The Lazarus Constellation

A study on North Korean malware

19/02/2020
# TABLE OF CONTENTS

I. **Rise of Lazarus**
   - Introducing the Lazarus APT group ........................................5
   - Attribution: Links with North Korea ....................................5
   - Targets & Capabilities ..........................................................6
   - Clarifying links with other attacker groups .........................7

II. **Lazarus’ New Motives (2016 - 2019)**
   - Fighting sanctions in cyber space ......................................9
   - Banks & ATM .................................................................10
   - Targeting cryptocurrency business ....................................11
   - New toolset .....................................................................11

III. **Technical analysis of key Lazarus attacks**
    - Lazarus TTP .................................................................13
    - MITRE ATT&CK Matrix ..................................................19

IV. **Incident Response: how to uncover an ongoing Lazarus attack**
    - Context ..........................................................................21
    - First assessment .............................................................21
    - Attributing the attack .......................................................22
    - Uncovering attackers’ activities ......................................23
    - Payload analysis ............................................................25

V. **Classifying North Korean malware and interpreting links**
    - Dataset ...........................................................................29
    - Methodology .................................................................30
    - Visualization .................................................................32
    - Reviewing results ..........................................................32
    - Working hours and days of the Lazarus developers ..........38
    - Conclusion ......................................................................39

VI. **Detection & Mitigation**
    - Vulnerability used ..........................................................41
    - Detecting Lazarus activities ............................................42
    - Recommendations ..........................................................44

VII. **Appendices**
    - Appendix A: Abbreviations ............................................45
    - Appendix B: List of studied samples .................................45
    - Appendix C: SMB Bruteforce password list .....................49

References .................................................................................51

Contact .......................................................................................53
I. RISE OF LAZARUS

INTRODUCING THE LAZARUS APT GROUP

Lazarus activities have been retroactively tracked back to 2007, under various names. For years, these activities were seen as acts of cyberterrorism and vandalism, since most of them systematically involved destruction of data and / or distributed denial of service attacks.

The Lazarus group was clearly identified and named in the 2016 Novetta report “Operation Blockbuster”. This report uncovered and attributed a large set of malware based on the analysis of the Sony Pictures Entertainment targeted attack. Attribution and tracking was made possible due to the group’s habits of reusing huge chunks of code in most of their malware.

This report showcased how active and diverse the group is: using more than 45 different home-developed malware families, Lazarus has been conducting destructive attacks but also advanced and persistent spying campaigns all over the world, making it worthy of the “APT” designation. TTP, arsenal and targets reveal that Lazarus is composed of at least three different subgroups: the Lazarus “core”, aiming at disrupting activities and causing damage, Andariel, hacking for profit and intelligence, and Bluenoroff, motivated by financial gains.

Uncovering its malware and activities didn’t stop the Lazarus group from continuing its operations or renewing its arsenal, as the rest of this report will show.

The U.S. Government, mostly through its CERT, is referring to Lazarus as Hidden Cobra.

ATTRIBUTION: LINKS WITH NORTH KOREA

Lazarus activities have often been wrongly attributed to China or to unknown cyberterrorist groups. After identifying the Lazarus arsenal in 2016, researchers were able to track and attribute the group’s attacks, as well as monitoring their command & control servers. During an investigation, Group-IB discovered that Lazarus operators connected to a C&C using two IP addresses from North Korea (210.52.109.22 and 175.45.178.222). Moreover, analyses of compilation timestamps of the binaries used by the group in their attacks were consistent with North Korean working hours (see our analysis below). Other artefacts can be mentioned as well, such as the YMD date format found in Lazarus log files, which is used almost exclusively in the Korean region.
It is believed that Lazarus operators are linked to Bureau 121, a division of the Reconnaissance General Bureau intelligence agency (Group-IB). This attribution to North Korea was confirmed by FBI and NSA investigations, based on internal sources and the technical elements previously mentioned.

**TARGETS & CAPABILITIES**

Lazarus targets are very disparate, as the group has very diverse motives: intelligence, financial gains and disruption. Lazarus and its subgroups have been focusing on attacking governments, financial institutions, defense industry actors, IT and videogame companies. Geographically, most targets are located in South Korea and in South America.

Despite operator mistakes and the fact that their attacks are most of the time technically simple, Lazarus and its subgroups are well-funded and able to discretely maintain persistence in networks for years. They were seen adapting very fast, fighting against forensic investigators in real-time by repacking malware, erasing files or modifying encryption keys and algorithms in less than an hour after being discovered.

Furthermore, they have been leveraging many 0day vulnerabilities they bought or developed on their own throughout the years.

All of these operations come at a cost. The Bluenoroff subgroup is supposedly in charge of financing the whole ecosystem through big money heists.
CLARIFYING LINKS WITH OTHER ATTACKER GROUPS

Lazarus shares some TTP with other North Korean APT groups and has been using crimeware malware.

- **APT37 (Reaper)**

  Other names:
  - Reaper (FireEye)
  - Ricochet Chollima (CrowdStrike) ScarCruft
  - Red Eyes

  APT37 is another North Korean attacker group focusing on the Middle East and South Korea. Reaper uses its own set of malware and infrastructure, and its activities don’t overlap with Lazarus’. The first known attack attributed to APT37 was traced back to 2014. They rely strongly on known or 0day exploits and spear phishing to infect their victims.

  The group was publicly exposed by FireEye.

- **APT38 (Bluenoroff)**

  APT38 targets financial companies mostly in Asia. The first known operation took place in 2014 according to FireEye. The group was publicly exposed by FireEye. This report doesn’t clearly draw a link between APT38 and Lazarus subgroup Bluenoroff, which comes from the fact that FireEye classify APT groups following its own strict rules and criteria. To remove any confusion, we will be less rigorous than FireEye and consider APT38 to be Bluenoroff, based on malware code overlaps and TTPs. See the “Classification” part of this report for technical links with Lazarus.

  APT38 TTP resemble those of Lazarus subgroups, especially how they carry out their attacks and chose their targets. They have been focusing on attacking banks connected to the SWIFT network. They will most of the time infiltrate a bank network through vulnerable exposed servers, spend months gathering information, doing reconnaissance and moving laterally in the network until they find a way to steal money. Once the theft is complete, they will try to destroy all evidence by deploying crimeware ransomware or wipers.

  APT38 has its own toolset to maintain persistence, move laterally and manipulate SWIFT transactions. Their targets are diverse and worldwide: Russia, Turkey, USA, Uruguay, Brazil, Vietnam, etc. This group has shown some amateurism and carelessness despite being quite sophisticated, which is a common trait amongst North Korean APT groups.

- **Clarifying links with TA505 (Emotet, TrickBot & Dridex)**

  TA505 is a financially-motivated threat actor mostly operating from Russia. This actor is known for phishing campaigns using banking trojans such as Dridex and TrickBot, ransomware campaigns deploying Locky and the wide use of the Emotet loader.

  - **McAfee’s mistake**

    Since early 2019, some reports mentioning links between Emotet/TrickBot and Lazarus were published. It appears, however, that these reports were filled with misconceptions and faulty logic, which led to misattributions.

    Emotet is one of the most common malware loaders in the wild. It has been used by the TrickBot gang to install their eponymous banking trojan. Both Emotet and TrickBot are believed to come from the Russian cybercrime.

    In late 2018, Emotet and TrickBot were seen deploying a ransomware called Ryuk in well-funded companies’ infected networks. Contrary to most ransomware, Ryuk asks for a huge amount of money
to decrypt files, sometimes more than $100,000 (see paid ransoms ⁶). Analysis of this malware revealed that it shared most of its code with another crimeware ransomware named Hermes. Hermes was sold on underground hacker forums for as little as $300 in 2017/2018 and was quite popular during those years. Lazarus has been buying and using Hermes to cover their tracks by encrypting disks after a completed operation multiple times. Given these facts, some hasty researchers spread the idea that Ryuk and Lazarus were tied due to Hermes. This was also supported by the fact that researchers reported that they saw previous Lazarus infections cohabit with Emotet and TrickBot, which can also be observed during a forensic mission.

McAfee, in charge of investigating a Ryuk outbreak at that time, published a blogpost to clarify the situation and reveal some findings supporting that Ryuk was in fact coming from a Russian-speaking country and probably linked to the TrickBot gang.

— Latest proof of actual links

In mid-2019, what were initially seen as coincidences became more and more suspicious and some strong links were found during incident response missions, with Lazarus samples being dropped shortly after TA505 malware infected the network. TA505 and Lazarus IOCs were found altogether in bank networks and PowerShell post-intrusion scripts attributed to TA505 and Lazarus appeared to be very similar ⁷ ⁸. From there, it is hard not to consider the fact that the TA505 attackers seem to be selling accesses to bank networks to Lazarus. LEXFO also found TA505 malware (TrickBot and Emotet) during its incident response involving Lazarus, which corroborates these assertions.

**MAIN OPERATIONS (2007 - 2015)**

Lazarus operations have been traced back to 2007. The first attack attributed to Lazarus was a DDoS against South Korean and U.S. websites leveraging the MyDoom botnet. The group has been very active ever since, conducting the operations below (Intezer ⁹):

<table>
<thead>
<tr>
<th>Year</th>
<th>Lazarus campaign</th>
<th>Year</th>
<th>Lazarus campaign</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>Silent Chollima</td>
<td>2015</td>
<td>Tdrop</td>
</tr>
<tr>
<td>2009</td>
<td>MYDOOM</td>
<td>2016</td>
<td>Bangladesh Bank Heist</td>
</tr>
<tr>
<td>2011</td>
<td>10 Days of Rain</td>
<td>2017</td>
<td>WannaCry</td>
</tr>
<tr>
<td>2011</td>
<td>Operation Troy</td>
<td>2017</td>
<td>Hidden Cobra</td>
</tr>
<tr>
<td>2011</td>
<td>SierraBravo</td>
<td>2017</td>
<td>Polish Attacks</td>
</tr>
<tr>
<td>2011</td>
<td>Blockbuster</td>
<td>2017</td>
<td>Ratankba</td>
</tr>
<tr>
<td>2011</td>
<td>Joanap</td>
<td>2017</td>
<td>RokRAT</td>
</tr>
<tr>
<td>2011</td>
<td>KorDLLBot</td>
<td>2018</td>
<td>South Korean Power Grid</td>
</tr>
<tr>
<td>2011</td>
<td>Brambul</td>
<td>2018</td>
<td>GoldDragon</td>
</tr>
<tr>
<td>2013</td>
<td>KorHigh</td>
<td>2018</td>
<td>NavRAT</td>
</tr>
<tr>
<td>2013</td>
<td>DarkSeoul</td>
<td>2018</td>
<td>Lazarus Bitcoin</td>
</tr>
<tr>
<td>2013</td>
<td>KimSuky</td>
<td>2018</td>
<td>NK Gambling</td>
</tr>
<tr>
<td>2014</td>
<td>Destover</td>
<td>2018</td>
<td>RedGambler</td>
</tr>
<tr>
<td>2015</td>
<td>Duuzer</td>
<td>2018</td>
<td>LEXFO’s incident response</td>
</tr>
</tbody>
</table>
II. LAZARUS’ NEW MOTIVES
(2016 - 2019)

FIGHTING SANCTIONS IN THE CYBER SPACE

North Korea has been targeted by multiple rounds of financial sanctions and restrictions. In 2017, the UN and the United States issued many resolutions and orders that had heavy negative impact on North Korea exchanges.

