★OPERATION★

BLOCKBUSTER

Unraveling the Long Thread of the Sony Attack

NOVETTA

NOVETTA

www.OperationBlockbuster.com

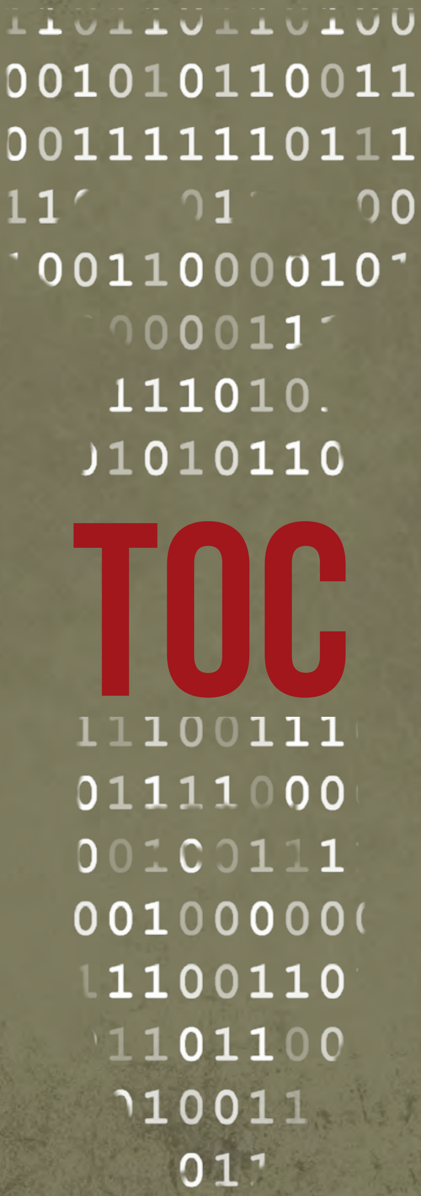


TABLE OF CONTENTS

CAVEATS

To the best of Novetta's knowledge and belief, participants in this effort did not disclose, access, or utilize any confidential information that would result in violation of any third party agreements including, but not limited to, non-disclosure agreements or customer agreements.

While this report discusses previous attribution claims made by outside parties, Novetta cannot definitively confirm any such attribution through the technical analysis detailed in this and other Operation Blockbuster reports.

Please note that this report includes terms that will not be familiar to everyone. We have included a glossary at the end of this report and denoted such defined terms with the superscript for your convenience.

CHAPTER

ONE

1. EXECUTIVE SUMMARY

Operation Blockbuster is a Novetta-led coalition of private industry partners, created with the intent to understand and potentially disrupt malicious tools and infrastructure that have been attributed to an adversary that Novetta has identified and named as the Lazarus Group. This group has been active since at least 2009, and potentially as early as 2007, and was responsible for the November 2014 destructive wiper attack against Sony Pictures Entertainment (SPE).

The attack against Sony Pictures Entertainment (SPE) was unprecedented in its media coverage and overt use of malicious destructive capabilities against a commercial entity. The SPE attack broke new ground not only as a destructive malware attack on a U.S. commercial entity but also due to the fact that the U.S. government attributed the attack to North Korea and enacted small reciprocal measures.[1] While the debate over who was responsible – North Korea, hacktivists, or SPE employees – was the primary subject played out in the media, the attack presented much larger implications, such as how little resistance a modern commercial enterprise is able to provide in the face of a capable and determined adversary with destructive intent.

1 "North Korea and the Sony Hack: Exporting Instability Through Cyberspace." Stephen Haggard, Jon R. Lindsay. Analysis from the East-West Center. May 2015. http://www.eastwestcenter.org/system/tdf/private/api117.pdf

Further, Novetta's analysis of the observed tooling and TTPs suggests that the group has executed numerous successful attacks due in large part to their organization and determination, more so than due to any highly sophisticated malware such as those reportedly used by similar classes of threat actors reported in the last few years, e.g., HDD malware[2] and Satellite Turla.[3]

Through careful analysis outlined in this report and other associated reverse engineering technical reports, Novetta has been able to link the malware used in the SPE attack to a widely varied malicious toolset. This toolset includes malware directly related to previously reported attacks, suggesting that these malicious tools have been actively developed and used over a span of at least 7 years, and that the attackers responsible for the SPE attack have a much larger collection of related malware outside of the set of reported SPE destructive malware. Due to this, we strongly believe that the SPE attack was not the work of insiders or hacktivists. Instead, given the malicious tools and previous cyber operations linked to these tools, it appears that the SPE attack was carried out by a single group, or potentially very closely linked groups sharing technical resources, infrastructure, and even tasking. We have dubbed this group the Lazarus Group. Although our analysis cannot support direct attribution of a nation-state or other specific group due to the difficulty of proper attribution in the cyber realm, the FBI's official attribution claims[4] could be supported by our findings.

While the SPE attack occurred over a year ago, we are releasing this report now to detail our technical findings, clarify details surrounding the SPE hack, and profile the Lazarus Group, who has continued to develop tools and target victims since then. Most importantly, Novetta continues to work with our public and private partner organizations in this Operation to ensure that Novetta's signatures and other data will have a meaningful impact on the Lazarus Group's abilities to function, as well as help potential victims understand in great detail not only the technical but also the operational methods. Novetta feels that this combination of sharing highly technical analysis with both the public and private industry is the best way to interdict these types of actors.

2 "NSA Planted Stuxnet-Type Malware Deep Within Hard Drive Firmware." The Hacker News. February 16, 2015. http://thehackernews.com/2015/02/hard-drive-firmware-hacking.html

3 "Satellite Turla: APT Command Control in the Sky." Securelist. September 9, 2015. https://securelist.com/blog/research/72081/satellite-turla-apt-command-and-control-in-the-sky/

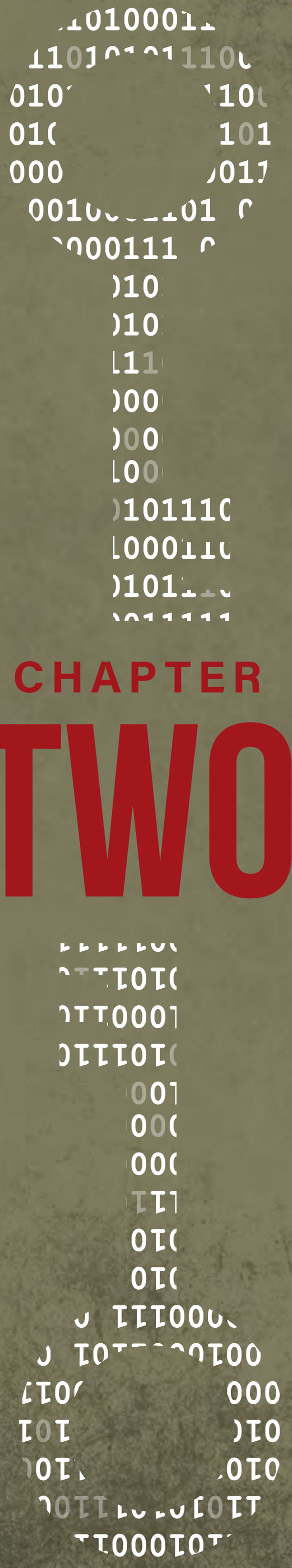
4 "Update on Sony Investigation." FBI. December 19, 2014. https://www.fbi.gov/news/pressrel/press-releases/update-on-sony-investigation

1.1 KEY TAKEAWAYS

1. The Lazarus Group is a well-established group that appears to be comprised of various sets of developers and operators for their custom malware.
2. The Lazarus Group demonstrates varying levels of technical aptitude and proficiency in computer network operations (CNO).
3. From a binary analysis perspective, this threat actor demonstrates a heavy reliance on shared code, techniques, and ideas from other previously developed Lazarus Group tool components as well as outside sources. Due to this, malware used in the November 2014 SPE attack can be linked to a much wider set of the Lazarus Group's malware that has been under active development since as early as 2009.
4. The malware analyzed in this Operation and attributed to the Lazarus Group has been used to target government, media, military, aerospace, financial, and critical infrastructure entities in a limited geographic area, primarily South Korea and the United States.
5. Because of the depth and scope of malware tools, structure of the analyzed code bases, TTP overlap with similar attacks, and long trail of activities attributed to the Lazarus Group, Novetta does not believe that the SPE attack was carried out by insiders or hacktivists, but rather by a more structured, resourced, and motivated organization.
6. **The set of malware uncovered and analyzed during this Operation, more than 45 unique families to date, consists of a wide variety of attack tools:**

RATS	INSTALLERS	SPREADERS	LOADERS	HARD DRIVE WIPERS
GENERAL TOOLS	UNINSTALLERS	PROXY	KEYLOGGER	DDOS BOT

7. The frequency and type of code sharing across malware families may suggest the same group of author(s) across families or extensive sharing of resources between closely linked groups
8. The Lazarus Group has also been observed to share cryptographic keys across malware families as well as general techniques observed in other unrelated malware families.



CHAPTER

TWO

2. OPERATION DETAILS

Operation Blockbuster began in December 2014, independent of any investigation conducted by law enforcement or Sony, with the intent to not only identify and impact the malicious tools and infrastructure used by the Lazarus Group, but also to clarify details surrounding the November 2014 SPE attack, which was the subject of widespread confusion. By investigating the malware linked to this attack, we have determined that the Lazarus Group has operated largely unfettered for nearly a decade, conducting cyber espionage, denial of service attacks, data theft, and destructive attacks.

Before discussing Novetta's hunting methods, it is important to note that the majority of our malware samples and other data were sourced from public sources such as VirusTotal. As a result, our samples are biased towards the footprint and usage of this service. We do have some partners who provided malware samples, representing commercial ecosystem protectors and maintainers. Here again, our visibility is limited to the visibility of these partners.

2.1 Hunting Method

On December 14, 2014, US-CERT released an alert[5] entitled "Targeted Destructive Malware." The alert described a set of malware families used by undefined attackers to compromise large network infrastructures and deploy hard drive wiping malware, RATs, and proxy Trojans. While the document did not specifically call out the Guardians of Peace (GOP)'s attack against SPE from the previous month, and only provided some basic YARA signatures and import hashes, members of the security community released specific hashes for the malware used within the SPE attack. From these hashes (MD5s listed below), a baseline of the Lazarus Group's malware capabilities was established.

- d1c27ee7ce18675974edf42d4eea25c6
- 760c35a80d758f032d02cf4db12d3e55
- e1864a55d5ccb76af4bf7a0ae16279ba
- 6467c6df4ba4526c7f7a7bc950bd47eb

Novetta Hunting Methodology:

Identify starting sample(s)
In this case, the starting samples were identified by the industry as being from the SPE attack

Begin analysis of samples
Attempt to identify unique components of the code base that can provide high confidence signatures

Write high confidence signatures
Signatures can help capture other samples that use the same or very similar code snippets identified in step #2

Run high confidence signatures against a large corpus of malware
This is more easily accomplished using Totem or similar elastic malware analysis or file triaging framework

Collect and verify the accuracy of results
By checking the signature match, samples can be verified to ensure that there are no false positives, or to refine high confidence signatures

Identify any divergence in samples
Such a divergence may communicate some structure change or change in capabilities, and in turn provide more information about a threat group's toolset, development activity, and capabilities

Write new high confidence
Signatures for those portions of code

Repeat steps 4-7 until done

5 US-CERT. "Alert (TA14-353A): Targeted Destructive Malware" https://www.us-cert.gov/ncas/alerts/TA14-353A December 14, 2014.

By analyzing the base set of malware associated with the Lazarus Group, Novetta determined that there were common code and libraries being used across multiple malware families (see Section 4 for more details).

From these common snippets of code and use of library functions, signatures were generated to detect additional malware samples using both open-source tools and Totem,[6] an open-source, Novetta-developed framework for large-scale file analysis and triage. While attempting to acquire all malware associated with a particular threat group is a Sisyphean task, given the active development of multiple various toolsets, Novetta was able to detect and analyze more than 45 distinct malware families that fall under the Lazarus Group's toolset. A thorough discussion on these families, organized by usage and intention, can be found in Novetta's supplemental reports.

In our investigation, we were able to scan signatures over hundreds of millions of samples we collected as well as using industry partners' AV scanning engines. The use of such a large corpus of malware allowed Novetta to fine-tune the signatures for shared code components to ensure a high reliability that the code fragments used for detection were specific to the Lazarus Group and not the result of commodity code. From the billions of files scanned, Novetta's signatures produced approximately 2000 samples, of which 1000 were manually vetted and catalogued as belonging to the Lazarus Group.

6 https://github.com/Novetta/totem

CHAPTER

THREE

3. LAZARUS GROUP DETAILS

By identifying the malware linked to the SPE attack (Section 2.1) and other related samples and capabilities, Novetta has been able to compile a picture of a group that has been active for nearly a decade. Based on analysis of the extensive malware set collected, as well as details found in public reporting from linked attacks, the Lazarus Group appears to have resources that allow for development of custom malware tools for extensive, targeted, and coordinated attacks, including long periods of reconnaissance. The Lazarus Group has also displayed the technical capability and will to perform destructive attacks against targets. The following sections detail the SPE attack and subsequent media reporting, the group's TTPs, targets based on known attacks and malware artifacts, and previous cyber campaigns that we have directly linked to the Lazarus Group.

