňák and Jakub Štellár for helping with the payload analyses.

IOCs

The list of IoCs together with their descriptions can be found at [64]. Representatives of the mentioned malware families, together with their brief descriptions and references in public reports, can be found at [65].

REFERENCES

[1] Department of Justice. Three North Korean Military Hackers Indicted in Wide-Ranging Scheme to Commit Cyberattacks and Financial Crimes Across the Globe. 2021. https://www.justice.gov/opa/pr/three-north-korean-military-hackers-indicted-wide-ranging-scheme-commit-cyberattacks-and.

[2] Ji-Young, K.; In, L. J.; Gon, K. K. The All-Purpose Sword: North Korea's Cyber Operations and Strategies. 2019. Tallinn , IEEE, pp. 143-162.

[3] Kan, P. R., Jr. B. B.; Collins, R. M. Criminal Sovereignty: Understanding North Korea's Illicit International Activities. 2010. s.l.: US Army War College Press.

[4] Greitens, S. C.. Illicit: North Korea's Evolving Operations to Earn Hard Currency, Committee for Human Rights in North Korea. 2014. Washington, DC: s.n.

[5] Breuker, R.; Gardingen, I. v. eds. People for Profit: North Korean Forced Labour on a Global Scale. 2018. s.l.:Leiden Asia Centre.

[6] Berger, A. Target Markets: North Korea's Military Customers. 2016 s.l.:Routledge.

[7] Defense Intelligence Agency. North Korea military power: a growing regional and global threat. 2021. Washington, D.C.: U.S. Government Publishing Office.

[8] Steffens, T. Attribution of Advanced Persistent Threats: How to Identify the Actors Behind Cyber-Espionage. 2020. Heidelberg: Springer.

[9] Kálnai, P.; Poslušný, M., 2018. Lazarus Group: a mahjong game played with different sets of tiles. 2018. https://www.virusbulletin.com/conference/vb2018/abstracts/lazarus-group-one-mahjong-game-played-different-sets-tiles.

[10] Weidemann, A. New campaign targeting security researchers. Google. 25 January 2021. https://blog.google/threat-analysis-group/new-campaign-targeting-security-researchers.

[11] Breitenbacher, D.; Osis, K. Operation In(ter)ception: Aerospace and military companies in the crosshairs of cyberspies. WeLiveSecurity. 2020. https://www.welivesecurity.com/wp-content/uploads/2020/06/ESET_Operation_Interception.pdf.

[12] Jakobsson, M. Case Study: Business Email Compromise. In: Understanding Social Engineering Based Scams. 2016. New York, NY: Springer, p. 115–122.

[13] ThreatBook. The Nightmare of Global Cryptocurrency Companies. 2019. https://threatbook.cn/ppt/The%20Nightmare%20of%20Global%20Cryptocurrency%20Companies%20-%20Demystifying%20the%20%E2%80%9CDangerousPassword%E2%80%9D%20of%20the%20APT%20Organization.pdf.

[14] Tani, T. Spear Phishing against Cryptocurrency Businesses. JPCERT/CC. 9 July 2019. https://blogs.jpcert.or.jp/en/2019/07/spear-phishing-against-cryptocurrency-businesses.html.

[15] Park, S. BlueNoroff introduces new methods bypassing MoTW. SecureList. 27 December 2022. https://securelist.com/bluenoroff-methods-bypass-motw/108383/.

[16] Jamf Threat Labs. BlueNoroff APT group targets macOS with 'RustBucket' Malware. 2023. https://www.jamf.com/blog/bluenoroff-apt-targets-macos-rustbucket-malware/.

[17] ClearSky Research Team. Operation 'Dream Job' Widespread North Korean Espionage Campaign. August 2020. https://www.clearskysec.com/wp-content/uploads/2020/08/Dream-Job-Campaign.pdf.

[18] Brumaghin, E.; Rascagneres, P. Fake Cisco Job Posting Targets Korean Candidates. Cisco Talos. 30 January 2019. https://blog.talosintelligence.com/fake-korean-job-posting/.

[19] HvS-Consulting AG. Greetings from Lazarus: Anatomy of a cyber-espionage campaign. 15 December 2020. https://www.hvs-consulting.de/media/downloads/ThreatReport-Lazarus.pdf.

[20] ASEC Analysis Team. Q1 2021. ASEC Report Vol.102. AhnLab. https://cn.ahnlab.com/global/upload/download/asecreport/ASEC REPORT_vol.102_ENG (4).pdf.