To compensate, we have seen the Lazarus group focus on hacking financial institutions all around the world to steal money. Even though disruptive attacks keep being conducted, it is clear that Lazarus prefer heists involving big sums of money. Likewise, spying operations are still being conducted by North Korea but are usually attributed to the fast-expanding APT37.

The Andariel subgroup illustrates how Lazarus changed its focus from information gathering to financial gains. Precisely, Andariel was actively targeting the defense industry until the end of 2016, when they switched to attacking financial institutions, as showed by the timeline of the main Andariel attacks below: (AhnLab)
<table>
<thead>
<tr>
<th>Date</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2015</td>
<td>Defense</td>
<td>Intelligence</td>
</tr>
<tr>
<td>February 2016</td>
<td>Security company</td>
<td>Intelligence</td>
</tr>
<tr>
<td>April 2016</td>
<td>Defense</td>
<td>Intelligence</td>
</tr>
<tr>
<td>June 2016</td>
<td>Defense</td>
<td>Intelligence</td>
</tr>
<tr>
<td>August 2016</td>
<td>Military</td>
<td>Intelligence</td>
</tr>
<tr>
<td>October 2016</td>
<td>Gambling</td>
<td>Financial gains</td>
</tr>
<tr>
<td>January 2017</td>
<td>Gambling</td>
<td>Financial gains</td>
</tr>
<tr>
<td>March 2017</td>
<td>ATM</td>
<td>Financial gains</td>
</tr>
<tr>
<td>April 2017</td>
<td>Energy</td>
<td>Intelligence</td>
</tr>
<tr>
<td>May 2017</td>
<td>Financial industry</td>
<td>Financial gains</td>
</tr>
<tr>
<td>June 2017</td>
<td>Financial industry</td>
<td>Financial gains</td>
</tr>
<tr>
<td>October 2017</td>
<td>Travel agency</td>
<td>Financial gains</td>
</tr>
<tr>
<td>December 2017</td>
<td>Travel agency</td>
<td>Financial gains</td>
</tr>
<tr>
<td>December 2017</td>
<td>Telecommunications</td>
<td>Spying</td>
</tr>
<tr>
<td>December 2017</td>
<td>Cryptocurrency exchange</td>
<td>Financial gains</td>
</tr>
<tr>
<td>February 2018</td>
<td>Cryptocurrency exchange</td>
<td>Financial gains</td>
</tr>
<tr>
<td>February 2018</td>
<td>Politics</td>
<td>Spying</td>
</tr>
<tr>
<td>October 2018</td>
<td>ATM (FastCash)</td>
<td>Financial gains</td>
</tr>
</tbody>
</table>

**BANKS & ATM**

Most bank attacks are carried out by the Bluenoroff subgroup, while ATM attacks are usually attributed to Andariel. In both cases, two methods were leveraged:

- Spear phishing
- Watering hole
- Vulnerabilities in specific and targeted software directly to perform supply chain attacks

One of largest attacks occurred in early 2017, when it was discovered that more than twenty Polish banks were infiltrated by Lazarus. The financial loss is unknown but the scale of the attack and its success is a testament to how capable the attackers are. Bank employees were targeted by several watering holes delivering a payload through a known Silverlight exploit (CVE-2016-0034).

Lazarus also unsurprisingly targets ATM to steal credit card information. Lazarus targeted the ATM operator VANXATM in February 2015. The attack was sophisticated and leveraged a 0day in the antivirus software as well as a bad configuration of the update server allowing the attackers to install their malware on more than 60 connected ATM. It was reported that 230,000 unique credit card information numbers were exfiltrated to Lazarus C&C. The attack was attributed to the Andariel subgroup.

Another example of a successful ATM attack by Lazarus was uncovered by US-CERT and Symantec and was named “FASTCash campaign”. This attack successfully targeted banks in Asia and Africa, and forced issuing banks to accept fraudulent withdrawal requests. Different tailor-made malware were used in each attack. Such an attack involving ATM jackpotting requires physical presence and a mule network, showing how experienced Lazarus attackers are in carrying out advanced cybercriminal operations. Tens of millions of dollar were successfully stolen from banks.

Lazarus has also been targeting Point-of-Sale businesses with the Ratankba malware family they developed, showing that they don’t miss any opportunity to make quick money using custom tools.
TARGETING CRYPTOCURRENCY BUSINESSES

Lazarus attackers have recently been focusing on hacking cryptocurrency businesses, with a particular emphasis on South Korean exchanges. These attacks are very profitable and most of the time quite unsophisticated, making them the perfect way for stealing money. The most significant attack was against Coincheck and ended up with Lazarus stealing about $534 million.

In 2018, Kaspersky uncovered a Lazarus attack they called “Operation AppleJeus”. The attack was sophisticated and targeted cryptocurrency users and exchanges. Victims were infected by a backdoored MacOS cryptocurrency trading software. Most samples used were compiled in 2017.

In the end of 2017, ProofPoint uncovered a new implant named PowerRatbanka. This malware was developed using PowerShell, which shows that Lazarus attackers are following the trends and their arsenal is in constant development.

Other Lazarus attacks were reported by Group-IB in 2018 against YouBit, Coinis and Yapizon with millions of dollars stolen in each case. All of the exchanges are located in South Korea, and spear phishing was the main intrusion vector.

NEW TOOLSET

Being exhaustive in the description of the Lazarus toolset would be a trite task, as the group is able to quickly develop custom malware for each target. They have also been seen using malware from other criminal groups, particularly ransomware, to make attribution harder and cover their tracks.

For instance, some Lazarus malware were found alongside Emotet and Trickbot, and the attackers will execute ransomware such as Hermes to hide their activities and fingerprints after a successful operation.

Recently, a new specific malware toolset was used by Lazarus in different attacks. LEXFO investigated such an attack involving malware from this set and will describe its findings in the next part.

LEXFO also noticed that the attackers were no longer using the VisualStudio C++ v6 compiler, and the most recent samples found were compiled using VisualStudio C++ v8.

- MacOS malware

Kaspersky uncovered an attack attributed to Lazarus leveraging a trojanized cryptocurrency trading application for MacOS. This discovery showed that the North Korean group is not slowing down and keeps improving its technical capabilities. The malware could be attributed to Lazarus mostly because of a hardcoded RC4 key found in other Lazarus malware and a reused C&C domain.

In the fall of 2019, TrendMicro also published a blog article where they uncovered a MacOS variant of the Nukesped trojan found in the wild, attributed to Lazarus.

- Mobile malware

Lazarus expanded their capabilities and developed their first mobile malware in 2017, by adding malicious code to a legitimate APK. This malware was discovered and analyzed by McAfee in a blogpost. The trojanized Android application was not spread through Google Play. Attribution to Lazarus is based on the communication protocol which was made to hide packets in the legitimate flow of TLS/SSL traffic, and some hardcoded values found in other Lazarus samples.
III. TECHNICAL ANALYSIS OF KEY LAZARUS ATTACKS

LAZARUS TTP

- Attack scheme

Considering the vast amount of attacks carried out by Lazarus throughout the years, it is possible to notice some recurring patterns in the way the group operates. These patterns have not changed much since their first attacks.

- Intrusion through spear phishing, watering hole, brute force or web vulnerabilities
- Network discovery using custom or publicly-available tools
- Gathering credentials through Mimikatz-like tools and keyloggers
- Lateral movements using custom or publicly-available tools
- Fulfilling the attack goal: stealing money and/or information
- Covering tracks by wiping systems or infecting the victims with crimeware malware or ransomware
— **Intrusion**

Lazarus operators use a wide range of tricks to try and infect their victims. Their main vector is spear phishing, sometimes using 0day or known vulnerabilities. They also perform watering hole attacks and RDP password brute-force. Furthermore, they often exploit bad network isolation by hacking into webservers in order to try and access the internal network of a targeted organization. In this way, they were able to reach the server connected to the SWIFT network in the case of the Bangladesh Central Bank attack.

In an attempt at attacking a Chilean bank, the Lazarus operators targeted an employee with a fake job offer. They set up an interview via a Skype call where the targeted employee was tricked into downloading and executing a payload. This shows that the attackers are becoming more and more aggressive.

— **Attempts to confuse attribution**

Lazarus malware developers have been trying to fool researchers by introducing some “false flag” Russian strings as command names. The attempt was not convincing as it was obvious for native speakers that names were lazily translated to Russian. The Russian command names are still used to this day and can be used as a signature. Here are some of them that LEXFO found in a very recent sample:

- Poluchit
- Nachalo
- ssylka
- ustanavlivat
- kliyent2podklyuchit

These strings were used in combination with commercial Russian packers to try and fool researchers and journalists, at a time where they are often too quick to attribute attacks to Russian groups.

— **Malware design**

Lazarus malware usually have the following patterns:

- Multistaged
- Command-line malware and tools
- Designed to be run as Svchost services (for persistence) API are loaded dynamically

Lazarus developers usually forget to strip the PDB path from compiled binaries, even when they disclose valuable information such as what the malware does, its goal, or even the developer’s name.