3.1 The SPE Attack and Conflicting Attribution

In November 2014, Sony Pictures Entertainment (SPE) was attacked with destructive malware whose various components were publicly reported as Destover or Wiper and which Novetta identified in this Operation as WhiskeyAlfa, malware associated with the Lazarus Group threat actors (see Section 4.1 for details about the naming scheme used for malware attributed to the Lazarus Group). Publicly, a previously unknown hacker group named Guardians of Peace (GOP) took credit for the wiper attack and stolen data. The group eventually publicized the files stolen from SPE networks, including unreleased movies, usernames, passwords, and other IT details for internal SPE networks,[7] employees' personal information, payroll information, employee termination details, TV scripts, and company emails.

Following the attack, an initial FBI investigation concluded that the hack was the work of the North Korean government, as the malware used in the attack was linked to other malware attributed to North Korean actors – specifically, code snippets, encryption algorithms, data deletion methods, and compromised infrastructure used during the attack.[8] Infrastructure used in the SPE attack has previously been linked by the U.S. government directly to other identified North Korea cyber activity. Several security researchers also stated that the destructive attack could be linked to malware variants used in attacks that have been suggested to be the work of North Korea,[9] with similar TTPs as previous events attributed to North Korea,[10] [11] and shared infrastructure.[12]

However, others stated that the evidence for North Korean involvement is circumstantial.[13] For instance, while the infrastructure used in the SPE attack overlaps with infrastructure attributed to malicious cyber activity linked to North Korea, previously malicious IP addresses are not necessarily still used by the same attackers. In fact, the publicly reported C2 addresses were almost all public proxies used by a variety of malware operators in the past. Other reporting claimed that the SPE attack was the work of insiders rather than a nation-state,[14] and that the ability to thoroughly infiltrate the SPE network and steal sensitive data required insider knowledge. The data leaked included details of planned layoffs, suggesting a motivation for disgruntled employees to aid or provide stolen data to other attackers, such as piracy hacktivists targeting SPE. The attackers also dumped the stolen data, rather than keeping it secret as, some allege, a state power interested in intelligence or propaganda might do instead.[15] In contrast, previous destructive attacks against South Korean organizations in March 2013, which were linked to North Korea, involved no extortion demands from attackers. Notably, other public comments even doubted that North Korea had the capabilities to launch such an attack largely due

7 "Sony's IT blueprints leaked by hackers." CSO. December 4, 2014. http://www.csoonline.com/article/2855005/business-continuity/sonys-it-blueprints-leaked-by-hackers.html

8 "Update on Sony Investigation." FBI. December 19, 2014. https://www.fbi.gov/news/pressrel/press-releases/update-on-sony-investigation

9 "Destover: Destructive malware has links to attacks on South Korea." Symantec. December 4, 2014. http://www.symantec.com/connect/blogs/destover-destructive-malware-has-links-attacks-south-korea

10 "South Korean paper hit by major cyber attack." Phys.org. June 11, 2012. http://phys.org/news/2012-06-south-korean-paper-major-cyber.html

11 "Four-star spymaster behind North Korean hacking; Sony's 'The Interview' available online." The Washington Times. December 24, 2014. http://www.washingtontimes.com/news/2014/dec/24/inside-the-ring-four-star-spy-master-behind-north-k/

12 "Sony Hack Mirrors Attack on South Korean Newspaper, Researcher Says." The Wall Street Journal. December 19, 2014. http://blogs.wsj.com/korearealtime/2014/12/19/sony-hack-mirrors-attack-on-south-korean-newspaper-researcher-says/

13 "No, North Korea Didn't Hack Sony." The Daily Beast. December 24, 2014. http://www.thedailybeast.com/articles/2014/12/24/no-north-korea-didn-t-hack-sony.html

14 "Norse Investigation Focusing on a Small Group, Including Sony Ex-Employees." Norse. December 29, 2014. http://web.archive.org/web/20150623023623/http://darkmatters.norsecorp.com/2014/12/29/ex-employee-five-others-fingered-in-sony-hack/http://darkmatters.norsecorp.com/2014/12/29/ex-employee-five-others-fingered-in-sony-hack/

15 "No, North Korea Didn't Hack Sony." The Daily Beast. December 24, 2014. http://www.thedailybeast.com/articles/2014/12/24/no-north-korea-didn-t-hack-sony.html

to insufficient infrastructure,[16] or that other[17] nation-states[18] were involved.

In addition to the conflicting attribution of the attacks, some initial reporting suggested that the attack shared some links to Shamoon, the destructive malware that hit Saudi Aramco and other oil company networks in August 2012. This was based on the use of the same commercially available drivers (EldoS RawDisk) and attack techniques rather than any shared malware code.[19] From Novetta's analysis of Shamoon, there is no clear link between Shamoon and any destructive malware variants tracked in this operation that would indicate shared author(s). However, the author(s) behind the SPE destructive malware may have copied Shamoon's attack techniques, or vice versa. It is worth noting that the two nation-states publicly blamed for the Saudi Aramco and SPE attacks (Iran and North Korea, respectively) have had a technology sharing treaty since 2012, with a specific focus on cyber.[20]

While some critics of the SPE attribution do ask important questions, such as whether the use of public proxies or open-source code libraries is sufficient evidence for attribution, many who have written off any possible nation-state involvement due to GOP's public actions have not fully considered the possible motives of a state's interest in attacking SPE. Furthermore, to discount nation-states like North Korea as too underdeveloped ignores the demonstrated fact that cyber attacks are no longer limited to highly resourced nation-states.[21] [22] [23] The cyber footprint of not only governments and critical infrastructure, but also corporate enterprises, has grown significantly while still largely lacking in sophisticated security operations, effectively lowering the barrier to entry even further for threat groups.

Although Novetta is unable to determine via technical malware analysis whether or not the SPE attack was carried out by an identified nation-state, we have been able to link the malware used in this attack to a widely varied malicious toolset profiled in this Operation, including tools directly related to previously reported attacks (Section 3.4). This link to known attacks suggests that these malicious tools have been actively developed and used over a span of at least 7 years, and that the attackers responsible for the SPE attack have a much larger collection of related malware outside of the SPE destructive malware. Due to this finding, we strongly believe that the SPE attack was not the work of insiders or hacktivists. Furthermore, given the malicious tools and previous cyber operations linked to these tools, it appears that the SPE attack was carried out by a single group, or potentially very closely linked groups sharing technical resources, infrastructure, and even tasking. We have dubbed this organization the Lazarus Group. However, rather than focus on the specifics of attribution, this report and subsequent technical reports are intended to detail our technical findings on the scope of the Lazarus Group's known tools and capabilities.

> Due to this finding, we strongly believe that the SPE attack was <u>not</u> the work of insiders or hacktivists. Furthermore, given the malicious tools and previous cyber operations linked to these tools, it appears that the SPE attack was carried out by a single group, or potentially very closely linked groups sharing technical resources, infrastructure, and even tasking. We have dubbed this organization the LAZARUS GROUP.

16 "Former Anonymous hacker doubts North Korea behind Sony attack." CBS News. December 17, 2014. http://www.cbsnews.com/videos/former-anonymous-hacker-doubts-north-korea-behind-sony-attack/"Sony Hackers Guardians of Peace Troll FBI, Anonymous Convinced Hack Didn't Come From North Korea."

17 "A security firm claims it was Russia that hacked Sony – and that it still has access." Business Insider. February 5, 2015. http://www.businessinsider.com/a-security-firm-claims-it-was-russia-that-hacked-sony-and-that-they-still-have-access-2015-2

18 "Evidence in Sony hack attack suggests possible involvement by Iran, China or Russia, intel source says." Fox News. December 19, 2014. http://www.foxnews.com/politics/2014/12/19/fbi-points-digital-finger-at-north-korea-for-sony-hacking-attack-formal.html

19 "Sony Pictures malware tied to Seoul, 'Shamoon' cyber-attacks." Ars Technica. December 4, 2014. http://arstechnica.com/security/2014/12/sony-pictures-malware-tied-to-seoul-shamoon-cyber-attacks/

20 "Iran and North Korea sign technology treaty to combat hostile malware." V3. September 3, 2012. http://www.v3.co.uk/v3-uk/news/2202493/iran-and-north-korea-sign-technology-treaty-to-combat-hostile-malware

21 "Profiling an enigma: The mystery of North Korea's cyber threat landscape." HP Security Research. August 2014. http://community.hpe.com/hpeb/attachments/hpeb/off-by-on-software-security-blog/388/2/HPSR%20SecurityBriefing_Episode16_NorthKorea.pdf

22 "Operation Cleaver." Cylance. December 2014. http://cdn2.hubspot.net/hubfs/270968/assets/Cleaver/Cylance_Operation_Cleaver_Report.pdf

23 "Malware-based Attacks Against POS Systems." Infosec Institute. February 11, 2014. http://resources.infosecinstitute.com/malware-based-attacks-pos-systems/

3.2 Tactics, Techniques, and Procedures (TTPs)

The Lazarus Group has developed an extensive and varied toolset which effectively combines a number of methods for delivering additional malicious tools, exfiltrating data, and launching destructive attacks. While the group's combined capabilities are not necessarily as polished or advanced as other publicly reported APT groups, the TTPs and malware connected to the Lazarus Group demonstrate that it is a capable and determined adversary. Particularly when considering the state of most, if not all, organizations who struggle with the complexity of computer network defense, it is clear that the Lazarus Group is taking advantage of a cyber attacker's asymmetric advantage in these scenarios. The generally lax defensive capabilities of their targeted organizations are reflected by the structure and complexity of their tooling and how they use these tools operationally - the Lazarus Group's tools are sufficiently advanced for the intended targets and level of impact. This is also typically seen in most malware tooling discovered and reported on, from the more advanced and complex malware frameworks like Flame[24] and Satellite Turla,[25] both observed targeting a narrow, hardened set of victims, to the off-the-shelf and simple malware (Plugx, Poison Ivy, etc) often used for softer targets or for initial access to target networks. Some threat groups make use of a full spectrum of malware, as was observed in Novetta's previous Operation SMN reporting,[26] where the Axiom group leveraged different tools and techniques dependent on the security posture and capabilities of target organizations. Compared to Axiom, Novetta's analysis of the Lazarus group's toolsets did not demonstrate the same widespread distribution between advanced, moderately advanced, and basic capabilities. Yet this clearly was not an impediment to the operators in the Lazarus Group, given the success of their attacks.

Despite evidence suggesting that their attacks to date have succeeded without the need for some of the more advanced techniques or capabilities, the Lazarus Group has shown creativity in their operations that set them apart.

Despite evidence suggesting that their attacks to date have succeeded without the need for some of the more advanced techniques or capabilities, the Lazarus Group has shown creativity in their operations that set them apart. For example, the group has several malware variants with TLS mimicking capabilities (Section 4.3.3.1) to evade network detection, as well as a P2P malware family that serves as a platform for an operator to access all infected instances. The Lazarus Group has also used master boot record (MBR) wiper malware since at least 2009, marking some of the earliest known instances of targeted destructive malware. Furthermore, the willingness to use destructive malware in such a wide scope, seen with the SPE attack as well as other linked attacks, distinguishes them from many other APT groups. However, the Lazarus Group is not limited solely to the deployment of destructive malware. In fact, the toolset identified during this Operation suggests that the Lazarus Group encompasses a wide spectrum of CNO capabilities, including distributed denial of service (DDoS) malware, keyloggers, and RATs, and even a P2P malware family that allows operators to establish a common program base and remote administration across all infected machines.