[21] Park, S. Multi-universe of adversary: multiple campaigns of the Lazarus group and their connections. Virus Bulletin. October 2021. https://vblocalhost.com/conference/presentations/multi-universe-of-adversary-multiple-campaigns-of-the-lazarus-group-and-their-connections/.

[22] Microsoft Security Threat Intelligence. ZINC weaponizing open-source software. 29 September 2022. https://www.microsoft.com/en-us/security/blog/2022/09/29/zinc-weaponizing-open-source-software/.

[23] Maclachlan, J. et al., 2022. It's Time to PuTTY! DPRK Job Opportunity Phishing via WhatsApp. Mandiant. 14 September 2022. https://www.mandiant.com/resources/blog/dprk-whatsapp-phishing.

[24] Kálnai, P.; Havránek, M. Lazarus & BYOVD: evil to the Windows core. Prague, Virus Bulletin. October 2022. https://www.virusbulletin.com/uploads/pdf/conference/vb2022/papers/VB2022-Lazarus-and-BYOVD-evil-to-the-Windows-core.pdf.

[25] Kálnai, P. Lazarus luring employees with trojanized coding challenges: the case of a Spanish aerospace company. WeLiveSecurity. September 2023. https://www.welivesecurity.com/en/eset-research/lazarus-luring-employees-trojanized-coding-challenges-case-spanish-aerospace-company/.

[26] Park, S. Following the Lazarus group by tracking DeathNote campaign. SecureList. 12 April 2023. https://securelist.com/the-lazarus-group-deathnote-campaign/109490/.

[27] CISA. Defending Against Supply Chain Attacks. April 2021. https://www.cisa.gov/sites/default/files/publications/defending_against_software_supply_chain_attacks_508_1.pdf.

[28] Cherepanov, A.; Kálnai, P. Lazarus supply-chain attack in South Korea. WeLiveSecurity. 16 November 2020. https://www.welivesecurity.com/2020/11/16/lazarus-supply-chain-attack-south-korea/.

[29] Weidemann, A. Countering threats from North Korea. Google. 24 March 2022. https://blog.google/threat-analysis-group/countering-threats-north-korea/.

[30] Johnson, J. et al. 3CX Software Supply Chain Compromise Initiated by a Prior Software Supply Chain Compromise; Suspected North Korean Actor Responsible. Mandiant. 20 April 2023. https://www.mandiant.com/resources/blog/3cx-software-supply-chain-compromise.

[31] Threat Hunter Team. X_Trader Supply Chain Attack Affects Critical Infrastructure Organizations in U.S. and Europe. Symantec. 21 April 2023. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/xtrader-3cx-supply-chain.

[32] Kálnai, P.; M.Leveillé, M.-É. Linux malware strengthens links between Lazarus and the 3CX supplychain attack. WeLiveSecurity. 20 April 2023. https://www.welivesecurity.com/2023/04/20/linux-malware-strengthens-links-lazarus-3cx-supply-chain-attack/.

[33] Trading Technologies. The End of an Era: X_TRADER®'s Final Sunset. 6 September 2018. https://trading-tech.medium.com/the-end-of-an-era-x-trader-s-final-sunset-8208832ec058.

[34] ASEC Analysis Team. New Malware of Lazarus Threat Actor Group Exploiting INITECH Process. AhnLab. 26 April 2022. https://asec.ahnlab.com/en/33801/.

[35] Ryu, S. Analysis of Lazarus malware abusing Non-ActiveX Module in South Korea. 2021. https://medium.com/s2wblog/analysis-of-lazarus-malware-abusing-non-activex-module-in-south-korea-7d52b9539c12.

[36] Threat Hunter Team. Lazarus Targets Chemical Sector. Symantec. 14 April 2022. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/lazarus-dream-job-chemical.

[37] MITRE. CVE-2021-26606. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-26606.

[38] ASEC Analysis Team. A Case of Malware Infection by the Lazarus Attack Group Disabling Anti-Malware Programs With the BYOVD Technique. AhnLab. 31 October 2022. https://asec.ahnlab.com/en/40830/.

[39] ASEC Analysis Team. APT Attacks on Domestic Companies Using Library Files. AhnLab. 4 June 2021. https://asec.ahnlab.com/en/23717/.

[40] Kálnai, P. Amazon-themed campaigns of Lazarus in the Netherlands. WeLiveSecurity. 30 September 2022. https://www.welivesecurity.com/2022/09/30/amazon-themed-campaigns-lazarus-netherlands-belgium/.