— **Communications**

Lazarus malware often use a communication protocol that has been named “Fake TLS” for communications. This protocol makes malicious packets look like legitimate TLS handshakes and communications might stay under the radar due to heavy TLS traffic on port 443.

This protocol can be found in most Lazarus malware. It is however hard to detect with Snort and Suricata rules considering the huge stream of TLS/SSL packets to monitor, which explains why it has been consistently used for years by the attackers.

Example of a Lazarus Fake-TLS packet:

```
0000 17 03 01 00 30 5d 15 3d a2 40 ef d2 01 25 ca 54 ....0].=.@...%T
0010 26 5f 5d b0 d2 2f 2f 6d 2d ec 56 85 b0 4c a9 bf &_
0020 eb 97 be 31 ad cd de 3a b4 71 1e c8 53 96 0b 2d ...1...:q..S..-
0030 c3 91 3d a2 15 ..=..
```
A legitimate TLS packet would be structured this way:

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>ApplicationData protocol type</td>
</tr>
<tr>
<td>03 01</td>
<td>SSL version (TLS 1.0)</td>
</tr>
<tr>
<td>00 30</td>
<td>Message length (48 bytes)</td>
</tr>
<tr>
<td>5d ... 15</td>
<td>Encrypted application data</td>
</tr>
</tbody>
</table>

In case of a Lazarus fake-TLS packet, the structure is:

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 03 01</td>
<td>Fake TLS header</td>
</tr>
<tr>
<td>00 10</td>
<td>Size of next packet &lt; 0x4000</td>
</tr>
</tbody>
</table>

The first packet is a fake-TLS handshake sent to the C&C server:

```
0000 17 03 01 00 04 ...
```

Data are then encrypted using algorithms and/or keys different for each malware, usually relying on XOR operations or standard algorithms such as RC4.

Different and more standard communication protocols have been used by Lazarus. Simple HTTP requests with hardcoded URLs were implemented in some cases where attackers didn’t care too much about detection.

Here is an example of a Lazarus HTTP request:

```
GET /sub/lib/lib.asp?id=dn678 HTTP/1.1 Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Host: www.secuvision.co.kr Connection: Keep-Alive
```

### Filenames

For payloads and modules, filenames are generally inspired by legitimate Windows services and end with “xxxsvc.(dll|exe)”:

- `swpsvc.dll`
- `sppsvc.dll`
- `sqcsvc.dll`
- `gpsvc.exe`
- `uploadmgrsvc.dll`
- `wmisecsvc.dll`
- ...

Lazarus has been using `[filename].tmp` and `[filename].dat` filenames for configurations or to store data to be sent to the C&C. Recently, they started using configuration files named `[filename].dll.mui`.

### Persistence

Persistence is usually achieved by setting the main payload as an AUTO_START svchost service, which means the malware will be loaded each time the user session starts through the command `svchost -k [service]`.

---

This text is an excerpt from the document titled "The Lazarus Constellation: Technical Analysis of Some Lazarus Attacks."
---

### Packers

As Lazarus reuse a lot of code in their malware, they manage to evade detection by signature using free and commercial packers. Here is a list of the main packers encountered:

- UPX
- VMProtect
- Themida
- Armadillo
- ASPack
- Enigma
- Protector

---

### Third-party libraries

Lazarus uses statically linked third-party libraries in their malware for communications and TLS/SSL implementation. The following libraries were seen:

- Libcurl (version 7.49.1)
- mbedTLS / PolarSSL
- wolfSSL

The Libcurl library with the same exact version is still being used in the most recent Lazarus samples. To compress data, Lazarus developers usually use inflate/deflate lib versions 1.1.3 and 1.1.4 as well as Zlib version 1.0.1 and 1.2.7.

---

### Third-party tools

Lazarus has its own toolbox, but operators will also use third-party legitimate tools when necessary. They mostly include credential-gathering tools and software allowing lateral movements. Attackers will pack tools that are widely flagged by anti-virus, such as Mimikatz, to evade signature-based detection.

The list of third-party tools includes:

- PsExec
- Mimikatz
- FreeRDP
- SC.exe
- Net.exe
- ...

---

### Encryption

The Lazarus group uses standard and custom encryption algorithms. Custom algorithms are usually based on several XOR operations with constant values, while standard ones are common such as RC4, AES, and DES.

They will sometimes use exotic ciphers like Spritz, an RC4-like algorithm they implemented in a set of malware described by Kaspersky. They have the bad habit of reusing encryption algorithms and keys in different malware, which helps detection and attribution.

Lazarus uses encryption for communications, hiding dynamically-imported API names to avoid heuristics and to encrypt their payloads. For the latter, they also use less sophisticated ways of hiding strings, such as Base64 encoding and alphabet substitution.

We will review some encryption algorithms found in samples below.
III. Technical Analysis of Some Lazarus Attacks

— XOR-based algorithms

Most custom algorithms implemented by Lazarus are based on XOR operations with hardcoded keys. While most of them are pretty straightforward to understand, some are quite imaginative. Here are some algorithms and keys found in multiple Lazarus samples:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Key</th>
<th>Campaign / Malware</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>0xA7</td>
<td>Blockbuster</td>
</tr>
<tr>
<td>XOR</td>
<td>0x9E</td>
<td>Lazarus downloader</td>
</tr>
<tr>
<td>XOR</td>
<td>0x23</td>
<td>FASTCash</td>
</tr>
<tr>
<td>XOR</td>
<td>QzEc, wPoF</td>
<td>Attack on Taiwanese banks + LEXFO incident response</td>
</tr>
<tr>
<td>XOR-based</td>
<td>0xF4F29E1B</td>
<td>Lazarus under the hood</td>
</tr>
<tr>
<td>XOR-based</td>
<td>0xC8F9A345</td>
<td>Lazarus under the hood</td>
</tr>
<tr>
<td>XOR-based</td>
<td>0xF833D0B</td>
<td>Lazarus under the hood</td>
</tr>
<tr>
<td>XOR-S^</td>
<td>/</td>
<td>Phandoor (Troy)</td>
</tr>
<tr>
<td>XOR-1FE</td>
<td>/</td>
<td>Phandoor</td>
</tr>
<tr>
<td>XOR-7F8</td>
<td>/</td>
<td>asdfdoor, FBI RAT, Passive backdoor</td>
</tr>
<tr>
<td>XOR-FFFFFFF0</td>
<td>/</td>
<td>Rifle</td>
</tr>
</tbody>
</table>

Below is an example of a custom XOR-based encryption algorithm using hardcoded keys and constants found in several Andariel samples.

```
lpBuffer = buff;
LOBYTE(key4) = 0x82u;
v13 = buff;
key3 = 5;
key1 = 0x556F9482;
key2 = 0xAFC12058;
if ((signed int)dwSize > 0 ){
    offset = encryptedBuffer - (char *)lpBuffer; i = dwSize;
    do{
        *lpBuffer = key3 ^ key2 ^ key4 ^ lpBuffer[offset];
        key3 = key3 & key2 ^ key4 & (key3 ^ key2);
        key4 = ((((unsigned int16)key1 ^ (unsigned int16)(8 * key1)) & 0x7F8) << 20) | (key1 >> 8);
        key2 = (((key2 << 7) ^ (key2 ^ 16 * (key2 ^ 2 * key2)) & 0xFFFFFF80) << 17) | (key2 >> 8);
        ++lpBuffer; nbBytesLeft = i--; nbBytesLeft = i-- == 1;
        key1 = ((((unsigned int16)key1 ^ (unsigned int16)(8 * key1)) & 0x7F8) << 20) | (key1 >> 8);
    } while ( !nbBytesLeft );
    lpBuffer = v13;
    size = dwSize;
}
```

A good example of code reuse is the “S^” algorithm (S-hat) recently seen in many Andariel malware compiled in 2016/2017. We found traces of the same algorithm in samples used in Operation Troy, compiled in 2010 and 2011 in a payload named bs.dll 31.

— RC4

The RC4 algorithm is found in a number of malware, as it is easy and quick to implement. Lazarus developers will sometimes modify it lightly and double the PRGA part of the algorithm to confuse analysts.

Below are some of the hardcoded keys found in samples 32.
Hardcoded key

4E 38 1F A7 7F 08 CC AA 0D 56 EF F9 ED 08 EF
E2 A4 85 92
f9 65 8b c9 ec 12 f9 ae 50 e6 26 d7 70 77 ac 1e
53 87 F2 11 30 3D B5 52 AD C8 28 09 E0 52 60 D0 6C C5 68 E2 70 77 3C 8F 12 C0 7B 13 D7 B3 9F 15

- AES

The AES algorithm was found in many Lazarus samples: Electricfish, backdoors involved in India attacks, Joanap, various Bluenoroff samples...

- Spritz

The Spritz encryption algorithm is not as common as the others but was used by Lazarus by one of their loaders to decrypt payloads. The key found was:

Hardcoded key

6B EA F5 11 DF 18 6D 74 AF F2 D9 30 8D 17 72 E4 BD A1 45 2D 3F 91 EB DE DC F6 FA 4C 9E 3A 8F 98

- C&C Architecture

Lazarus uses a standard C&C architecture with several layers of proxy servers. These proxies will relay packets from the operators to the implants or the other way around through fake TLS packets.

According to a Group-IB investigation, operators set up a three-layer architecture using non standard ports.

Domains and servers are usually leased in Asian countries and paid with bitcoins or other cryptocurrencies for anonymity. Lazarus used to leverage hacked servers for their C&C infrastructure but recent attacks show that they have moved away from it.

From a geographic point of view, most C&C appear to be hosted in the US and in Asian countries. The diagram below shows locations of more than 50 C&C that have been used by Lazarus in different attacks the past two years.
## MITRE ATT&CK MATRIX

### Techniques used

The ATT&CK matrix related to Lazarus clearly shows how active and diverse the group is.