24 "Meet 'Flame,' the massive spy malware infiltrating Iranian computers." Wired. May 28, 2012. DNS-Calc APT Trojan Uses DNS Queries to Generate C&C Port Number

25 "Satellite Turla: APT Command Control in the Sky." Securelist. September 9, 2015. https://securelist.com/blog/research/72081/satellite-turla-apt-command-and-control-in-the-sky/

26 "Operation SMN: Axiom Threat Actor Group Report." Novetta. November 2014. http://www.novetta.com/wp-content/uploads/2014/11/Executive_Summary-Final_1.pdf

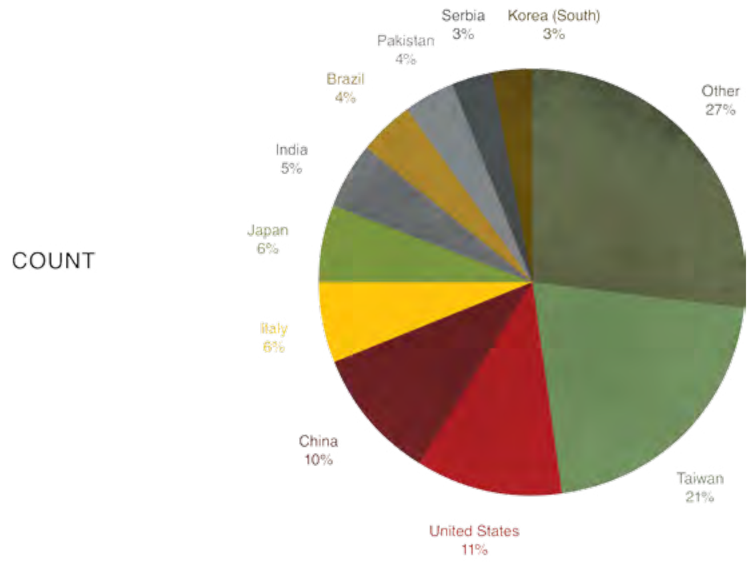
Among the TTPs we have seen, based on the identified malware corpus and linked cyber campaigns tied to the Lazarus Group, including SPE, the **Lazarus Group's primary TTPs are:**

DDoS malware	Espionage campaigns marked by a long initial reconnaissance period of targeted networks, including malware customized specifically for target networks
Destructive malware	Compromised IPs and websites as command-and-control (C2)
Extensive use of various types of obscure encryption	Proxies to mask true C2
Integration of publicly available tools, libraries, and other code	Email as C2
Re-use of malicious code across multiple malware families	Mimicking TLS as a means of network detection evasion
Multiple attack components/vectors	
Spear phishing	
Targeting of South Korean AV and indigenous Korean software	
Use of other legitimate software to gain access to victim networks	
Decoy documents	

3.3 Targeting

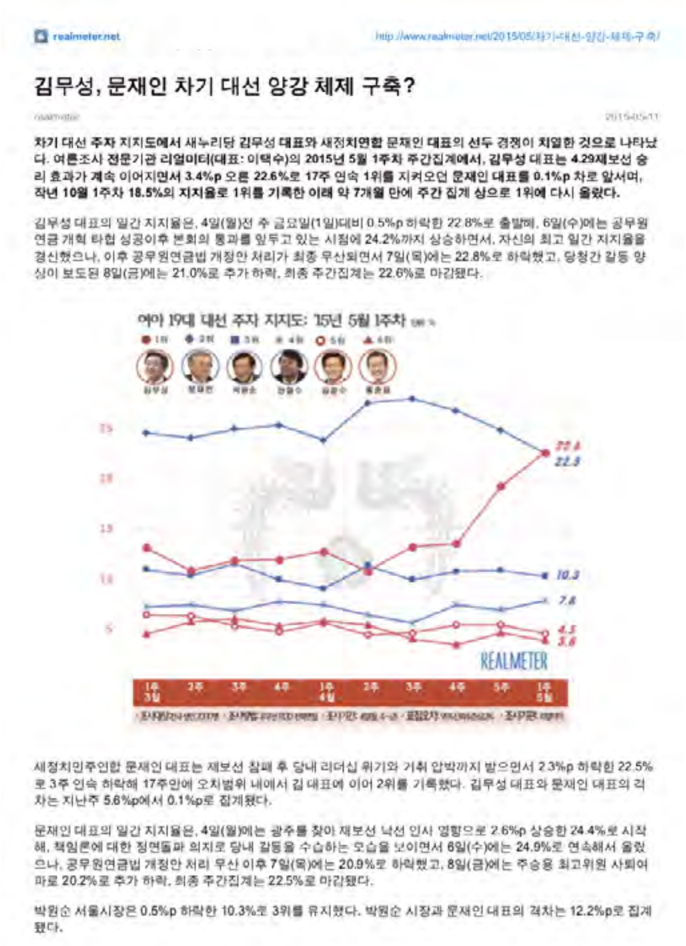
The Lazarus Group has targeted a number of industry verticals over the years, including government, military, financial, media and entertainment, and critical infrastructure.

According to previous public research and reporting, the Lazarus Group has targeted a number of industry verticals over the years, including government, military, financial, media and entertainment, and critical infrastructure. These victims have largely been limited to South Korea and the United States. Based on three months of telemetry gathered from initial signatures created and shared with industry partners, however, possible infections were found in a much wider geographic area, including concentrations of detected Lazarus Group malware found in other Asian countries like Taiwan, China, Japan, and India. While these initial signature detections provide a general overview of some possible malicious activity, these numbers should not be considered reflective of the totality of Lazarus Group tools detected in this Operation, due to the nature of our approach in this effort and our partners' visibility into these geographic areas.



Several recent examples of targeting were observed in spear-phishing documents dropped by samples of an installer developed by the Lazarus Group, which Novetta has named IndiaAlfa.[27]

27 <reference external report on this>



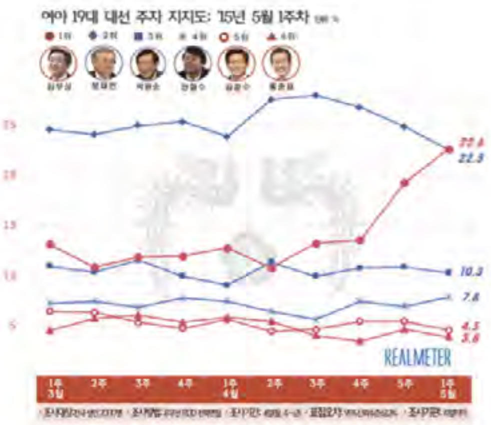
realmeter.net http://www.realmeter.net/2015/05/차기-대선-양강-체제-구축/

김무성, 문재인 차기 대선 양강 체제 구축?

realmeter 2015-05-11

차기 대선 주자 지지도에서 새누리당 김무성 대표와 새정치연합 문재인 대표의 선두 경쟁이 치열한 것으로 나타났다. 여론조사 전문기관 리얼미터(대표: 이택수)의 2015년 5월 1주차 주간집계에서, 김무성 대표는 4.29재보선 승리 효과가 계속 이어지면서 3.4%p 오른 22.6%로 17주 연속 1위를 지켜오던 문재인 대표를 0.1%p 차로 앞서며, 작년 10월 1주차 18.5%의 지지율로 1위를 기록한 이래 약 7개월 만에 주간 집계 상으로 1위에 다시 올랐다.

김무성 대표의 일간 지지율은, 4일(월)전 주 금요일(1일)대비 0.5%p 하락한 22.8%로 출발해, 6일(수)에는 공무원연금 개혁 타협 성공이후 본회의 통과를 앞두고 있는 시점에 24.2%까지 상승하면서, 자신의 최고 일간 지지율을 경신했으나, 이후 공무원연금법 개정안 처리가 최종 무산되면서 7일(목)에는 22.8%로 하락했고, 당청간 갈등 양상이 보도된 8일(금)에는 21.0%로 추가 하락, 최종 주간집계는 22.6%로 마감됐다.



새정치민주연합 문재인 대표는 재보선 참패 후 당내 리더십 위기와 거취 압박까지 받으면서 2.3%p 하락한 22.5%로 3주 연속 하락해 17주만에 오차범위 내에서 김 대표에 이어 2위를 기록했다. 김무성 대표와 문재인 대표의 격차는 지난주 5.6%p에서 0.1%p로 집계됐다.

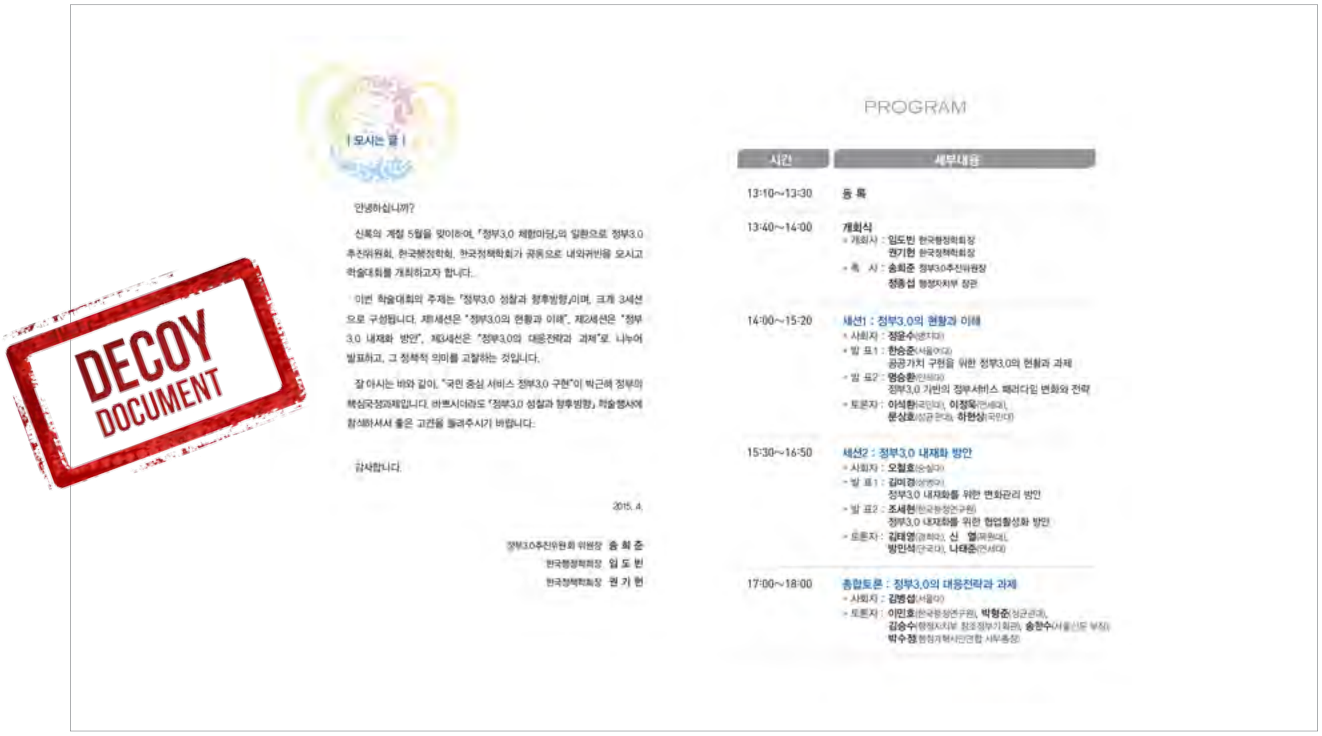
문재인 대표의 일간 지지율은, 4일(월)에는 광주를 찾아 재보선 낙선 인사 영향으로 2.6%p 상승한 24.4%로 시작해, 책임론에 대한 정면돌파 의지로 당내 갈등을 수습하는 모습을 보이면서 6일(수)에는 24.9%로 연속해서 올랐으나, 공무원연금법 개정안 처리 무산 이후 7일(목)에는 20.9%로 하락했고, 8일(금)에는 주승용 최고위원 사퇴여파로 20.2%로 추가 하락, 최종 주간집계는 22.5%로 마감됐다.

박원순 서울시장은 0.5%p 하락한 10.3%로 3위를 유지했다. 박원순 시장과 문재인 대표의 격차는 12.2%p로 집계됐다.

DECOY DOCUMENT

Figure [3-1]: Decoy document dropped by IndiaAlfa variant relating to the May 2015 parliamentary election in South Korea

The above example is a media report discussing the May 2015 South Korean parliamentary election, which included candidates for the Saenuri Party, South Korea's ruling party since 2008. Interestingly, Saenuri has taken a much stronger stance toward North Korea aggressions in comparison to the pre-2008 "Sunshine Policy" which actively sought cooperation between the two states. Saenuri actively supports the North Korean Human Rights Law and founded Open Radio for North Korea, an organization which spreads information about democracy. Saenuri is also a major advocate of cyber security and the National Intelligence Service. Despite being amidst corruption allegations, the Saenuri Party won three of the four parliamentary seats during the election.


DECOY
DOCUMENT

| 모시는 글 |

안녕하십니까?

신록의 계절 5월을 맞이하여, 『정부3.0 체험마당』의 일환으로 정부3.0 추진위원회, 한국행정학회, 한국정책학회가 공동으로 내외귀빈을 모시고 학술대회를 개최하고자 합니다.

이번 학술대회의 주제는 『정부3.0 성찰과 향후방향』이며, 크게 3세션으로 구성됩니다. 제1세션은 "정부3.0의 현황과 이해", 제2세션은 "정부3.0 내재화 방안", 제3세션은 "정부3.0의 대응전략과 과제"로 나누어 발표하고, 그 정책적 의미를 고찰하는 것입니다.

잘 아시는 바와 같이, "국민 중심 서비스 정부3.0 구현"이 박근혜 정부의 핵심국정과제입니다. 바쁘시더라도 『정부3.0 성찰과 향후방향』 학술행사에 참석하셔서 좋은 고견을 들려주시기 바랍니다.

감사합니다.

2015. 4.