[41] Firsh, A. Not a dream job: Hunting for malicious job offers from an APT. VirusTotal. 3 November 2022. https://blog.virustotal.com/2022/11/not-dream-job-hunting-for-malicious-job.html.

[42] SumatraPDF. https://github.com/sumatrapdfreader/sumatrapdf.

[43] MuPDF library. https://git.ghostscript.com/?p=mupdf.git;a=summary.

[44] UXReader PDF Reader Library for Windows. https://github.com/vfr/UXReader-Windows.

[45] PDFium. https://pdfium.googlesource.com/pdfium/.

[46] Apple. PDFKit. https://developer.apple.com/documentation/quartz/pdfkit.

[47] Imgui. https://github.com/ocornut/imgui.

[48] Saini, A.; Jazi, H. North Korea's Lazarus APT leverages Windows Update client, GitHub in latest campaign. Malwarebytes. 27 January 2022. https://www.malwarebytes.com/blog/threat-intelligence/2022/01/north-koreas-lazarus-apt-leverages-windows-update-client-github-in-latest-campaign.

[49] Tomonaga, S. Operation Dream Job by Lazarus. JPCERT/CC. 26 January 2021. https://blogs.jpcert.or.jp/en/2021/01/Lazarus_malware2.html.

[50] Poslušný, M.; Kálnai, P. Rich Headers: leveraging this mysterious artifact of the PE format. Virus Bulletin. October 2019. https://www.virusbulletin.com/virusbulletin/2020/01/vb2019-paper-rich-headers-leveraging-mysterious-artifact-pe-format/.

[51] Park, S.; Kamluk, V. The BlueNoroff cryptocurrency hunt is still on. SecureList. 13 January 2022. https://securelist.com/the-bluenoroff-cryptocurrency-hunt-is-still-on/105488/.

[52] MemoryModule. https://github.com/fancycode/MemoryModule.

[53] Mandiant Intelligence And Consulting. Stealing the LIGHTSHOW (Part One) — North Korea's UNC2970. 9 March 2023 https://www.mandiant.com/resources/blog/lightshow-north-korea-unc2970.

[54] Panama. https://github.com/winlibs/libmcrypt/blob/master/modules/algorithms/panama.c.

[55] Park, S. Lazarus covets COVID-19-related intelligence. SecureList. 23 December 2020. https://securelist.com/lazarus-covets-covid-19-related-intelligence/99906/.

[56] CISA. 2017. Malware Analysis Report (MAR) - 10135536-B. (Archived on 24 January 2022.) https://web.archive.org/web/20220124183620/https://www.cisa.gov/uscert/sites/default/files/publications/MAR-10135536-B_WHITE.PDF.

[57] CISA. MAR-10135536-10 – North Korean Trojan: BADCALL. 2019. https://www.cisa.gov/news-events/analysis-reports/ar19-252a.

[58] CISA. MAR-10322463-5.v1 – AppleJeus: CoinGoTrade. 2021. https://www.cisa.gov/news-events/analysis-reports/ar21-048e.

[59] Kucherin, G.; Berdnikov, V.; Kamalov, V. Not just an infostealer: Gopuram backdoor deployed through 3CX supply chain attack. SecureList. 3 April 2023. https://securelist.com/gopuram-backdoor-deployed-through-3cx-supply-chain-attack/109344/.

[60] CISA. MAR-10295134-1.v1 – North Korean Remote Access Trojan: BLINDINGCAN. 2020. https://www.cisa.gov/news-events/analysis-reports/ar20-232a.

[61] Tomonaga, S. BLINDINGCAN – Malware Used by Lazarus. JPCERT/CC. 29 September 2020. https://blogs.jpcert.or.jp/en/2020/09/BLINDINGCAN.html.

[62] Takai, H., Hayashi, S. & Koike, R. Unveiling The Cryptomimic. Virus Bulletin. September 2020. https://vblocalhost.com/uploads/VB2020-Takai-etal.pdf.

[63] Kopeytsev, V.; Park, S. Lazarus targets defense industry with ThreatNeedle. Kaspersky ISC CERT. 2021. https://ics-cert.kaspersky.com/media/Kaspersky-ICS-CERT-Lazarus-targets-defense-industry-with-Threatneedle-En.pdf.

[64] IOCs. https://github.com/eset/malware-ioc/tree/master/nukesped_lazarus.

[65] Malpedia. Lazarus Group. https://malpedia.caad.fkie.fraunhofer.de/actor/lazarus_group.