<table>
<thead>
<tr>
<th>Initial Access</th>
<th>Execution</th>
<th>Persistence</th>
<th>Privilege Escalation</th>
<th>Defense Evasion</th>
<th>Credential Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drive-by Compromise</td>
<td>Command-Line Interface</td>
<td>Account Manipulation</td>
<td>Access Token Manipulation</td>
<td>Access Token Manipulation</td>
<td>Account Manipulation</td>
</tr>
<tr>
<td>Spearphishing Attachment</td>
<td>Compiled HTML File</td>
<td>Bootkit</td>
<td>New Service</td>
<td>Compiled HTML File</td>
<td>Brute Force</td>
</tr>
<tr>
<td>Exploitation for Client Execution</td>
<td>Hidden Files and Directories</td>
<td>Process Injection</td>
<td>Connection Proxy</td>
<td>Credential Dumping</td>
<td></td>
</tr>
<tr>
<td>Scripting</td>
<td>New Service</td>
<td>Disabling Security Tools</td>
<td>Input Capture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Execution</td>
<td>Registry Run Keys / Startup Folder</td>
<td>File Deletion</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Management Instrumentation</td>
<td>Shortcut Modification</td>
<td>Hidden Files and Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Discovery</th>
<th>Lateral Movement</th>
<th>Collection</th>
<th>Command And Control</th>
<th>Exfiltration</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Window Discovery</td>
<td>Remote Desktop Protocol</td>
<td>Data from Local System</td>
<td>Commonly Used Port</td>
<td>Data Compressed</td>
<td>Data Destruction</td>
</tr>
<tr>
<td>File and Directory Discovery</td>
<td>Remote File Copy</td>
<td>Data Staged</td>
<td>Connection Proxy</td>
<td>Data Encrypted</td>
<td>Disk Content Wipe</td>
</tr>
<tr>
<td>Query Registry</td>
<td></td>
<td>Data Encoding</td>
<td>Exfiltration Over Command and Control Channel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Information Discovery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Network Configuration Discovery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Owner/ User Discovery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Time Discovery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Uncommonly Used Port | Standard Cryptographic Protocol | Service Stop | System Shutdown/ Reboot | Resource Hijacking | |

| Standard Application Layer Protocol | | | | |
| Remote File Copy | | | | |
Software

Similary, they have been using the set of software below (list is not exhaustive):

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0347</td>
<td>AuditCred</td>
</tr>
<tr>
<td>S0245</td>
<td>BADCALL</td>
</tr>
<tr>
<td>S0239</td>
<td>Bankshot</td>
</tr>
<tr>
<td>S0181</td>
<td>FALLCHILL</td>
</tr>
<tr>
<td>S0246</td>
<td>HARDRAIN</td>
</tr>
<tr>
<td>S0376</td>
<td>HOPLIGHT</td>
</tr>
<tr>
<td>S0271</td>
<td>KEYMARBLE</td>
</tr>
<tr>
<td>S0002</td>
<td>Mimikatz</td>
</tr>
<tr>
<td>S0108</td>
<td>netsh</td>
</tr>
<tr>
<td>S0238</td>
<td>Proxysvc</td>
</tr>
<tr>
<td>S0241</td>
<td>RATANKBA</td>
</tr>
<tr>
<td>S0364</td>
<td>RawDisk</td>
</tr>
<tr>
<td>S0263</td>
<td>TYPEFRAME</td>
</tr>
<tr>
<td>S0180</td>
<td>Volgmer</td>
</tr>
<tr>
<td>S0366</td>
<td>WannaCry</td>
</tr>
</tbody>
</table>
IV. INCIDENT RESPONSE: HOW TO UNCOVER AN ONGOING LAZARUS ATTACK

CONTEXT

In late December 2018, LEXFO was contacted by a company following multiple infections. The company was alerted of outgoing malicious traffic to a known Lazarus C&C that was being monitored.

About 5 machines were identified as infected in the network at the time. LEXFO immediately asked for RAM and disk dumps of the infected systems, as well as all captured encrypted traffic and began investigating.

FIRST ASSESSMENT

LEXFO was provided with several RAM and disk dumps of the infected machines and three binaries as well as a configuration file and a batch installer, found on the computers and believed to have been used by the attackers.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>igfx.exe</td>
<td>PE32+ executable (GUI) x86-64</td>
<td>260K</td>
</tr>
<tr>
<td>sqcsvc.dll</td>
<td>PE32+ executable (DLL) (GUI) x86-64</td>
<td>2,6M</td>
</tr>
<tr>
<td>sqcsvc.dll.mui</td>
<td>Data</td>
<td>236</td>
</tr>
<tr>
<td>svc.bat</td>
<td>Batch script</td>
<td>643</td>
</tr>
</tbody>
</table>
A quick look at the batch script revealed that its purpose was to deploy and install the RAT payload sqcsvc.dll and its encrypted configuration. The script also takes care of installing a persistent service named sqcsvc.

`svc.bat` installer script content:

```bash
mkdir "c:\programdata\microsoft\sqcsvc"
move "c:\perflogs\1.dat" "c:\programdata\microsoft\sqcsvc\sqcsvc6.ldx"
mov "c:\perflogs\1.dll" "c:\windows\system32\sqcsvc.dll"
mov "c:\perflogs\1.dll.mui" "c:\windows\system32\sqcsvc.dll.mui"
sc create sqcsvc binPath= "%SystemRoot%\System32\svchost.exe -k sqcsvc" start= auto
reg add "HKLM\SYSTEM\ControlSet001\Services\sqcsvc\Parameters"
reg add "HKLM\SYSTEM\ControlSet001\Services\sqcsvc\Parameters" /v ServiceDll /t REG_EXPAND_SZ /d "%SystemRoot%\System32\sqcsvc.dll"
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Svchost" /v sqcsvc /t REG_MULTI_SZ /d sqcsvc
```

Then, LEXFO started reverse-engineering the `sqcsvc` payload that was found in the RAM and disk dumps in order to assess the attackers’ capabilities and find the decryption algorithm for communications.

**ATTRIBUTING THE ATTACK**

Our classifier tool didn’t show any strong link with other Lazarus samples, as the payloads we found were part of the new arsenal of Lazarus at the time. The only link found is a Yara rule match between IGFX and a sample involved in a Lazarus heist in Taiwan (9a776b895e93926e2a758c09e341accb9333edc1243d216a5e53f47c6943c852). The rule matched strings from the static library libcurl with the specific version 7.49.1. We had to investigate further to confirm.

Filenames match Lazarus’ habits, as we have the payload named “*svc.dll”, its encrypted configuration file as a MUI-disguised file and a batch script to install the malware. The payload is also made persistent by registering it as a service, which is how Lazarus usually operate.

Looking closely at the SQCSVC payload metadata, we can see that its original name was `sock_64.dll`, the compilation timestamp is Sat, 03 Nov 2018 00:47:21 UTC which is consistent with North Korea working hours (UTC+9) and that it was packed using Themida Code-Virtualizer. At that point, Lazarus can already be considered the #1 suspect.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Compilation timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>sqcsvc.dll</td>
<td>Sat, 03 Nov 2018 00:47:21 UTC</td>
</tr>
<tr>
<td>igfx.exe</td>
<td>Mon, 02 May 2016 03:24:39 UTC</td>
</tr>
<tr>
<td>hs.exe</td>
<td>Mon, 01 Oct 2018 10:30:58 UTC</td>
</tr>
<tr>
<td>iehelp.exe</td>
<td>Mon, 24 Sep 2018 11:12:22 UTC</td>
</tr>
<tr>
<td>iehelp2.exe</td>
<td>Wed, 14 Nov 2018 14:02:19 UTC</td>
</tr>
<tr>
<td>swpsvc.dll</td>
<td>Sat, 11 Aug 2018 14:14:54 UTC</td>
</tr>
</tbody>
</table>
UNCOVERING ATTACKERS’ ACTIVITIES

Having reverse-engineered the communication protocol and the encryption algorithm, LEXFO started developing a Python implementation to decrypt packets.

Here is the identified decryption function in the SQCSVC payload:

```
XOR decryption stub

def decryptTCPData(data):
    output = ''
    i = 0
    j = 0
    while j < len(data):
        i = 5423
        car = ord(data[j])
        while i >= 1870:
            k = i % 256
            i -= 187
            car = (k ^ (car + k)) & 0xFF
            j += 1
        output += chr(car)
    return output
```

A Python implementation is quite straightforward:

```
def decryptTCPData(data):
    output = ''
    i = 0
    j = 0
    while j < len(data):
        i = 5423
        car = ord(data[j])
        while i >= 1870:
            k = i % 256
            i -= 187
            car = (k ^ (car + k)) & 0xFF
            j += 1
        output += chr(car)
    return output
```

From there, we were able to write a script to automatically decrypt all traffic in the PCAP files exchanged between the implant and the Lazarus C&C.
Other manually executed commands:

```
"cmd.exe" /c "time /t"
"cmd.exe" /c "echo 1000 > c:\windows\temp\tmp1105.tmp"
"cmd.exe" /c "type c:\\windows\temp\tmp1105.tmp"
"cmd.exe" /c "dir c:\\windows\temp\tmp1105.tmp"
"cmd.exe" /c "echo 1000 > c:\windows\temp\tmp1105.tmp"
"cmd.exe" /c "dir c:\\windows\temp\tmp1105.tmp" "cmd.exe"
"cmd.exe" /c "type c:\\windows\temp\tmp1105.tmp" "cmd.exe" /c "type C:\\Windows\Temp\TMP0389A.tmp" "cmd.exe" /c "type C:\\Windows\Temp\TMP0389A.tmp"
"cmd.exe" /c "ping -n 1 XXXROOM0099"
"cmd.exe" /c "ping -n 1 XXXROOM0099"
"cmd.exe" /c "time /t"
"cmd.exe" /c "type C:\\Windows\Temp\TMP0389A.tmp"
```

The Lazarus operators also leveraged the RAT to get information on the infected machines, using the directory and process listing feature of SQCSVC. We decrypted many fragmented packets exfiltrating folders and files as well as running processes.