정부3.0추진위원회 위원장 송 희 준
한국행정학회장 임 도 빈
한국정책학회장 권 기 헌

PROGRAM

시간	세부내용
13:10~13:30	등 록
13:40~14:00	**개회식** • 개회사 : **임도빈** 한국행정학회장 **권기헌** 한국정책학회장 • 축 사 : **송희준** 정부3.0추진위원장 **정종섭** 행정자치부 장관
14:00~15:20	**세션1 : 정부3.0의 현황과 이해** • 사회자 : **정윤수**(명지대) • 발 표1 : **한승준**(서울여대) 공공가치 구현을 위한 정부3.0의 현황과 과제 • 발 표2 : **명승환**(인하대) 정부3.0 기반의 정부서비스 패러다임 변화와 전략 • 토론자 : **이석환**(국민대), **이정욱**(연세대), **문상호**(성균관대), **하현상**(국민대)
15:30~16:50	**세션2 : 정부3.0 내재화 방안** • 사회자 : **오철호**(숭실대) • 발 표1 : **김미경**(상명대) 정부3.0 내재화를 위한 변화관리 방안 • 발 표2 : **조세현**(한국행정연구원) 정부3.0 내재화를 위한 협업활성화 방안 • 토론자 : **김태영**(경희대), **신 열**(목원대), **방민석**(단국대), **나태준**(연세대)
17:00~18:00	**종합토론 : 정부3.0의 대응전략과 과제** • 사회자 : **김병섭**(서울대) • 토론자 : **이민호**(한국행정연구원), **박형준**(성균관대), **김승수**(행정자치부 창조정부기획관), **송한수**(서울신문 부장) **박수정**(행정개혁시민연합 사무총장)

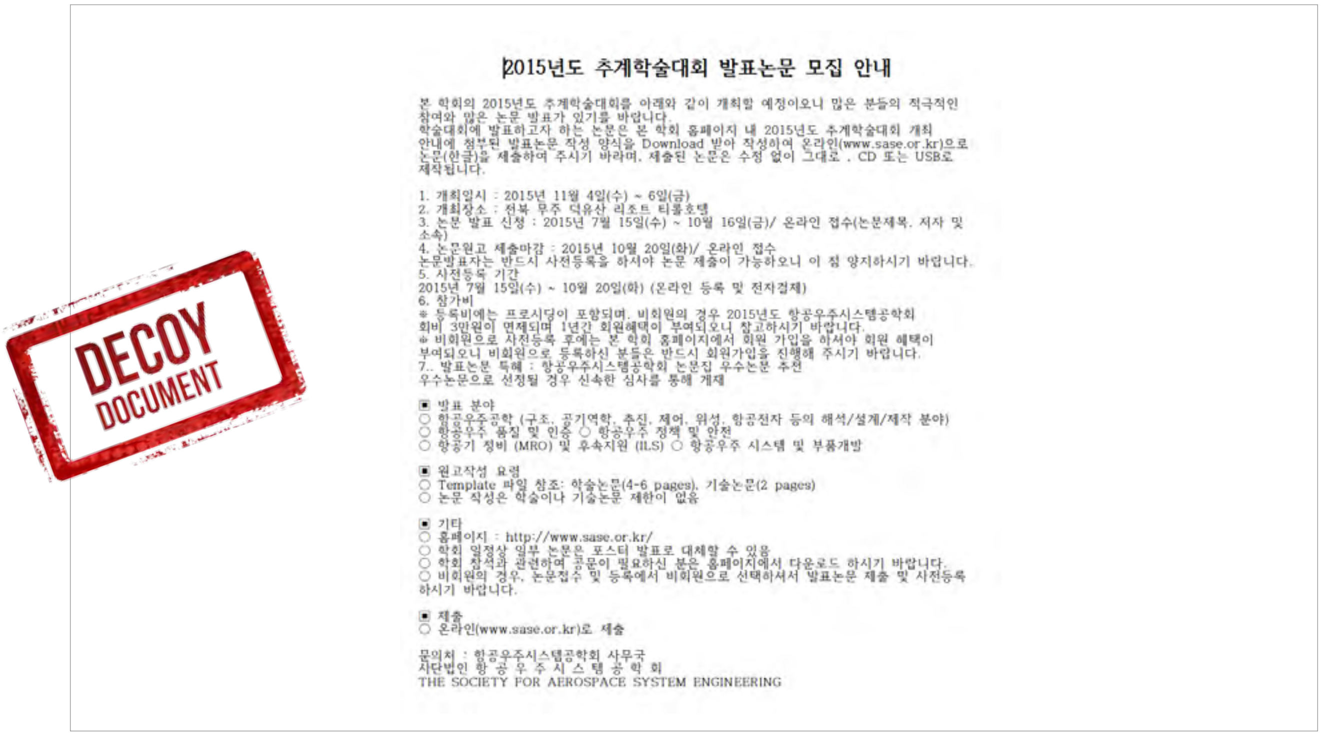
Figure [3-2]: Decoy document from April 2015 dropped by an IndiaAlfa variant about the Government 3.0 conference in May

Another document dropped by India Alfa includes information about the Government 3.0 Conference, held in May 2015. South Korea's Government 3.0 emphasizes transparency and collaboration. Of note is the program's 24-hour online portal service which connects citizens to multiple central and local government agencies.

More recently, a variant compiled in October 2015 contains a decoy document asking speakers at the Society for Aerospace System Engineering's (SASE) 2015 autumn conference to register their papers. A warning that same month warned users not to click on these SASE documents, as it exploits a vulnerability (CVE-2015-6585) in the Hangul Word Processor (HWP) to deliver a malicious payload.[28] This same vulnerability, patched in September 2015, was reportedly exploited in zero-day attacks tied by researchers to North Korean threat actors.[29]

28 "[Warning] Do not open an E-mail that includes a document titled '2015년도 추계학술대회 안내문.hwp' (2015 Fall Conference Announcement)." Division of Information Security, Seoul National University. October 20, 2015. http://community.snu.ac.kr/bbs/bbs.enmessage.view.screen?bbs_id=403&message_id=157326&search_field=title&search_word=&classified_value=

29 "Hangul Word Processor (HWP) Zero-Day." FireEye. September 9, 2015. https://www.fireeye.com/content/dam/fireeye-www/global/en/blog/threat-research/FireEye_HWP_ZeroDay.pdf


DECOY
DOCUMENT

2015년도 추계학술대회 발표논문 모집 안내

본 학회의 2015년도 추계학술대회를 아래와 같이 개최할 예정이오니 많은 분들의 적극적인 참여와 많은 논문 발표가 있기를 바랍니다.
학술대회에 발표하고자 하는 논문은 본 학회 홈페이지 내 2015년도 추계학술대회 개최 안내에 첨부된 발표논문 작성 양식을 Download 받아 작성하여 온라인(www.sase.or.kr)으로 논문(한글)을 제출하여 주시기 바라며, 제출된 논문은 수정 없이 그대로 . CD 또는 USB로 제작됩니다.

1. 개최일시 : 2015년 11월 4일(수) ~ 6일(금)
2. 개최장소 : 전북 무주 덕유산 리조트 티롤호텔
3. 논문 발표 신청 : 2015년 7월 15일(수) ~ 10월 16일(금)/ 온라인 접수(논문제목, 저자 및 소속)
4. 논문원고 제출마감 : 2015년 10월 20일(화)/ 온라인 접수
논문발표자는 반드시 사전등록을 하셔야 논문 제출이 가능하오니 이 점 양지하시기 바랍니다.
5. 사전등록 기간
2015년 7월 15일(수) ~ 10월 20일(화) (온라인 등록 및 전자결제)
6. 참가비
※ 등록비에는 프로시딩이 포함되며, 비회원의 경우 2015년도 항공우주시스템공학회 회비 3만원이 면제되며 1년간 회원혜택이 부여되오니 참고하시기 바랍니다.
※ 비회원으로 사전등록 후에는 본 학회 홈페이지에서 회원 가입을 하셔야 회원 혜택이 부여되오니 비회원으로 등록하신 분들은 반드시 회원가입을 진행해 주시기 바랍니다.
7.. 발표논문 특혜 : 항공우주시스템공학회 논문집 우수논문 추천
우수논문으로 선정될 경우 신속한 심사를 통해 게재

▣ 발표 분야
○ 항공우주공학 (구조, 공기역학, 추진, 제어, 위성, 항공전자 등의 해석/설계/제작 분야)
○ 항공우주 품질 및 인증 ○ 항공우주 정책 및 안전
○ 항공기 정비 (MRO) 및 후속지원 (ILS) ○ 항공우주 시스템 및 부품개발

▣ 원고작성 요령
○ Template 파일 참조: 학술논문(4-6 pages), 기술논문(2 pages)
○ 논문 작성은 학술이나 기술논문 제한이 없음

▣ 기타
○ 홈페이지 : http://www.sase.or.kr/
○ 학회 일정상 일부 논문은 포스터 발표로 대체할 수 있음
○ 학회 참석과 관련하여 공문이 필요하신 분은 홈페이지에서 다운로드 하시기 바랍니다.
○ 비회원의 경우, 논문접수 및 등록에서 비회원으로 선택하셔서 발표논문 제출 및 사전등록 하시기 바랍니다.

▣ 제출
○ 온라인(www.sase.or.kr)로 제출

문의처 : 항공우주시스템공학회 사무국
사단법인 항 공 우 주 시 스 템 공 학 회
THE SOCIETY FOR AEROSPACE SYSTEM ENGINEERING

Figure [3-3]: Document dropped by an IndiaAlfa sample asking speakers to register papers for the upcoming Society for Aerospace System Engineering (SASE) conference

The above decoy document is a .hwp file, meant to be used with Hangul Word Processor (HWP), an indigenous South Korean word processing software. Other IndiaAlfa samples have also been observed dropping other decoy documents for HWP, such as a Korean-language resume and a directory for the Saejong Institute's National Strategy Training Courses, the latter of which was identified in an article referencing North Korean spear-phishing strategies.[30] In fact, HWP appears to be a popular attack vector for targeting South Korean victims,[31] [32] which may be due to the fact that 80% of documents attached to South Korean government and public agencies' websites are reportedly HWP documents.[33]

Based on the analysis of malware identified in this Operation and tied to the Lazarus Group based on code reuse, as well as the public reporting of events that we have linked to the Lazarus Group's activity, we believe that this threat group has targeted a wide variety of victims, in addition to the SPE attack.

30 "공공기관등 주의 촉구…특정 사용자 대상 '정밀공격' 피해 가능성 높아 (Public institutions urged to use caution…high likelihood of precision attacks targeting specific users)." 매일 경제 (Daily News). May 10, 2015. http://news.mk.co.kr/newsRead.php?year=2015&no=444993

31 "한글 파일 제로 데이(Zero-Day) 취약점 악용 공격 (Attacks exploit Hangul file Zero-Day vulnerabilities)" AhnLab. January 29, 2013. http://asec.ahnlab.com/902

32 "한글 제로데이 취약점을 이용한 악성코드 (Malware exploits Hangul Zero-Day vulnerabilities) ." AhnLab. May 20, 2015. http://asec.ahnlab.com/1035

33 "北, 한글 제로데이 공격 시도…정부 '비밀문서' 노렸나 (North Korea, Hangul Zero-Day attack attempt…were government "secret documents" revealed)?" Focus news. September 11, 2015. http://www.focus.kr/view.php?key=2015091100120249472

3.4 Links to Previous Reporting

Some of the malware variants identified during Operation Blockbuster have been correlated to previously reported incidents and attacks, either because the malware was specifically identified in the attack, the Lazarus Group malware shared notable code overlap with the publicly reported malware, or the C2 infrastructure publicly reported was also found hard coded in malicious tools used by the Lazarus Group. Additionally, several events also had TTPs highly similar to those of the Lazarus Group and have been linked to other notable attacks by security researchers. While some of these indicators, such as overlapping C2s or some TTPs, may not be definitive proof of a linked activity, the collective picture of these events together provide a stronger link.

These ties strongly suggest that the Lazarus Group has been active since at least 2009, and potentially as far back as 2007, or has extensively shared resources with other closely linked groups responsible for these attacks. In the scenario that the GOP were a real organization and responsible for the SPE attack, this would suggest that SPE was not the only operation by the hacktivist group. However, Novetta's analysis and findings suggest that the SPE attack was one of several attacks attributable to the Lazarus Group, who may have posed as the pop up hacktivist collective to mislead or distract the public.

The Lazarus Group Timeline

MARCH 7, 2007:

Development of first generation malware used in "Operation Flame," activity that is eventually tied to "Operation 1Mission," "Operation Troy," and the DarkSeoul 2013 attacks.

JULY 4, 2009:

A large-scale DDoS attack on US and South Korean websites uses the MYDOOM and Dozer malware, which is suspected to have arrived in email messages. The malware places the text "Memory of Independence Day" in the Master Boot Record (MBR).

2009 – 2013:

Operation Troy cyber espionage campaign is active for several years, culminating in the March 2013 DarkSeoul attacks.

MARCH 2011:

"Ten Days of Rain" attack targets South Korean media, financial, and critical infrastructure targets. Compromised computers within South Korea are used to launch DDoS attacks.

APRIL 2011:

DDoS attack targets Nonghyup Bank.

2012:

"Operation 1Mission" campaign, also linked to the March 2013 DarkSeoul attacks, begins. Attackers behind this activity have reportedly been active since 2007.

JUNE 2012:

Conservative South Korean newspaper claims to have been attacked unsuccessfully with wiper malware. Website is defaced by an unknown hacker group, "IsOne."

MARCH 20, 2013:

DarkSeoul wiper attack targets three South Korea broadcast companies, financial institutes, and one ISP. Two unknown groups take credit: NewRomanic Cyber Army Team and WhoIs Team.

MARCH 2014:

A hacking attempt to steal South Korean military data reportedly uses a server also seen in the March 2013 DarkSeoul attack.[43,44] Due to a lack of publicly available information on the C2 details, Novetta was unable to verify whether or not this attack was related.

NOVEMBER 24, 2014:

SPE networks are attacked with destructive malware. Information stolen from the company's networks is distributed online by previously unknown hacker group Guardians of Peace (GOP).

Various security researchers have connected multi-staged attacks over a period of several years, largely against South Korean targets. Attack methods used include hard disk wiping and DDoS attacks that triggered on historically significant dates, overwriting disk content with political strings, using legitimate third-party update mechanisms to move across target networks, specific encryption and obfuscation methods, and using similar C2 structures across campaigns. We have been able to directly link several of these attacks to the Lazarus Group.

Operation Flame and Operation 1Mission: 2007 – 2012

IssueMakersLab researchers have connected malicious activity as recent as the March 2013 DarkSeoul wiper attack to activity as far back as 2007,[36] as the attackers used the same passwords, RSA encryption keys, and C2 protocol across attacks.[37] Since 2012, these attackers have reportedly carried out activities under the name "Operation 1Mission," based on a PDB path found in a plurality of the malware linked to identified attack activity.

The group behind Operation 1Mission used legitimate third-party software (an ActiveX vulnerability) as an initial infection vector, shared public RSA key across malware variants for six years, exfiltrated data and downloaded additional malware using Stage 1 C2 servers using the same primary C2 protocol and C2 code, and distributed destructive malware via Stage 2 C2 servers using altered antivirus update files. The Operation 1Mission TTPs have been reflected in multiple reported events listed in this section as well as in the Lazarus Group's malware: although we cannot confirm a link to the malware used in Operation 1Mission, Novetta has also observed shared public RSA keys across malware families, shared C2 infrastructure between unrelated families, and Stage 1 C2 servers used to distribute and download additional malware tools.

IssueMakerLabs' analysis linking DarkSeoul to malicious activity from 2007 has also been supported by Fortinet research, which connected cyber activity from 2007, dubbed Operation Flame,[38] to Operation 1Mission, Operation Troy, and the DarkSeoul attack. While the earliest compilation date for Lazarus Group malware identified by Novetta during this Operation is 2009, Novetta has directly linked Lazarus Group tools to Operation Troy and at least two other attacks that

34 "South Korea Detects Suspected North Korea Hacking Attempt." Security Week. March 27, 2014. http://www.securityweek.com/south-korea-detects-suspected-north-korea-hacking-attempt

35 "S. Korean military research agency kept mum about hacking." The Dong-A Ilbo. April 11, 2014. http://english.donga.com/List/3/all/26/408162/1

36 "South Korea identified who's behind the cyber attack." IssueMakersLabs. https://docs.google.com/file/d/0B6CK-ZBGuMe4dGVHdTZnenJMRUk/edit?pli=1

37 "[단독] 3.20 사이버테러 공격주체, 그 실체 드러나다(The 3.20 cyber terrorism subject, the realities emerge)!" boannews.com April 9, 2013. http://www.boannews.com/media/view.asp?idx=35578

38 "Z:\Make Troy\, Not War: Case Study of the Wiper APT in Korea, and Beyond." Fortinet. 2014. https://www.blackhat.com/docs/asia-14/materials/Yang/Asia-14-Yang-Z-Make-Troy-Not-War-Case-Study-Of-The-Wiper-APT-In-Korea-And-Beyond.pdf

have been connected by researchers to the DarkSeoul attack (discussed below). Based on IssueMakersLab's and Fortinet's analyses, this could suggest that the Lazarus Group has been actively developing malware and conducting attacks since as early as 2007, or that they have links to another group active since that time.

Operation Troy: 2009 – 2012

Several of the malware variants collected and analyzed during Operation Blockbuster were reportedly used in the cyber-espionage campaign Operation Troy, active from 2009 to 2012. This campaign has been connected not only to the March 2011 "Ten Days of Rain" attacks but also to the widely reported March 2013 DarkSeoul attack on South Korean broadcasters and financial institutions.[39] The DarkSeoul wiper malware was said to have been uploaded to networks using prior access from Operation Troy's long reconnaissance and data exfiltration campaign.[40] The various malware tools used in Operation Troy were linked together by researchers based on shared code, and several of the malware hashes associated with Operation Troy also matched YARA signatures and known malware hashes for several Lazarus Group tools: DeltaAlfa, IndiaJuliett, IndiaGolf, IndiaHotel, LimaDelta, TangoBravo, and WhiskeyBravo (see Section 4.1 for details about the naming scheme used for malware attributed to the Lazarus Group).

Ten Days of Rain: March 2011

The March 2011 "Ten Days of Rain" attacks were a prolific series of DDoS attacks that targeted South Korean government, military, financial, and corporate organizations as well as U.S. military entities.[41] [42] The attack used the destructive malware payload identified by Novetta in this operation as WhiskeyBravo, as well as the DDoS malware DeltaAlfa, which was also later tied by researchers to the Operation Troy campaign. Additionally, an IP address embedded in another malware tool uncovered during the investigation into the Lazarus Group, a variant of SierraJuliett, was used as a first tier C2 server in these attacks.

The "Ten Days of Rain" attacks also bore many similarities to the July 2009 DDoS attacks against U.S. and Korean sites.[43] [44] [45] Notably, one sample of malware identified in the 2009 attacks includes a suicide script (Section 4.3.4) containing strings that appear to match the suicide script seen with KiloAlfa, a keylogger linked to the Lazarus Group's malware corpus during this operation. This would suggest that malware code widely used by the Lazarus Group can be linked via code reuse to publicly reported attacks as far back as 2009.

Other attacks on South Korean targets appear to share the same TTPs and infrastructure attributed to the above attacks, such as a June 2012 attack on conservative media organization JoongAng. An investigation into the attack by South Korean officials found that the attackers used two North Korean servers and 17 servers in 10 other countries. One of the servers used in the attack on JoongAng was also used in the March 2011 "Ten Days of Rain" attacks as well as the April 2011 Nonghyup Bank attack.[46] The JoongAng attack was claimed by the previously unknown hacking group IsOne.[47] Like GOP, IsOne emerged from complete obscurity and has done nothing since. The attack used destructive malware that reportedly affected databases and the newspaper editing system. Additionally, the JoongAng Ilbo website was defaced. The attack followed threats made the previous week by North Korea in response to reporting by South Korean media, though this does not necessarily suggest a motive for the attacker(s).

39 "2013年3月に┌生した韓┌へのサイバ┌攻┌をまとめてみた。 (I tried to summarize the cyber attacks on South Korea in March 2013)" piyolog. March 23, 2013. http://d.hatena.ne.jp/Kango/20130323/1363986809

40 "Dissecting Operation Troy: Cyberespionage in South Korea." McAfee. 2013. http://www.mcafee.com/us/resources/white-papers/wp-dissecting-operation-troy.pdf

41 "Ten Days of Rain: Expert analysis of distributed denial-of-service attacks targeting South Korea." McAfee. 2011. http://www.mcafee.com/us/resources/white-papers/wp-10-days-of-rain.pdf

42 "Check your zombie device! Analysis of the DDoS cyber terrorism against the country and future attacks on various devices." DongJoo Ha, SangMyung Choi, TaeHyung Kim, SeungYoun Han. Presentation at Black Hat Abu Dhabi, 2011. https://media.blackhat.com/bh-ad-11/Ha/bh-ad-11-Ha-Check_Your_Zombie_Devices_Slides.pdf

43 "MYDOOM Code Re-Used in DDoS on U.S. and South Korean Sites." Trend Micro. July 9, 2009. http://blog.trendmicro.com/trendlabs-security-intelligence/mydoom-code-re-used-in-ddos-on-u-s-and-south-korean-sites/

44 "McAfee Fingers North Korea in Attacks on South Korean Sites." Threatpost. July 6, 2011. https://threatpost.com/mcafee-fingers-north-korea-attacks-south-korean-sites-070611

45 "DDOS Madness Continued..." FireEye. July 11, 2009. https://www.fireeye.com/blog/threat-research/2009/07/ddos-madness-climax.html

46 "North behind hacking attack on JoongAng Ilbo." JoongAng Ilbo. January 17, 2013. http://koreajoongangdaily.joins.com/news/article/article.aspx?aid=2965629

47 "South Korean paper hit by major cyber attack." Phys.org. June 11, 2012. http://phys.org/news/2012-06-south-korean-paper-major-cyber.html

DarkSeoul: March 2013

Novetta has not found any definitive links between the publicly reported Jokra/DarkSeoul malware samples used in the March 2013 DarkSeoul attack and identified Lazarus Group malware. However, the attack has been linked to Operation Troy, as discussed above, as well as the Ten Days of Rain attacks,[48] both of which have direct links to the Lazarus Group's malware toolkit. In addition, it is also worth noting that, as with the SPE attack where a previously unknown hacktivist group took credit, the DarkSeoul attack was claimed by two previously unknown groups: the New Romanic Cyber Army Team and the WhoIs Hacking Team.

The same group behind the March 2013 DarkSeoul attack has also been linked to multiple other attacks over a period of four years, including the July 2009 DDoS attack whose malware shares suicide strings with KiloAlfa,[49] a May 2013 attack on South Korean financial institutions, a June 2013 Castov malware attacks on South Korean websites[50] and two DNS servers,[51] and a December 2014 MBR wiper attack on a South Korean power plant.[52] In the case of the June 2013 attacks, the attack reportedly took 6 months to plan,[53] during which attackers hacked file-sharing sites, again suggesting an extensive planning period prior to the ultimate attacks. Using compromised file-sharing sites is a tactic that has been observed in an older Lazarus Group malware family from 2011, LimaDelta. However, due to a lack of publicly available hashes, Novetta has not been able to analyze these events for any direct links to the Lazarus Group's code.

Based on our hunting method, starting with only a few of the samples publicly linked to the November 2014 SPE attack, Novetta was able to connect attacks since as early as 2009 to shared malware code we have associated with the Lazarus Group. Work by other security researchers has linked this activity as far back as 2007. These linked cyber operations over several years, including the SPE attack, suggest actions of a single group, or perhaps very close groups with similar goals who share tools, methods, taskings, and even operational duties. The span and destructive damage accomplished by these attacks further illustrate that this is a determined adversary with the resources to develop unique, mission-oriented malware tools.

48 "Ten Days of Rain: Expert analysis of distributed denial-of-service attacks targeting South Korea." McAfee. 2011. http://www.mcafee.com/us/resources/white-papers/wp-10-days-of-rain.pdf

49 "Four Years of DarkSeoul Cyberattacks Against South Korea Continue on Anniversary of Korean War." Symantec. June 26, 2013. http://www.symantec.com/connect/blogs/four-years-darkseoul-cyberattacks-against-south-korea-continue-anniversary-korean-war

50 "South Korea Blames North Korea for Cyberattack." Hamodia. July 17, 2013. http://hamodia.com/2013/07/17/south-korea-blames-north-korea-for-cyberattack/

51 "Analysis of Korean War Anniversary Cyber Attack and Malware." Tripwire. June 27, 2013. http://www.tripwire.com/state-of-security/vulnerability-management/analysis-of-korean-war-anniversary-cyber-attack-malware/

52 "MBR Wiper Attacks Strike Korean Power Plant." Trend Micro. December 23, 2014. http://blog.trendmicro.com/trendlabs-security-intelligence/mbr-wiper-attacks-strike-korean-power-plant/

53 http://hamodia.com/2013/07/17/south-korea-blames-north-korea-for-cyberattack/

CHAPTER

FOUR

4. MALWARE TOOLING

The tool set used by the Lazarus Group overtime has been extensive. To date, more than 45 different malware families have been observed, with the bulk of these families containing strong code-based relationships (code sharing). The Lazarus Group's malware collection breaks down into larger classifications: installers/uninstallers, loaders, destructive malware, remote administration tools (RATs), data exfiltration tools, attack staging/content distribution, distributed denial of service tools, and specific use tools. This section will cover the naming scheme used to classify the malware families, the known infrastructure of the Lazarus Group, and the code relationships that Novetta found, allowing us to link all of these malware families together.

To date, more than 45 different malware families have been observed, with the bulk of these families containing strong code-based relationships (code sharing).