In some other captures, we saw that the attackers were checking the state of a service named swpsvc. This name if consistent with other Lazarus malware such as the first payload sqcsvc, makes it very suspicious.

```
"[SC] EnumQueryServicesStatus:
OpenService \x1a chec(s) 1060 :\r\n\nLe service sp\x1a cifi\x1a n\'existe pas en tant que service install\x1a .\r\n\n"\r\nSERVICE_NAME: swpsvc \r\n\nTYPE : 30  WIN32  \r\n\nSTATE : 4  RUNNING  \r\n\n(STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)\r\n\nWIN32_EXIT_CODE : 0 (0x0)\r\n\nSERVICE_EXIT_CODE : 0 (0x0)\r\n\nCHECKPOINT : 0x0\r\n\nWAIT_HINT : 0x0\r\n\n```

The Lazarus operators deleted this file when they realized that the company security team was investigating. Fortunately, they failed to delete it safely and LEXFO managed to recover the swpsvc.dll using carving tools.

This payload appeared to be a stage 1 RAT with a similar communication protocol.

Further investigations of the decrypted PCAP files also revealed two other DLL plugins that were sent and written to disk by the attackers: an injector performing payload injection in the explorer.exe process, and a keylogger / screencapper. Both these plugins were unknown at the time.

We provided the client with newly-made YARA rules to detect all discovered payloads as well as a PowerShell script to automate the deployment process. We also implemented Suricata rules to detect the Lazarus fake-TLS and custom protocol traffic that can be used along with our Python script to decrypt the packets. This successfully stopped the attack and helped identify all infected machines.
PAYLOAD ANALYSIS

- **IGFX tool**

This binary was compiled on **Monday, May 02 05:24:39 2016 UTC**. This sample appeared to be a version of the Lazarus tool **Client_TrafficForwarder** described by Group-IB !REF.

This tool’s purpose is to forward traffic to another infected host in order to relay operators’ commands.

One interesting particularity of this tool is that the Lazarus developers used non-native Russian strings for command names, trying to confuse attribution:

```
Translated Russian strings to mess with attribution
```

This binary was compiled with a static version of libcurl v7.49.1, which is common amongst Lazarus’ samples.

- **SQCSVC RAT**

This binary was compiled on **Saturday Nov 03 01:47:21 2018 UTC**.

```
// strConfFile = "C:\Windows\system32\sqcsvc.dll.mui"
file_read((__int64)strConfFile, (__int64)lpBuffer, filesize);
    v38 = 0164;
v39 = 0;
v40 = 0;
v41 = 0;
v37 = 0;
if ( filesize ) {
    i = filesize - 1;
    if ( filesize != 1 ) {
        lpByte = &lpBuffer[i];
        do {
            byte = *(lpByte-- - 1);
            lpByte[1] += byte;
            --i;
        } while ( i );
    }
sizeencoded = filesize - 32;
RC4((__int64)lpBuffer, (__int64)(lpBuffer + 32), sizeencoded);
MD5 hash((__int64)v37, (__int64)(lpBuffer + 32), sizeencoded);
```

SQCSVC configuration decryption
This configuration contains two C&C addresses: member.itemdb.com and 180.235.132.206, both to be contacted on port 443, which is consistent with the Fake-TLS protocol implemented. The payload is packed using a powerful virtualization-based packer called Themida Code-Virtualizer. However, the attackers did not use the packer correctly and the non-obfuscated payload code can be dumped easily from memory.

According to BinDiff, the non-obfuscated payload code is up to 65% similar to the code of the IGFX.exe tool used by the attackers, compiled two years prior to SQCSVC, proving that they probably come from the same developer team or the same code base.

The SQCSVC payload is able to:

- Download and write files on disks
- Execute files or bash commands
- Inject code in a running process
- Listen to commands on a specified port (server mode)
- Rewrite the configuration file with new values

The payload was similar to the one described by TrendMicro after a Lazarus bank heist in Latin America in November, 2018.  

SWPSVC (Stage 1)

The analyzed malware sample of the group Lazarus is a “stage 1” reconnaissance malware which implements Remote Administration Tool features.

The analyzed sample is a DLL library which is loaded by the svchost service, as it is registered as an AUTO_START service for persistence. The delivery method is most likely manual. In such case, the attacker drops the malware on an already compromised machine. The malware configuration is stored encrypted in the registry, unlike most Lazarus malware that come with an encrypted file as configuration. In our case, the configuration data could not be retrieved as it was fully erased before the investigation began by the attackers that didn't need this component anymore since the SQCSVC RAT was installed.

The malware uses different kind of encryption for different kind of purposes. The first substitution-based encryption is used for decrypting encrypted strings in the static binary. The XOR-based encryption is used to obfuscate communications between the server and the client and to decrypt configuration content such as the Command-and-Control (C&C) server name and port number stored in the Windows registry.

This RAT uses the already mentioned Fake-TLS protocol for communications:
The Lazarus Constellation

IV. Incident Response: how to uncover an ongoing Lazarus attack

Fake-TLS handshake sent by the RAT

The following commands are implemented in the RAT:

<table>
<thead>
<tr>
<th>Command ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x19283746</td>
<td>Get information on the infected system (processor architecture, network interfaces...)</td>
</tr>
<tr>
<td>0x1928374C</td>
<td>Write file on system</td>
</tr>
<tr>
<td>0x1928374A</td>
<td>Read file on system</td>
</tr>
<tr>
<td>0x1928374F</td>
<td>Delete file</td>
</tr>
<tr>
<td>0x1928374F</td>
<td>Get process info</td>
</tr>
<tr>
<td>0x19283753</td>
<td>Kill process</td>
</tr>
<tr>
<td>0x1928374D</td>
<td>Create process</td>
</tr>
<tr>
<td>0x19283756</td>
<td>Execute process as a given user</td>
</tr>
<tr>
<td>0x19283748</td>
<td>List files in a directory</td>
</tr>
<tr>
<td>0x19283755</td>
<td>Modify C&amp;C configuration by changing the value in Windows registry</td>
</tr>
<tr>
<td>0x19283747</td>
<td>List local drives and network shares</td>
</tr>
<tr>
<td>0x19283750</td>
<td>Move file</td>
</tr>
</tbody>
</table>

---

Downloaded modules

LEXFO found two downloaded modules in the decrypted packets that were deployed on specific targets.

The first one is an injector that takes a file path as a parameter and injects it in an `explorer.exe` process. The injected file is executed in a new thread. This injector uses RC4 encryption with the hardcoded key to hide suspicious strings that are decrypted at runtime, and will write some log data to the file `C:\windows\temp\temp0917.tmp`.

The second module is a keylogger and screen capper. This file is a DLL originally named `capture_x64.dll` by the attackers. The keylogging and screen capping features are implemented standardly.
V. CLASSIFYING NORTH KOREAN MALWARE AND INTERPRETING LINKS

DATASET

We gathered more than 290 malware attributed to North Korea from various sources:

- Twitter
- Various RE and malware forums
- VirusTotal (Hunting)
- Online sandboxes (HybridAnalysis, Any.RUN...)
- Malware repository (VirusBay, VirusShare, Malshare)
- U.S. Cyber command malware uploads
- Threat intelligence reports
- LEXFO’s own incident responses

We ended up with the following families:

<table>
<thead>
<tr>
<th>Malware family</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>apt38_contopee</td>
<td>polishbanks</td>
</tr>
<tr>
<td>powerratankba</td>
<td>joanapbrambul</td>
</tr>
<tr>
<td>nukesped</td>
<td>bankshot</td>
</tr>
<tr>
<td>killdisk</td>
<td>mydoom</td>
</tr>
</tbody>
</table>
Samples were compiled from 2004-05-23 to 2019-10-22 according to compilation timestamps that seemed legitimate.

**METHODOLOGY**

After several manual analyses of Lazarus samples, we concluded that the following links where relevant:

- **Identifying links**

  After several manual analysis of Lazarus samples, we figured that the following links where relevant:

  - **Standard links:**
    - Code reuse (Fuzzy hashes SSDEEP + MACHOKE)
    - Import hashing
    - Timestamps PDB

  - **Advanced links:**
    - Rich Headers
    - Yara signatures (see next part)
A word on Rich headers

Rich headers are added to standard PE headers in executables compiled using VisualStudio. It is a fingerprint of the compilation environment that can be easily decrypted and decoded. It can then be used to identify if binaries were compiled in the same environment, which is a strong relation. As we empirically saw that North Korean groups have been using VisualStudio almost exclusively, and there is a high chance that their malware-building infrastructure is quite conservative, we chose to develop a script to parse rich headers from samples and included it as a relation link in our classifier.

Building Yara rules

For each North-Korean malware family we identified, we built Yara rules in order to keep signatures of the following implementations that are likely to be reused by Lazarus:

- Specific strings
- Cryptographic algorithms
- Cryptographic keys
- Unique implementations of features:
  - mapping of files
  - lateral movement
  - installing service
  - wiper implementation
  - handling logs
  - ...
- Way of dynamically loading API
- Obfuscation
- ...

We also built rules for statically linked library like OpenSSL, libcurl, ZIP etc. of specific versions, as Lazarus was seen to be pretty conservative in using the same versions over the years. Those rules were named `lib_static_[lib name]_[version]` and we attributed them a lesser weight than implementation rules as it doesn’t illustrate a strong enough link between two samples.

We built a set of about 100 rules that we ran on our sample dataset. To our own Yara rule set, we added auto-generated rules from Malpedia when they showed accurate results.

Before adding them to our ruleset, we ran tests on a huge malware set to make sure that the rules were accurate and there were close to zero false positive.

Building similarity profiles

We produced a profile for each sample with fuzzy hashes, decoded rich header, compilation timestamps and matching Yara rules. We then compared profiles using nearest neighbor algorithms with weight we empirically tested to get the best results. Jaccard distance was used to compare fuzzy hashes. We attributed heavy weights for identical rich headers encountered in different samples and for every non `lib_static_*` Yara rule matches. Weights (W) were roughly according to this order relation:

\[
W(\text{Exact same Rich header}) > W(\text{Yara match (non lib_static_*))}) > W(\text{Machoke code reuse}) > W(\text{Compilation timestamp}) > W(\text{Yara match (lib_static_*))}) > W(\text{Rich header similarity}) > W(\text{Imphash}) > W(\text{Various metadata})
\]

Handling packers

Non-specific packers like UPX are handled separately: as fuzzy hashes become irrelevant, we dismissed them when computing weights. For more specific packers (Themida, Enigma...), we built Yara rules to identify them and considered them as a valid relation of similarity between samples since Lazarus uses specific versions of those packers.

When possible, we reversed the packed samples and tried to get clean unpacked executables so our tool could classify them indiscriminately and accurately.
— Result review and improvements

We ran our tool multiple times and tried to analyze samples that seemed to be oddly placed or unique. We reversed each of them and adapted our classification methodology and criteria according to our findings, and ran the test again and iteratively re-applied this process until the classification was accurate enough.

— Building the graph

From there, we created a graph with samples as nodes and weighted links as relations between them.

VISUALIZATION

We used the Fruchterman-Reingold spacialisation to visualize links and identify clusters. We ended up with the following constellation, where each dot is a sample and each link represents the strength of a relation between two samples:

![Fruchterman-Reingold spacialisation applied to our relation graph](image)

REVIEWING RESULTS

Our tool revealed more than 2500 actual links between around 290 samples, which showcases that such a classification is relevant as Lazarus samples are rarely unique in a 10 years span period. We see clear clusters and many overlaps. This will shed some light on North Korean malware and groups, as the number of reports and campaign names grows and it can be hard to keep track and attribution is often confusing or unsure.

— Kimsuky

The Kimsuky group has its own cluster but we see Rich header and compilation timestamp overlaps with other Lazarus samples. Kimsuky and Lazarus are therefore likely to be working together, which is confirmed by the fact that Kimsuky malware were found on Lazarus targets several times.
Moreover, our tool revealed links between DTrack samples from the Kudankulam Nuclear Power Plant (KNPP) and Kimsuky samples: both use SQLite as a statically-linked library, but a different version (compiled on 2017-10-24 18:55:49 for the latter vs 2017-02-13 16:02:48 for the former). Looking at compilation timestamps, we can see that some DTrack and Kimsuky samples were also compiled the same day (or close to) as other Lazarus malware used in campaigns:

(2019-07-29 13:36:26) ./dtrack/np
bfb39f486372a509f307cde3361795a2f9f759cbeb44ac07562dcbab6c00364
= Timestamp = (2019-07-29 07:08:01) ./andariel_rifle/javaupdatemain_unpack.exe

(2019-03-01 00:07:25) ./dtrack/npp_3cc9d9a17f3b884582e5c4daf7d83c4a510172a8346de90b8743398e3cbe3682
= Timestamp = (2019-03-01 09:08:44) ./kimsuky_shark2/4b3416fb61ed1f767277b24dd4f652e63ba41f7809825c5f0e9010f7da7.bin

This could mean that the groups are working together for some operations, with Lazarus doing the intrusion and handing the exfiltrating part to Kimsuky when the target matches their interest.

Finally, interesting findings stand out when looking at Rich header similarities. The Kimsuky stolen pencil sample has the exact same Rich header as samples found in Lazarus campaigns such as DarkSeoul and GoldDragon:

(2018-12-21 00:34:35) ./kimsuky_stolenpencil/1.bin

- Rich -> (2012-07-06 12:24:18) ./troydarkseoul/DarkSeoul/DarkSeoul_S0E-03200C3A8EBCBF333B388DC4CD46
- Rich -> (2017-12-24 08:16:57) ./golddragon/e6f843eb03330f6420a07b61933583b4144585
- Rich -> (2017-12-24 08:47:21) ./golddragon/4f586e6a7a04be2b2eacbc0dcbae6f281778f9dbd9f9
- Rich -> (2017-12-24 08:29:04) ./golddragon/11a38a9d23193d9582d20ab0eae
- Rich -> (2017-12-24 08:37:57) ./golddragon/3a0c617d177f819775e48f7ede-f9af84a1446b
- Rich -> (2017-12-24 08:44:08) ./golddragon/bf21667e4b4b8b857020baa455331c-9c4f2569740

- DarkHotel

Samples attributed to the DarkHotel group have identical Rich header as a lot of APT38 Nukesped samples, which is a strong link:

(2011-04-07 06:58:03) ./darkhotel/2b628b8bb811bb9d4666c3a0372f26ed64c8c9ff11c604037982d1654fb9e850.ViR
- Rich -> (2017-07-14 22:40:25) ./cybercom/dd2675a8ade6e9d8c14615484ff62.bin
- Rich -> (2017-07-11 18:26:59) ./nukesped/3DCE64D49A2F31BB8A98A0DB86E54563
- Rich -> (2017-08-11 05:03:45) ./cybercom/2a791769a73ac757f210f8546125b57.bin
- Rich -> (2017-08-01 16:39:36) ./ghostsecret/Sample_5ae56e2077dc8d38c3bf7d
Though DarkHotel TTPs and malware are different from Lazarus, those groups seem to be working in tandem.

- **Andariel subgroup**

Clear Andariel clusters stand out, with malware involved in operations Red Gambler and Rifle. Those samples are closely linked with each other by specific and custom cryptographic algorithms found in malware from both operations. For instance:

```plaintext
yara_andariel_7F8: (2016-04-21 10:41:15) ./andariel_rifle/d246669cf1e25860f8601e456edd7156aa7384026ff4eadac18a2a82a18fabbf
yara_andariel_7F8: (2016-12-01 13:56:28) ./redgambler/9a50be3def3681242f35d3c0911e2e70
yara_andariel_7F8: (2017-03-21 16:05:58) ./redgambler/2573d0ad00f4aba8ee86d7ffe745d963_unpack
```

RedGambler samples seem to be related to older samples from the Troy/DarkSeoul operation as well as an APT37 Navrat sample by their Rich headers which are identical:

```plaintext
(2016-12-01 13:56:28) ./redgambler/9a50be3def3681242f35d3c0911e2e70
<-- Rich --> (2016-05-01 05:53:43) ./apt37_summit/navrat_old_e0257d187be69b9beea0a731437bf050d56213b50a6fd29dd6664e7969f286ef.bin
<-- Rich --> (2017-03-21 16:05:58) ./redgambler/2573d0ad00f4aba8ee86d7ffe745d963_unpack
```

Other Andariel samples are linked with various Lazarus samples. For example, Andariel uses inflate library version 1.1.4 which is the same version found in several other North Korean samples (GoldDragon, Fallchill, Dtrack...).
APT38/Bluenoroff

Bluenoroff clusters are linked by Rich headers, timestamps and code similarity. Most of the links are quite strong and make APT38 clusters the most distinguishable ones, meaning that the group doesn’t think it’s necessary to be sneaky and reinvent itself, but will reuse a lot of elements, from architecture to malware implementations. These clusters are mainly composed of the following malware families:

- Nukesped
- Fallchill
- Volgmer
- Electricfish
- Dyepack
- SWIFT-related malware
- Hoplight
- Some Sony / Blockbuster samples
- Malware from bank attacks (Poland, Vietnam...)
- Destover
- Bankshot
- Fastcash
- ...

Looking at the links, we can see that a Yara rule we built is matching almost 40 samples from our dataset, all of them attributed to APT38. The Yara rule was built to detect a specific RC4 implementation and called `yara_apt38_rc4`:

```
rule yara_apt38_rc4 { strings:
    $s1 = { 8A 90 01 01 00 00 // mov dl, byte [eax + 0x101]
          8A 88 00 01 00 00 // mov cl, byte [eax + 0x100] 8A 14 02 // mov dl, byte [edx + eax]
          8A 1C 01 // mov bl, byte [ecx + eax]
          02 D3 // add dl, bl
          8A 1C 2E // mov bl, byte [esi + ebp]
          81 E2 FF 00 00 00 // and edx, 0xff
          8A 0C 02 // mov cl, byte [edx + eax]
          32 CB } // xor cl, bl
condition:
    uint16(0) == 0x5a4d and any of ($s*)
}
```

This rule showcases once again that Lazarus groups reuse a lot of code for their malware. Here are some of the samples using this RC4 implementation:

```
yara_apt38_rc4: ./apt38_contopee/766d7d591b9ec1204518723a1e5940fd6ac777f606ed64e731fd91b0b4c3d9fc.bin
yara_apt38_rc4: ./nukesped/3EDCE4D49A2F31B8BA9BAD0B8EF54963
yara_apt38_rc4: ./nukesped/sample2.bin
yara_apt38_rc4: ./nukesped/sample3.bin
yara_apt38_rc4: ./nukesped/34E560565741F33D823859E7723SED9 yara_apt38_rc4: ./nukesped/sample (9).bin
yara_apt38_rc4: ./nukesped/sample (1).bin
yara_apt38_rc4: ./nukesped/F315BE41D976D9A6D6F084D29E4300
yara_apt38_rc4: ./cybercom/38fc56965dccc218f39a945f9ebc3f49.bin
yara_apt38_rc4: ./cybercom/5Cc1b4c3b1cf0d455ac05ace994aed4b.bin
```

This rule showcases once again that Lazarus groups reuse a lot of code for their malware. Here are some of the samples using this RC4 implementation:
Other Yara rules are matching several APT38 samples from different malware families: some related to file wiping implementations, Fallchill success codes, string decoding algorithms, inflate 1.1.3 strings...

On another hand, Rich header analysis reveals that some recent malware found in India, Vietnam and Taiwan, as well as samples LEXFO found during incident responses share the same Rich headers, which are strong links.

— WannaCry

WannaCry samples are timestamped, but we see that the WannaCry cluster is close to the Bluenoroff ones. In particular, we see that the wannacry_rand Yara rule we built from the WannaCry sample 3e6de9e2baacf930949647c9981e7a2caea2626df6a468407854aaa51eed9 matches the Contopee malware attributed to APT38 (766d7d91b9ec1204518723a1e5940fd6ac777f606ed64e731fd91b0b4c3d9fc).

yara_wannacry_rand: (2015-02-23 01:32:57) ./wannacry/dropper.bin (2018-04-28 02:53:06) ./vietnam/efd470cfa930949647c39981e7a2caea2626df6a468407854aaa51eed9

Shared code between Bluenoroff Contopee and WannaCry

Most WannaCry samples were statically linked with inflate lib version 1.1.3, which links them to some Bluenoroff samples that are using the exact same version (for instance the recent APT38 keylogger efd470cfa90b91e5d558e5c8c3281343af06eedfd4844f2eb2c4605f9bdc30e used on Vietnamese targets).

yara_lib_static_inflate_113: (2018-04-28 02:53:06) ./wannacry/dropper.bin (2018-04-28 02:53:06) ./vietnam/efd470cfa90b91e5d558e5c8c3281343af06eedfd4844f2eb2c4605f9bdc30e

— DTrack

DTrack is a malware attributed to Lazarus / APT38. Recent DTrack samples found on critical infrastructures like nuclear power plants are linked with a sample from the Troy/DarkSeoul campaign compiled in 2011. The link comes from the reuse of the specific ZIP password dkwero38oeRA^t@#. This is surprising and could be a false flag.
Those DTrack samples are also weakly linked with other Lazarus samples by statically-linked libraries such as TZip and SQLite.

— GoldDragon campaign

GoldDragon samples are linked to Lazarus by two main features: the reuse of a specific RC4 implementation that was seen in old Joanap dropper samples and detected by our Yara rules, and the overlaps of rich headers. Here is an example of a GoldDragon sample sharing its Rich header with other known Lazarus samples (as well as other GoldDragon samples):

(2017-12-24 08:37:57) ./golddragon/3a8c617d17e7f819775e48f7ede9e9af8a41a466b

<- Rich -> (2017-12-24 08:29:04) ./golddragon/11a38a9d32313d9582d92aba0eae-767c3933066ec
<- Rich -> (2017-12-24 08:44:08) ./golddragon/bf21667e4b48b857020ba455531c-9c4f5f50740
<- Rich -> (2017-12-24 08:47:21) ./golddragon/4f58e6a7a04b2b2ebecdbcb6e6f-281778fd9d99
<- Rich -> (2018-12-21 00:34:35) ./kimsuky_stolenpencil/1.bin
<- Rich -> (2017-12-24 08:16:57) ./golddragon/e6f43ecb03330f-0f049e47b61935539b444585

Another link is the statically-linked inflate v. 1.1.4 that we found in GoldDragon samples, as this version is widely used in a lot of Lazarus samples.

— APT37

Samples attributed to APT37 (Reaper) seem to be quite unique and only linked with Lazarus samples by statically-linked library or encryption algorithms, which are weak links. This confirms what FireEye stated in its report: this group needs to be tracked separately from Lazarus.

A lot of APT37 samples share the same Rich header. We also found the following identical Rich headers between an APT37 malware and a Bluenoroff Nukesped sample:

(2019-01-02 01:43:47) ./.apt37_ evilnewyear/2019_636844ce36f41641d854a1b239df91daa103873d3cde0c25087582e3f64e4d.bin
<- Rich -> (2018-02-12 20:06:28) ./cybercom/07d2b057d2385a4cdef413ed342305df.bin
<- Rich -> (2018-02-12 20:06:28) ./nukesped/07D2B057D2385A4CDFA413ED342305DF

Finally, we found a Navrat sample attributed to APT37 and an Andariel sample from the RedGambler operation with the same Rich header, which connects the two groups (see the part about Andariel below).
To complicate attribution, the attackers behind OlympicDestroyer copied a Rich header from Lazarus samples to replace the rich header of some of their malware. Our tool gives the following result, showing that the Rich header was taken from Bluenoroff samples (one of them from the Bangladesh SWIFT heist):

(2017-12-27 09:03:48) ./olympicdestroyer/3c0d740347b0362331c882c2dee96dbf
<- Rich -> (2016-02-04 13:45:39) ./bangladesh_swift/evtsys.exe
<- Rich -> (2017-03-02 16:46:13)
./blockbuster_sequel/032ccd6ae0ae49ac93b7bd10c7d249f853fff3f5771a1fe3797f733f09db5a0.bin

Kaspersky published an article about this false flag operation 38.

**WORKING HOURS AND DAYS OF THE LAZARUS DEVELOPERS**

We extracted all compilation timestamps from the samples in our dataset and removed those that were either altered or inconsistent. Some samples appeared to be legitimate software infected by Lazarus without recompiling, making the timestamps irrelevant. For instance, we ignored the sample 2223a93521b261715767f0f0f0d1ae4e692bd593202be40f3508cb4fd5e21712b which turned out to be a version of the FTP tool FileZilla that the attackers altered by adding some malicious code without recompiling it, leaving its original compilation timestamp and compiler fingerprints unmodified.

Analyzing unaltered compilation timestamps, we see that the Lazarus developers are mostly working between 8AM and 8PM UTC+9 (KST). We can even notice some breaks at lunchtime, and that Lazarus developers are working overnight. Most samples were compiled from Monday to Saturday included.
CONCLUSION

Such a classification proved to be very relevant for North Korean malware. It highlighted heavy links illustrating code and architecture reuse inside established groups, as well as relations (or lack of) between these separate groups.

Attacker groups like Lazarus are so active they struggle or are reluctant to renew their arsenal. Studying their TTPs prove to be very valuable and will greatly help properly reacting to incident. As a defender, being able to exploit Lazarus laziness and carelessness by quickly identifying their TTPs will give you some key information: you know what they want, how they plan to achieve it and with which tools.

The information given in this report, the classification LEXFO established and the associated internally developed tools helped the incident response team practically during missions involving Lazarus, as it narrowed down the analysis and gave good hints on where to look for technical clues: persistence, communications, lateral movements, exfiltration etc.
VI. DETECTION & MITIGATION

VULNERABILITY USED

North Korean groups have been exploiting a lot of vulnerabilities, such as 0days and as 1days. Most exploits target Adobe Flash Player as well as the Hangul Word Processor, though groups like Andariel have also been seen finding and exploiting vulnerabilities in specific corporate software. The list of CVE that have been exploited by DPRK groups below shows once again that keeping its software updated is crucial.

- Lazarus

Lazarus and its subgroups Andariel and Bluenoroff often rely on software vulnerabilities to infect their targets. Here are some of them:

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Oday</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2014-0497</td>
<td>Yes</td>
<td>Flash exploit</td>
</tr>
<tr>
<td>CVE-2015-6585</td>
<td>Yes</td>
<td>Vulnerability in HWP</td>
</tr>
<tr>
<td>CVE-2015-8651</td>
<td>No</td>
<td>Flash exploit</td>
</tr>
<tr>
<td>CVE-2016-0034</td>
<td>Yes</td>
<td>Silverlight exploit</td>
</tr>
<tr>
<td>CVE-2016-0189</td>
<td>Yes</td>
<td>Internet Explorer Scripting Engine Remote Memory Corruption Vulnerability</td>
</tr>
<tr>
<td>CVE-2016-1019</td>
<td>No</td>
<td>Flash exploit</td>
</tr>
<tr>
<td>CVE-2016-4117</td>
<td>Yes</td>
<td>Flash exploit used in watering hole attacks</td>
</tr>
<tr>
<td>CVE-2017-0261</td>
<td>Yes</td>
<td>EPS restore use-after-free</td>
</tr>
<tr>
<td>CVE-2018-8373</td>
<td>Yes</td>
<td>VBScript Engine vulnerability used by the DarkHotel subgroup</td>
</tr>
<tr>
<td>CVE-2018-4878</td>
<td>Yes</td>
<td>Flash exploit used by APT37 and Lazarus</td>
</tr>
<tr>
<td>CVE-2018-20250</td>
<td>No</td>
<td>WinRar exploit targeting Israeli companies</td>
</tr>
<tr>
<td>CVE-2018-8174</td>
<td>Yes</td>
<td>Internet Explorer VBS engine vulnerability</td>
</tr>
</tbody>
</table>
APT37 / Reaper

APT37 usually exploits 1day to target unpatched systems, mostly through Adobe vulnerabilities. The vulnerabilities below were attributed to APT37 by FireEye:

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Oday</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2013-4979</td>
<td>No</td>
<td>Buffer overflow in EPS Viewer</td>
</tr>
<tr>
<td>CVE-2014-8439</td>
<td>No</td>
<td>Adobe Flash Player arbitrary code execution</td>
</tr>
<tr>
<td>CVE-2015-2387</td>
<td>No</td>
<td>Adobe Type Manager Font Driver memory corruption vulnerability</td>
</tr>
<tr>
<td>CVE-2015-2419</td>
<td>No</td>
<td>Internet Explorer JScript RCE</td>
</tr>
<tr>
<td>CVE-2015-2545</td>
<td>No</td>
<td>Microsoft Office Malformed EPS File Vulnerability</td>
</tr>
<tr>
<td>CVE-2015-3105</td>
<td>No</td>
<td>Adobe Flash Player arbitrary code execution</td>
</tr>
<tr>
<td>CVE-2015-5119</td>
<td>No</td>
<td>Adobe Flash Player Use-After-Free leading to code execution</td>
</tr>
<tr>
<td>CVE-2015-5122</td>
<td>No</td>
<td>Adobe Flash Player Use-After-Free leading to code execution</td>
</tr>
<tr>
<td>CVE-2015-7645</td>
<td>No</td>
<td>Adobe Flash Player vulnerability</td>
</tr>
<tr>
<td>CVE-2016-1019</td>
<td>No</td>
<td>Adobe Flash Player vulnerability</td>
</tr>
<tr>
<td>CVE-2016-4117</td>
<td>No</td>
<td>Adobe Flash Player vulnerability</td>
</tr>
<tr>
<td>CVE-2017-0199</td>
<td>No</td>
<td>Microsoft Office/WordPad Remote Code Execution Vulnerability</td>
</tr>
<tr>
<td>CVE-2018-4878</td>
<td>Yes</td>
<td>Flash exploit also used by Lazarus</td>
</tr>
</tbody>
</table>

DETECTING LAZARUS ACTIVITIES

Network detection rules

US-CERT Snort rules to detect Fake TLS packets:

```
alert tcp any any -> any any (msg:"Malicious SSL 01 Detected"; content:"\x17 03 01 00 08"); pcre:"/\x17\x03\x01\x00\x08\x04\x88\x4d\x76/"; rev:1; sid:2;)
alert tcp any any -> any any (msg:"Malicious SSL 02 Detected"; content:"\x17 03 01 00 08"); pcre:"/\x17\x03\x01\x00\x08\x06\x88\x4d\x76/"; rev:1; sid:3;)
alert tcp any any -> any any (msg:"Malicious SSL 03 Detected"; content:"\x17 03 01 00 08"); pcre:"/\x17\x03\x01\x00\x08\x04\x30\x7b\x7f/"; rev:1; sid:4;)
alert tcp any any -> any any (msg:"Malicious SSL 04 Detected"; content:"\x17 03 01 00 08"); pcre:"/\x17\x03\x01\x00\x08\x04\x30\x7b\x7f/"; rev:1; sid:5;)
```

The following rule will specifically detect the SWPSVC RAT LEXFO discovered:

```
alert tcp any any -> any any (msg:"Lazarus Stage 1 SWPSVC Handshake"; dsize:5; content:"\x17 03 01 00 04");)
```

Yara rules

LEXFO produced the following YARA rules to sign and allow detection of the latest Lazarus samples encountered during the investigation.

The rules `lazarus_forward_libcurl` and `themida_virtualizer` can produce false-positives, as they will respectively detect any file with a specific statically compiled libcurl library and files packed with Themida Code-Virtualizer, which can be legitimate in some cases. These rules will work on uncompressed disk and memory dumps, as well as network capture files.
The full Yara ruleset we used for this report will be available to our clients.
RECOMMENDATIONS

Several Lazarus infection vectors can be severely mitigated to prevent or block an attack.

— Preventing an infection

The WannaCry incident showed how important it is keeping one’s OS updated. Lazarus will certainly continue to implement and leverage such 1day vulnerabilities to target unpatched systems quickly after a fix is deployed.

As shown in this report, Lazarus leverages known vulnerabilities in webservers to try and get a first access to the internal network of a target. To mitigate this vector, it is necessary to make sure all exposed servers and their components are up-to-date and isolated from the internal networks of the organizations.

Furthermore, Lazarus leveraged several 0day and 1day vulnerabilities in popular software such as Flash Player, HWP and Silverlight. Keeping those software up-to-date is mandatory. The group is also able to quickly find and exploit vulnerabilities in custom internal software used by companies, sometimes leading to supply chain attacks. Auditing software used internally is also advised to mitigate this vector.

— Mitigating lateral movements

Lazarus uses mostly legitimate tools for lateral movements. When a form of authentication is needed, they will either reuse stolen passwords gathered with Mimikatz-like tools or keyloggers or try to bruteforce it with dictionaries.

Tools like PSExec can be monitored through log analysis. As Lazarus implants usually achieve persistent by installing services, event id 7045 and 4697 with the Service_Start_Type information set to SERVICE_AUTO_START must be closely monitored.

Last but not least, enforcing a strong password policy is obviously advised.

— Threat intelligence

As Lazarus activities are actively monitored by many security firms such as LEXFO, it is important for security teams to stay up-to-date and follow threat intelligence reports. As we showed in this paper, Lazarus will most of the time reuse known and easy-to-detect communication protocols and tools, and most infections can therefore be prevented.

If any indicator of a compromised system is found, it is strongly advised to quickly contact a specialized firm that knows how the attackers work and can quickly assess the impact of the attack and mitigate it.
APPENDIX A: ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAT</td>
<td>Remote Access Tool</td>
</tr>
<tr>
<td>PCAP</td>
<td>Packet Capture</td>
</tr>
<tr>
<td>MUI</td>
<td>MultiLanguage User Interface extension</td>
</tr>
<tr>
<td>DDoS</td>
<td>Distributed Denial of Service</td>
</tr>
<tr>
<td>TTP</td>
<td>Tactics, Techniques/Tools and Procedures</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>C&amp;C</td>
<td>Command &amp; Control server</td>
</tr>
<tr>
<td>CERT</td>
<td>Computer Emergency Response Team</td>
</tr>
<tr>
<td>APK</td>
<td>Android Package Kit</td>
</tr>
</tbody>
</table>

APPENDIX B: LIST OF STUDIED SAMPLES

Hashes are SHA256.

```
766d7d591b9ec1204518723ae16940f606ac777f686ed64e731f91d1bb4cd39fcd4de16f97064030a5a2fa9872623a4693e49c58df5c753ccc684f3d3542e295c7f0e03547bcba00a4d051bf14908f5230c6c5fd16686d09861c7f7f48aad99027e9f0a149939206bfac8c2de741e5b32bbbf6a0134fe13fda7d764c2a2f8bb710d9295e8a88bb9b2b8d346603a52768b161eda12ebeffe2df4e5c32bd3190a60a8e3f47e3429bf1601f6f88c7fe3a1f13fda7d
8e74c2a67bb710d9295e8a88bb9b2b8601f6f88c7fe3a1f13fda7d
31c1ed43c4a26b0e121b8abdb8aa19ebd378fe2ed761209ff326665c6d61c7b8077c736b9fdd5e4ed3b32a687ae1a69062f4071f6d8c1029f57649878c0d836e0692e2a3f9ada8106safef78b13541cf789a2f6a9f2b2d3aca3be0e425690545efdf73cb6739c1ff1d5f7c49312d3277bf0787db95855888c4b29b1058b2d5a7ca33c6e8f6e52e15b579b48917658c95d5cfd6231c7ce039b172692f9117e7df6f8ff7270f87fe75bde4f7ca2cede63f99ab6c29b73619162dca6a4f0a5b03 
32ec329301aa5447b4e4f8b95785f4b1b6d85a14d3e3e06f2b0c116cfe8f6e52e15b579b48917658c95d5cfd6231c7ce039b172692f9117e7df6f8ff7270f87fe75bde4f7ca2cede63f99ab6c29b73619162dca6a4f0a5b03 
32ec329301aa5447b4e4f8b95785f4b1b6d85a14d3e3e06f2b0c116cfe8f6e52e15b579b48917658c95d5cfd6231c7ce039b172692f9117e7df6f8ff7270f87fe75bde4f7ca2cede63f99ab6c29b73619162dca6a4f0a5b03 
32ec329301aa5447b4e4f8b95785f4b1b6d85a14d3e3e06f2b0c116cfe8f6e52e15b579b48917658c95d5cfd6231c7ce039b172692f9117e7df6f8ff7270f87fe75bde4f7ca2cede63f99ab6c29b73619162dca6a4f0a5b03 
```

0688411344993154a267a9ea5f3f3727d51f95c4a4fde435998f69e6f18571fe3b3a835b3879f6e5e264fad279393c7e866b91c9d64c2b3f89a1fe3a1fe85c084b322ec3f2e9f85ae8e8284a14883ce7eabef668481b91955522e8f81b7cc66ef8645694472edfe13e6f99bc886ba1
```
APPENDIX C: SMB BRUTEFORCE PASSWORD LIST

This password list has been used on numerous occasions by Lazarus to perform SMB bruteforce attacks.

!@#$
!@#$%
!@#$%^&
!@#$%^&*
!@#$%^&*()
0000
00000
0000000
00000000
1111
11111
1111111
11111111
1212
121212
123123
12321
1234
12345
123456
1234567
12345678
123456789
123456789%
1234guwer
123abc
123asdg
123gwe
1313
lgw3le
lgw3we4r
lgwe2wea
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
4321
54321
654321
6969
666666
7777
8888
888888
8888888
88888888
Administrador
admin
admin123
admin123
admin123
as"d
as"dg
as"g
admin
asdf
asdf
asdf
asdf
REFERENCES

2. HiddenCobra, US-CERT
3. FBI boss: Sony hack was DEFINITELY North Korea, haters gonna hate, The Register, 2015
4. APT37 (Reaper) - The Overlooked North Korean Actor, FireEye, 2018
5. APT38 Un-usual suspects, FireEye, 2018
6. Big Game Hunting with Ryuk: Another Lucrative Targeted Ransomware, CrowdStrike, 2019
7. Targeted TrickBot activity drops ‘PowerBrace’ backdoor, NTTSecurity, 2019
8. OSINT Reporting Regarding DPRK and TA505 Overlap, NorfolkInfosec, 2019
9. DPRK cyberattack timeline, Intezer
10. Sanctions against North-Korea, Wikipedia
11. From stealing confidential data to revenue-generating attacks, AhnLab, 2018
12. Full Discloser of Andariel, A Subgroup of Lazarus Threat Group, AhnLab, 2018
14. NATION-STATE MONEYMULE’S HUNTING SEASON - APT ATTACKS TARGETING FINANCIAL INSTITUTIONS, BlackHat EU, 2017
16. FASTCash: How the Lazarus Group is Emptying Millions from ATMs, Symantec, 2018
17. North Korea Bitten by Bitcoin Bug - Financially motivated campaigns reveal new dimension of the Lazarus Group, Proofpoint, 2017
18. Lazarus Under The Hood, PDF, Securelist, 2017
19. Group-IB: 14 cyber attacks on crypto exchanges resulted in a loss of $882 million, Group-IB, 2018
20. Operation AppleJeus: Lazarus hits cryptocurrency exchange with fake installer and macOS malware, Securelist, 2018
22. Operation AppleJeus: Lazarus hits cryptocurrency exchange with fake installer and macOS malware, Securelist, 2018
23. Mac Backdoor Linked to Lazarus Targets Korean Users, TrendMicro, 2019
24. Android Malware Appears Linked to Lazarus Cybercrime Group, McAfee, 2017
25. Lazarus Arisen - ARCHITECTURE / TOOLS / ATTRIBUTION, Group-IB, 2017
26. Disclosure of Chilean Redbanc Intrusion Leads to Lazarus Ties, Flashpoint, 2019
27. Lazarus Arisen - ARCHITECTURE / TOOLS / ATTRIBUTION, Group-IB, 2017
30. Android Malware Appears Linked to Lazarus Cybercrime Group, McAfee, 2017
31. Dissecting Operation Troy: Cyberespionage in South Korea, McAfee, 2013
32. Lazarus Under The Hood, PDF, Securelist, 2017
33. Lazarus Arisen - ARCHITECTURE / TOOLS / ATTRIBUTION, Group-IB, 2017
34. Lazarus Group, MITRE ATT&CK
35. Lazarus Group, MITRE ATT&CK
36. Lazarus Continues Heists, Mounts Attacks on Financial Organizations in Latin America, TrendMicro, 2018
37. Malpedia
38. The devil’s in the Rich header, Securelist, 2018
39. APT37 (Reaper) - The Overlooked North Korean Actor, FireEye, 2018
The Lazarus Constellation

A study on North Korean malware

06/01/2020

LEXFO  avisa partners