4.1 Naming Scheme

For Operation Blockbuster, Novetta uses a naming scheme to allow the reader to quickly identify the larger class to which a particular malware family belongs. The naming scheme consists of at least two identifiers which each identifier coming from the International Civil Aviation Organization (ICAO)'s phonetic alphabet,[54] commonly referred to as the NATO phonetic alphabet. The first identifier specifies the general classification of the malware family while the second identifier specifies the specific family within the larger general classification. For example, RomeoAlfa specifies a RAT family identified as Alfa.

For the purposes of this paper, the term "family," with respect to malware grouping, is defined as a collection of like malware samples that have a common code base, design and function with a clear evolutionary path. Within a single family there may exists variants that exhibit the same primary criteria of the overall family, but have significant evolutionary differences that allow for additional grouping, but not such that the overall design and functionality of the code base changed to the point of dictating the need for an entirely new family classification. While many of the families are dropped by another family of malware (e.g. a "dropper"), a distinction is made between the malware that drops/installs another piece of malware and the family to which the dropped malware belongs because the two families of malware serve two different functions and have two different designs.

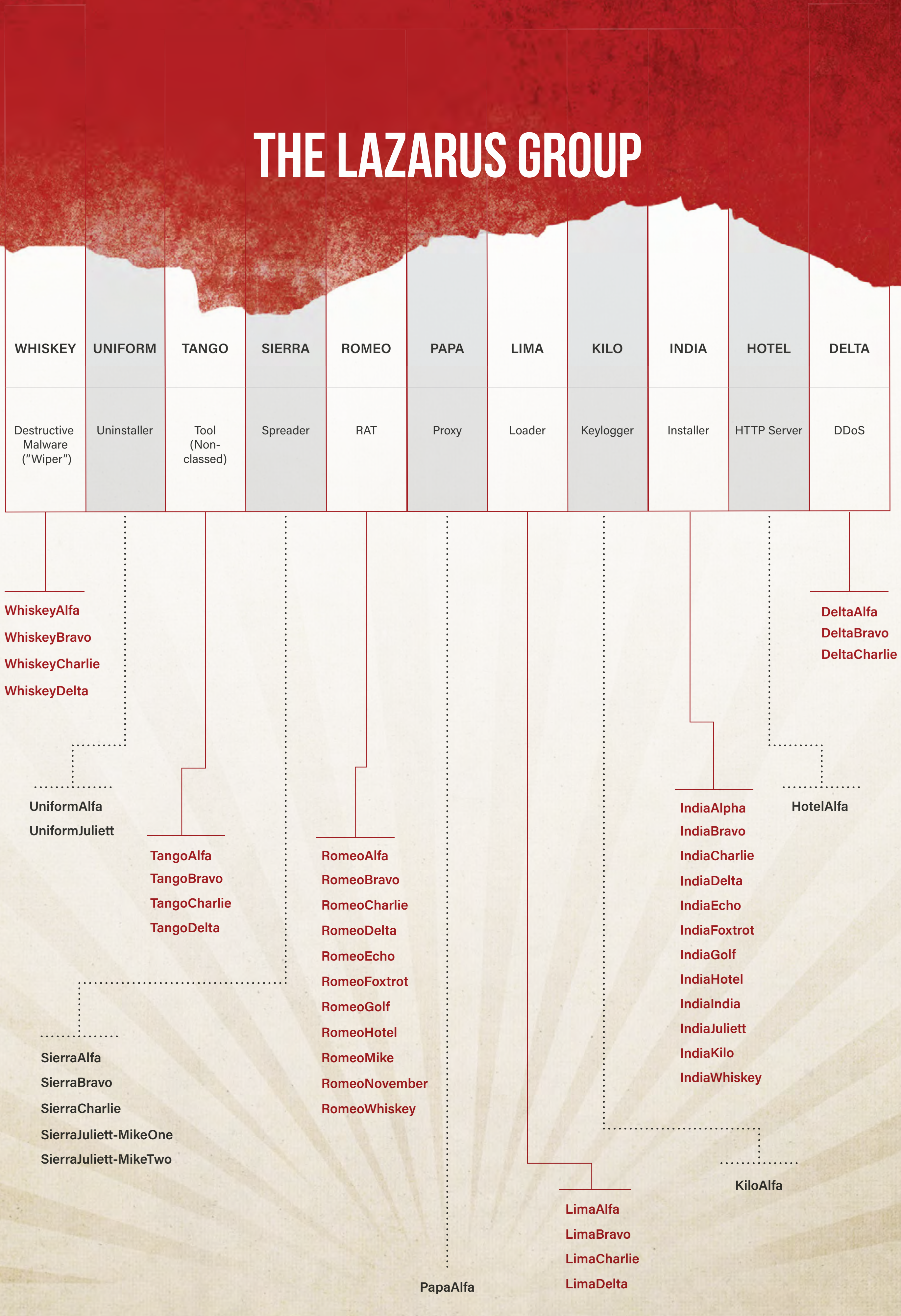
FIRST LEVEL IDENTIFIER	GENERAL CLASSIFICATION
Delta	DDoS
Hotel	HTTP Server
India	Installer
Lima	Loader
Kilo	Keylogger
Papa	Proxy
Romeo	RAT
Sierra	Spreader
Tango	Tool (Non-Classed)
Uniform	Uninstaller
Whiskey	Destructive Malware ("Wiper")

Table 4-1: First Level Identifiers for the Lazarus Group Family Names and their Classification Meanings

There is no temporal component to the second level identifiers given to malware families. While generally the second identifiers are largely sequential (Alfa, Bravo, Charlie, and so on), the identifier does not indicate that one family came before another chronologically. Instead, the second level identifiers were assigned by the order Novetta discovered each particular family.

54 International Civil Aviation Organization. "Alphabet – Radiotelephony". http://www.icao.int/Pages/AlphabetRadiotelephony.aspx Accessed 1 December 2015.

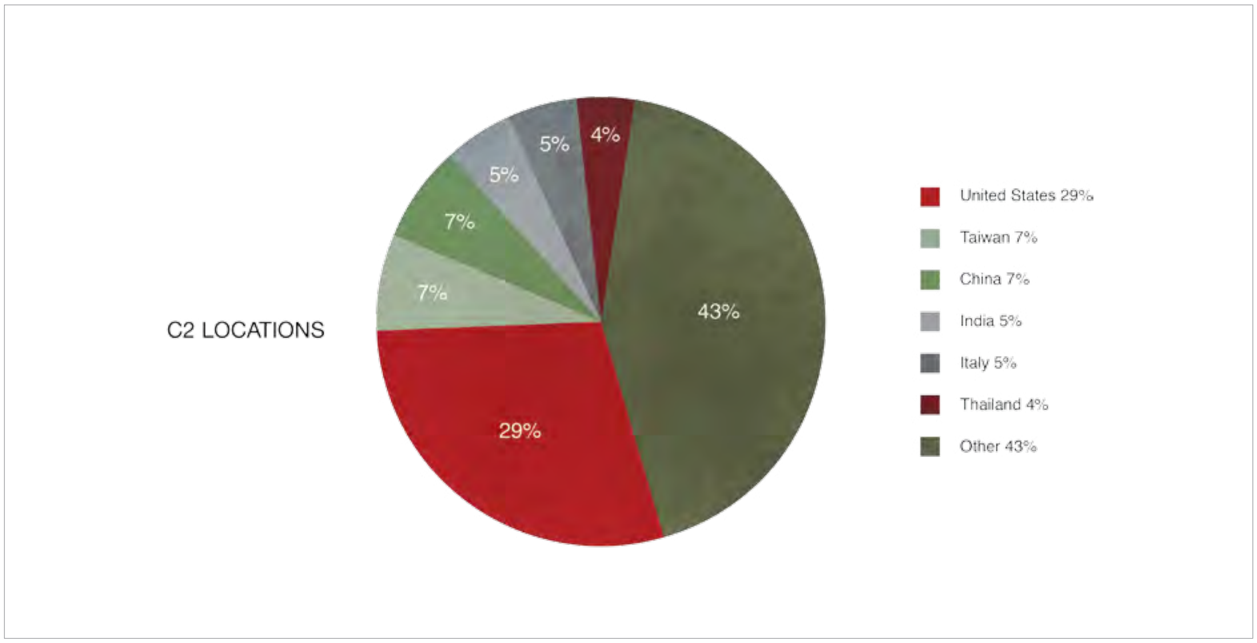
THE LAZARUS GROUP



4.2 Infrastructure

Evidence suggests that parts of the infrastructure used for the malware variants' set C2 touch points are unaffiliated compromised hosts. IP addresses used as C2s include mail server and gaming server IPs (some of which have been listed for spam activity), compromised IPs allocated to educational institutions, public VPNs and proxies, and several IPs that have been publicly posted on forums or pastebin posts with associated usernames and passwords. Given that several identified malware families contain proxy components, it is highly likely that the set C2 touch points are being used as proxies to mask the real C2 server. In the samples Novetta has collected and analyzed the Lazarus Group almost exclusively uses IP addresses over DNS addresses when specifying C2 server locations.

The plurality of identified IP addresses used by the Lazarus Group geo-locate to the United States. Other C2 locations include Taiwan, Indonesia, India, and China.



The Lazarus Group also makes use of P2P-based C2 infrastructure, as seen with the malware family SierraJuliett, whose variants are used as content distribution and attack staging platforms. Notably, such an environment would facilitate access to operators with even low skillsets across all infection instances by providing them a consistent and common operational environment (COE). Based on samples identified by Novetta, this P2P platform has been under active development since 2011, suggesting it was an early developmental priority for the group, likely due to its effectiveness at facilitating sustained operations. The importance of such a uniform environment for operations is not limited to threat actors like the Lazarus Group, but is a real-world priority for the U.S. Army,[55] [56] among others. This suggests that a cyber COE is integral for any well-organized, resourced group tasked with executing difficult operations with varying levels of expertise at an individual operator's level.

55 "Common Operating Environment Architecture: Appendix C to Guidance for 'End State' Army Enterprise Network Architecture." U.S. Army CIO/G-6. October 1, 2010. http://ciog6.army.mil/LinkClick.aspx?fileticket=udbujAHXm-K0%3D&tabid=79

56 "Common Operating Environment assists Army Modernizatio." Army.mil. February 15, 2013. http://www.army.mil/article/96650/Common_Operating_Environment_assists_Army___/

4.3 Code Relationships

The Lazarus Group reuses a significant amount of code, to the point where the reused code snippets have formed a kind of software development kit. As a result of this code sharing and reuse, it is possible to link seemingly disconnected malware families together, as mentioned in Section 2.1. From the extensive similarities of common libraries and shared snippets of code across such a wide variety of malware types, these relationships have allowed Novetta to link the SPE destructive malware to installers, loaders, DDoS malware, network tools, spreaders, RATs, and other destructive malware compiled over a period of several years.

> From the extensive similarities of common libraries and shared snippets of code across such a wide variety of malware types, these relationships have allowed Novetta to link the SPE destructive malware to installers, loaders, DDoS malware, network tools, spreaders, RATs, and other destructive malware compiled over a period of several years.

This section will explore the various shared code fragments found throughout the Lazarus's collection of malware in order to provide a better understanding of why these particular pieces of code are prevalent and how the codes manifest themselves. The shared code breaks down into four major categories: encryption, dynamic API loading, network functionality, and miscellaneous. An appendix to this report details the specific malware families and how they are linked to the Lazarus Group's collective arsenal by code fragments, and Novetta is releasing additional in depth technical reports that further detail the individual malware families.

4.3.1 Encryption

Encryption is a powerful tool for obfuscating the true meaning of information both stored on the victim's hard drive in the form of data files or even within a malware's binary, and when the information is traversing a more public arena such as the Internet. The Lazarus Group has a relatively small set of encryption and encoding schemes that the developer(s) of the various Lazarus Group malware families rely upon. There are several of these encryption and encoding schemes which make excellent indicators of the presence of the Lazarus Group based on their obscurity and uniqueness.

4.3.1.1 Caracachs Encryption

An obscure encryption scheme developed by Alexandre Pukall in 2000, Caracachs is a symmetric stream cipher that takes a minimum of 20 characters (160-bits) as the key. The C source code for Caracachs is freely available on the Internet,[57] but with respect to the implementation of Caracachs within the malware used by the Lazarus group, Caracachs is typically seen encapsulated as a C++ class rather than a C library.

The most notable feature of Caracachs, when viewed within the binaries of the families that use it, is the stream function. The source code for this function takes the form seen in Figure 6-1.

57 `"CARACACHS Cipher" http://ftp.icm.edu.pl/packages/replay.old/libraries/caracash/CARACACH.C 15 September 2015

```
stream(unsigned int *r,unsigned long *index,unsigned long *a,unsigned long *b)
{
  b[*index] = ( b[*index] * (*a) ) + 1;
  *r = _rotl( (*r + (( b[*index] >> 16 ) & 0x7fff)), ((*r)%16) );
}
```

Figure 6-1: Caracachs **stream** Function

After compilation, and subsequent decompilation through Hex-Rays, the function takes the form seen in Figure 6-2.

```
void __stdcall caracachs_stream(DWORD *r, DWORD *index, DWORD *a, DWORD *b)
{
  unsigned int v4; // edx@1
  char v5; // cl@1
  b[*index] = *a * b[*index] + 1;
  v4 = b[*index];
  v5 = ((v4 >> 16) + *(_BYTE *)r) & 0xF;
  *r = ((((v4 >> 16) & 0x7FFF) + *r) << v5) | ((((v4 >> 16) & 0x7FFF) + *r) >> (16 - v5));
}
```

Figure 6-2: Caracachs **stream** Function after Decompilation

The four lines that make up the stream function make a suitable pattern for detecting Caracachs code within a binary. The authors using Caracachs for Lazarus's malware were not terribly original in their use of the cipher suite. In many families, the key used to initialize Caracachs is set to "**abcdefghijklmnopqrstuvwxyz012345\0\0\0\0\0**", which is the similar to the key found within the Caracachs source code. The common function found in multiple families using Caracachs to set the key takes the form seen in Figure 6-3.

```
void __thiscall CCaracachs::GenerateKey(CCaracachs *this)
{
  qmemcpy(this->szPassword, "abcdefghijklmnopqrstuvwxyz012345", sizeof(this-
>szPassword));
  this->dwPasswordLength = 0x20;
  CCaracachs::SetKey(this, 0x20u, this->szPassword);
}
```

Figure 6-3: Caracachs Class's **GenerateKey** Function

The original source code performs the same key initialization feat by using the code snippet seen in Figure 6-4.

```
  strcpy(code,"abcdefghijklmnopqrst"); /* the password */
  longueur=20; /* length of the key up to 256 characters */
  /* init the key */
  pc3init(longueur,code);
```

Figure 6-4: Establishing the Key for Caracachs in the Original Source Code

The authors merely encapsulated the initialization of the cipher within a single member of the C++ class, all without changing the password or even the order of variable assignments. This process of reusing entire code snippets without any modifications appears to be repeated by the developer(s) throughout a number of Lazarus Group tools.

4.3.1.2 Basic XOR with Constant 0xA7

It is not uncommon for malware to use a simple XOR to obfuscate strings and data within a binary. It is also not uncommon for authors to use the same byte across multiple variants of the same malware and even multiple families that can be attributed to the same (set of) authors. By itself, looking at the XOR function within a binary as an indicator of authorship is usually a poor choice. However, combined with other attributes of the surrounding code, an XOR function found in multiple variants and families can provide reassurance that those variants and/or families have some code familiarity.

The Lazarus Group uses simple, but somewhat distinct, XOR obfuscation systems. When dealing with string obfuscations, the Lazarus Group uses the value `0xA7` to transform null-terminated strings by means of XOR each byte within the string by `0xA7`. The `0xA7` scheme is exclusively used for null-terminated strings, as the XOR function depends on a null character to indicate the end of the data to transform. Slight variations appear between families (Figure 6-5 provides one representative example), but two features of the `0xA7` scheme remain constant: the length of the data to transform is calculated by locating the first null and each byte is XOR transformed against the byte `0xA7`.

```
char* __cdecl XorA7(const char *pBuffer)
{
  unsigned char *pOut = malloc(strlen(pBuffer) + 1);
  int j = 0;
  if ( strlen(pBuffer) != 0 )
  {
    p = pOut;
    for ( int i = pBuffer - pOut; ; i = pBuffer - pOut)
    {
      ++j;
      *p = p[i] ^ 0xA7;
      ++p;
      if ( j >= strlen(pBuffer) )
        break;
    }
  }
  pOut[j] = 0;
  return pOut;
}
```

Figure 6-5: Lazarus Group's `0xA7` Transform Function

4.3.1.3 DNSCALC-Style Encoding

DNSCALC is an older malware family, used by several APT groups and first profiled in 2010, whose claim to fame was the use of DNS lookups for domain names that would return specific IP addresses used to calculate the listening port number for the C2 server. One notable feature of DNSCALC was the use of a combination of XOR with an ADD operation and XOR with a SUB operation for the purposes of encrypting and decrypting data streams. Since at least 2011, the Lazarus Group has commandeered this technique for use in a variety of their malware families. The DNSCALC version of this encoding/decoding scheme performed the transformation operation on each byte using two lines of C code such as

```
d += 122;
d ^= 25;
```

where the values 122 and 25 constitute the encryption and decryption keys. The Lazarus Group performs the same operation in a single line of code, such as

```
d = (e ^ 25) – 122
```

and

```
e = (d + 122) ^ 25
```

This subtle, but important, distinction in style indicates that the code was not directly copied from DNSCALC, but rather was inspired by DNSCALC or another source that performs the same transform. It should be noted that DNSCALC modified the Ghost RAT **MyEncode** function, seen below, by reversing the order of operations meaning that the Lazarus Group's use of the encoding scheme represents a derivation of an existing derivation.

```
char* MyEncode(char *str)
{
        int             i, len;
        char    *p;
        char    *s, *data;
        len = strlen(str) + 1;
        s = (char *)malloc(len);
        memcpy(s, str, len);
        for (i = 0; i < len; i++)
        {
                s[i] ^= 0x19;
                s[i] += 0x86;
        }
        base64_encode(s, len, &data);
        free(s);
        return data;
}
```

The DNSCALC-style encoding scheme code is heavily used throughout many of the various malware families for which the Lazarus Group is responsible.

4.3.1.4 Space-Dot Encoding

Strings, especially when used to dynamically load Windows API functions at runtime, provide a significant amount of surface for antivirus and host-based IDS to detect potentially malicious code. For this reason, it is not uncommon for malware authors to obfuscate strings that identify the Windows API functions the malware will attempt to dynamically load. Simple obfuscations are generally more than adequate to defeat string-based detection systems, allowing attackers to use simple XORs or character substitution techniques to get around detection. The Lazarus Group used a simple method to confuse systems looking for the API names they were to load. Instead of obfuscating the name by transforming individual characters, the names were interrupted with unnecessary characters such as dots, spaces, greater than, less than, and underscore characters. This broke up names such as **ChangeServiceConfig2A** into “**Cha>nge>Ser>vi>>ceCo>nfi>g2A**.”

Novetta has dubbed this scheme of inserting junk characters into API name strings as “Space-Dot Encoding” based on the fact that the bulk of the implementations of the system only introduces spaces and dots. In order to recover the original, unmolested string, the Space-Dot decoding function will scan character by character through the supplied string, copying each byte to a global buffer so long as the character does not match one of the undesirable characters. Upon completion of the function, a pointer to the buffer containing the desired string is returned to the caller. The function that performs the decoding takes the form of seen in Figure 6-6.

```
char *__cdecl DecodeString(char *pzString)
{
  char* p = pzString;
  char* b = g_decodingBuffer;
  memset(decodingBuffer, 0, 0x50u);
  while ( *p )
  {
    char c = *p;
    if ( *p != '<' && c != '>' && c != '_' && c != ' ' && c != '.' )
      *b++ = c;
    ++p;
  }
  return g_decodingBuffer;
}
```

Figure 6-6: Space-Dot Decoding Function

As the usage of the Space-Dot Encoding aged, the authors removed “**>**”, “**<**”, and “ **_** ” from the character set and instead relied on only spaces and dots to provide the necessary junk characters to throw off detection systems. The result is a slightly simpler if statement, but otherwise the remainder of the Space-Dot decoding function remained constant throughout the use of the scheme in the Lazarus Group’s malware.

4.3.1.5 RSA Encryption

Several families within the Lazarus Group’s malware collective use public/private key encryption. Some use the encryption for securing documents that the malware exfiltrates, while others use it for signing and authenticating commands. Regardless of the use, the malware families using the RSA scheme share a common code library to implement the cryptographic functionality.

Public/private key encryption, or asymmetric encryption, is a form of encryption where the key used to encrypt data differs from the key used to decrypt the data. The effect of having asymmetric encryption in malware is that the authors and/or operators of the malware can embed the decryption key for commands into the malware while retaining the encryption key for themselves. This restricts others from issuing commands to the malware since the encryption key is not known, thereby preventing those not associated with the malware from attempting to inject commands.

Based on **CRSA**,[58] the Lazarus Group's implementation of RSA wraps the **CRSA** class into a single function for encryption and decryption (Figure 6-7).

```
char * _ _cdecl RSATransform(int mode, char *pvKey, int dwKeyLength, char *pvIn, int
dwOutBufSize, char *pvOut, DWORD *pdwOutputLength)
{
  int v8; // ecx@2
  int v9; // eax@4
  char *result; // eax@7
  signed int v11; // eax@12
  CRSA rsa; // [sp+10h] [bp-58h]@1
  int eh; // [sp+64h] [bp-4h]@1
  CRSA::CRSA(&rsa);
  eh = 0;
  if ( pvOut
    || ((v8 = (dwKeyLength + 7) >> 3, !mode) || mode == 1 ? (v9 = (dwOutBufSize - 1) /
(v8 - 8) + 1) : (v9 = (dwOutBufSize - 1) / v8 + 1, v8 -= 8),
        (pvOut = (char *)LocalAlloc(0x40u, v8 * v9)) != 0) )
  {
    if ( mode && mode != RSA _ PUB _ DEC )
      CRSA::SetPrivKey(&rsa, pvKey, dwKeyLength);
    else
      CRSA::SetPubKey(&rsa, pvKey, dwKeyLength);
    v11 = CRSA::transform(&rsa, mode, pvIn, dwOutBufSize, pvOut);
    if ( pdwOutputLength )
      *pdwOutputLength = v11;
    eh = -1;
    CRSA::Dstr(&rsa);
    result = pvOut;
  }
  else
  {
    eh = -1;
    CRSA::Dstr(&rsa);
    result = 0;
  }
  return result;
}
```

Figure 6-7: The Lazarus Group's RSA Encapsulation Function as seen after Decompilation

The **RSATransform** function is a unique implementation that appears to be specific to the Lazarus Group, thereby making it a valuable identifier of malware related to the group. The function can operate in one of four modes: public key encryption, public key decryption, private key encryption, and private key decryption. However, across the various identified samples that use **RSATransform**, only the public key encryption and decryption modes have been observed by Novetta.

58 "RSAUtil.cpp RSA.cpp" http://read.pudn.com/downloads145/sourcecode/windows/system/633068/RSAUtil/RSA.cpp__.htm 16 March 2004

4.3.1.6 Shared Public Key

While not necessarily a shared library, the use of a common public key is a definitive, identifiable characteristic that can link multiple families of malware to a common actor or actor set. With respect to the Lazarus Group, there is a common public key that is used in multiple families within the group's collective. This fact would indicate that there is a single private key that is shared across malware for decryption/authentication, controlled by the Lazarus Group. The reuse of cryptographic keys has also been discussed by security researchers profiling both Operation Troy and Operation 1Mission. Found originally in a variant of SierraJuliett[59] family of malware from 2011, the following 1024-bit key has been identified in malware as recently as 2015:

```
47A713F89BBC74CBCE771E0F00A039561BC566F394B1EA2271DE2B42CCE9F72F31E722B06FBB0203FC0A2F51E-
ED054250EE34FF09FBAE7AC20D694E6BAD3AB4CD98CFD1C7FBA4875E5853966881EE9C9745106DECBC1D13747B-
61C629AB2DCFCB809CE88C5927DF017E75B8262F96AE4EEDBE65DC9185D202A32C3E807CD99CE
```

To date, the 1024-bit key has been observed in samples from the RomeoWhiskey, SierraBravo, and SierraCharlie families.

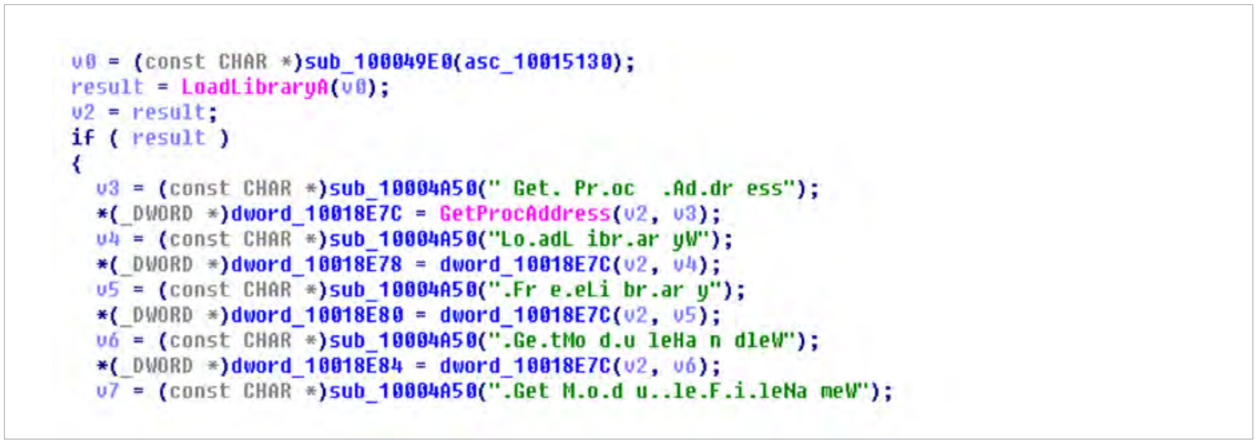
4.3.2 Dynamic API Loading

Dynamic API loading is a technique in which the standard Windows functions `LoadLibrary` and `GetProcAddress` are used to dynamically load desired API functions at run time. The import table of a binary can easily give away the intent of the executable. For example, a binary that has SetWindowsHookEx and several of the Winsock API functions is most likely a network-capable keylogger. As such, certain combinations of API function imports can indicate suspicious behavior, allowing antivirus vendors to use such indicators when determining the intent of a binary through their various heuristic detection schemes. Therefore, it is beneficial for malware authors to obfuscate the more severe or telling API functions they need to load and keep these functions out of the import table of the binary. This leads to the use of dynamic API loading schemes.

Dynamic API loading allows the malware authors to remove the names of the telling APIs from the import table but still requires the malware authors to provide the full name of the desired API functions to `GetProcAddress`. This leads to another facet of dynamic API loading: API name obfuscation. `GetProcAddress`, in order to load any API function into memory, requires either an ordinal number identifying the API function in question or the name of the API function. It is rare that the ordinal number is used, as the ordinal number could, in theory, change from version to version of Windows and therefore requires a significant amount of code maintenance on the part of the author. However, API names do not change between versions, so authors can simply obfuscate the name of the desired API functions up to the point of calling `GetProcAddress`. The obfuscation of API names, in string form within the binary, can allow malware authors to avoid string-based signature detection which increases the chances of a malware binary evading simpler AV signature detection. Additionally, the use of API name obfuscation requires additional work on the part of the reverse engineers analyzing the malware since the analyst must now reconstruct the original API names.

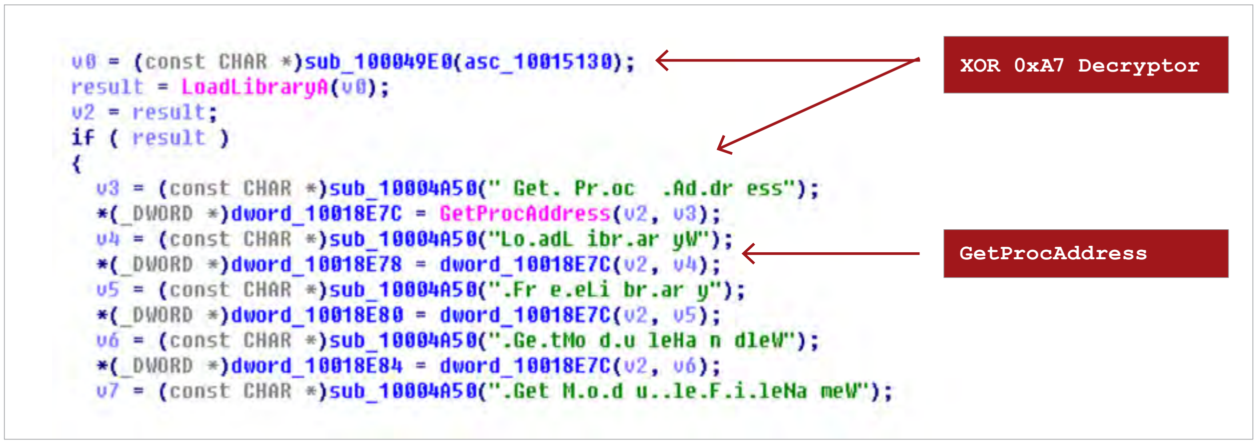
A common feature of the malware families under the Lazarus Group's umbrella is the use of dynamic API loading. The structure of dynamic API loading in most malware is typically to decrypt the API name string then load the API via `GetProcAddress`. The Lazarus Group adheres to this same model. However, it is the use of the decryption schemes that are specific to the Lazarus Group and allow for easy identification of malware related to the group. There are two predominate versions of dynamic API loading found in the majority of the Lazarus Group's malware: XOR 0xA7 with Space-Dot (Figure 6-8) and simply XOR 0xA7 (Figure 6-9).

59 <reference to external report>



```
v0 = (const CHAR *)sub_100049E0(asc_10015130);
result = LoadLibraryA(v0);
v2 = result;
if ( result )
{
  v3 = (const CHAR *)sub_10004A50(" Get. Pr.oc  .Ad.dr ess");
  *(_DWORD *)dword_10018E7C = GetProcAddress(v2, v3);
  v4 = (const CHAR *)sub_10004A50("Lo.adL ibr.ar yW");
  *(_DWORD *)dword_10018E78 = dword_10018E7C(v2, v4);
  v5 = (const CHAR *)sub_10004A50(".Fr e.eLi br.ar y");
  *(_DWORD *)dword_10018E80 = dword_10018E7C(v2, v5);
  v6 = (const CHAR *)sub_10004A50(".Ge.tMo d.u leHa n dleW");
  *(_DWORD *)dword_10018E84 = dword_10018E7C(v2, v6);
  v7 = (const CHAR *)sub_10004A50(".Get M.o.d u..le.F.i.leNa meW");
```

Figure 6-8: Dynamic API Loading Function using Both XOR 0xA7 and Space-Dot Encoding



```
v0 = (const CHAR *)sub_100049E0(asc_10015130);
result = LoadLibraryA(v0);
v2 = result;
if ( result )
{
  v3 = (const CHAR *)sub_10004A50(" Get. Pr.oc  .Ad.dr ess");
  *(_DWORD *)dword_10018E7C = GetProcAddress(v2, v3);
  v4 = (const CHAR *)sub_10004A50("Lo.adL ibr.ar yW");
  *(_DWORD *)dword_10018E78 = dword_10018E7C(v2, v4);
  v5 = (const CHAR *)sub_10004A50(".Fr e.eLi br.ar y");
  *(_DWORD *)dword_10018E80 = dword_10018E7C(v2, v5);
  v6 = (const CHAR *)sub_10004A50(".Ge.tMo d.u leHa n dleW");
  *(_DWORD *)dword_10018E84 = dword_10018E7C(v2, v6);
  v7 = (const CHAR *)sub_10004A50(".Get M.o.d u..le.F.i.leNa meW");
```

Figure 6-9: Dynamic API Loading Function Utilizing only a Single Encoding Scheme (`XOR 0xA7`)

Another feature of the dynamic API loading used by the Lazarus Group is not immediately apparent at first glance: consistency. Typically, when the Lazarus Group uses dynamic API loading within a binary, each function will load one DLL at a time. For example, there is a function that will load the necessary API functions from `kernel32.dll`, there is another function for loading API functions from `advapi32.dll`, and so on. These individual functions are shared across samples both within families and among other families. The dynamic API loading functions generally are not tailored for a specific malware family. This is seen in many examples where a dynamic API loading function will load API functions into memory that the malware does not use it, or even reference it, beyond the initial load. This indicates that the dynamic API loading functions are part of a larger library of functions and, as such, provide a viable indicator of code specific to the Lazarus Group.

4.3.3 Network Functionality

The way a developer interacts with a network touch point can provide a fingerprint of the developer. When the developer builds a library for network interaction and uses the library in multiple malware families, analysts can easily identify related families based on the code reuse. The developer(s) of the Lazarus Group's malware routinely use network routines and techniques across multiple families within the Lazarus Group's malware collective. This section explores several of the more prominent techniques the developer(s) used in the Lazarus Group malware families.

4.3.3.1 Fake TLS Communication

Several of the families within the Lazarus Group's arsenal employ a rather unique form of communication encryption that mimics TLS communication but ultimately uses a completely different encryption method. This type of communication has the advantage of appearing to be legitimate TLS traffic, thereby evading many network-based IDS detections and at the same time protecting against SSL man-in-the-middle decryption attacks that would reveal the contents of the encrypted communication.

The fake TLS communication begins when a sample opens a socket between the itself and its corresponding C2 server, and the client side of the channel sends a TLS **`ClientHello`** packet. The basic format of a TLS **`ClientHello`** packet is as follows:

```
struct {
    ProtocolVersion client _ version;
    Random random;
    SessionID session _ id;
    CipherSuite cipher _ suites<2..2^16-2>;
    CompressionMethod compression _ methods<1..2^8-1>;
    select (extensions _ present) {
        case false:
            struct {};
        case true:
            Extension extensions<0..2^16-1>;
    };
} ClientHello;
```

Figure 6-10: RFC 5246 Definition of the **`ClientHello`** Packet

The **`ClientHello`** packet will vary for each communication but will contain some common characteristics. When constructing the **`ClientHello`** packet, the Trojan probabilistically determine which sections to include and the values of those sections, with the exceptions of the **`client _ version`** field, which is static at TLS 1.0 **`(0x301)`**, and the **`compression _ methods`** field, which is set to empty. The Trojan fills the random field with a 32-byte random value generated using the **`rand`** API function. The first four bytes of the field are replaced with the current time as supplied by the **`time`** API function. The **`session _ id`** field will only appear if the value of **`fIncludeSessionIDTest2`** is non-zero as defined by the following section of code:

```
fIncludeSessionIDTest1 = rand() & 0x80000007;
  fIncludeSessionIDTest2 = fIncludeSessionIDTest1 == 0;
  if ( (fIncludeSessionIDTest1 & 0x80000000) != 0 )
    fIncludeSessionIDTest2 = ((( _ BYTE)fIncludeSessionIDTest1 - 1) | 0xFFFFFFF8) == -1;
```

If the **`session _ id`** field is included in the **`ClientHello`**, the value is filled with a 32-byte randomly generated value, again using the **`rand`** API function.

The **`cipher _ suite`** value is always present and is one of four predefined values. To determine which of the predefined suite sets to use, the fake TLS scheme will again rely on the **`rand`** API function. Assuming the PRNG of rand is suitably random, this means that there is a 25% chance for any particular cipher suite being selected. Table 6-1 below provides the possible cipher suites that the fake TLS scheme uses.

SUITE	SUITE 2
(12 Entries)	(11 Entries)
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	TLS_DHE_DSS_WITH_DES_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
TLS_RSA_WITH_RC4_128_MD5	
SUITE 3	**SUITE 4**
(36 Entries)	(36 Entries)
TLS_EMPTY_RENEGOTIATION_INFO_SCSV	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA	TLS_DHE_DSS_WITH_RC4_128_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_ECDH_RSA_WITH_RC4_128_SHA	TLS_ECDH_RSA_WITH_RC4_128_SHA
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
TLS_ECDH_ECDSA_WITH_RC4_128_SHA	TLS_ECDH_ECDSA_WITH_RC4_128_SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_SEED_CBC_SHA	TLS_RSA_WITH_SEED_CBC_SHA
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_MD5	TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA

Table 6-1: Fake TLS Scheme's Predfined Cipher Suites

The extensions field provides the area of the greatest variability within a `ClientHello` packet generated as part of the fake TLS communication scheme. The Trojan may include zero or more of the following extensions (in order):

The extensions field provides the area of the greatest variability within a ClientHello packet generated as part of the fake TLS communication scheme. The Trojan may include zero or more of the following extensions (in order):

- renegotiation _ info (80% probability)
- server _ name (80% probability)
- status _ request (80% probability)
- ellipic _ curves with ec _ point _ formats (80% probability)
- SessionTicket TLS (10% probability)
- next _ protocol _ negotiation (10% probability)

The renegotiation _ info, SessionTicket TLS and next _ protocol _ negotiation extensions all have a 0 byte length, thereby remaining static in their values. The server _ name extension will use either www.amazon.com or www.google.com as the name of the server to which the TLS client appears to be connecting for the majoriy of the Lazarus Group's Trojans that employ the fake TLS scheme (there is a 50% probability of either domain name being choose by the Trojan). A smaller number of Trojans that employ the fake TLS communication scheme can have up to 34 domain names to choose from. Table 6-2 identifies the list of hardcoded domains found in various families within the Lazarus Group's collection for use in the server _ name extension. Note that not all family members contain all domain names.

accounts.google.com	secureir.ebaystatic.com
apps.skypeassets.com	securemetrics.apple.com
b.stats.ebay.com	signin.ebay.com
daw.apple.com	skydrive.live.com
extended-validation-ssl.verisign.com	ssl.google-analytics.com
fls-na.amazon.com	ssl.gstatic.com
images-na.ssl-images-amazon.com	sstats.adobe.com
login.live.com	startpage.com
login.skype.com	support.msn.com
login.yahoo.com	support.oracle.com
s.imp.microsoft.com	supportprofile.apple.com
s1-s.licdn.com	urs.microsoft.com
sc.imp.live.com	verify.adobe.com
secure.logmein.com	www.adobetag.com
secure.shared.live.com	www.apple.com
secure.skype.com	www.amazon.com
secure.skypeassets.com	www.google.com

Table 6-2: Observed server _ name Field Values

The status _ request extension will, if present, always have the Certificate Status Type field set to OCSP (1). Table 6-3 lists the possible sets that the fake TLS scheme may apply to the elliptic _ curves field.

ELLIPTICAL CURVE SET 1 (3 CURVES)	ELLIPTICAL CURVE SET 2 (4 CURVES)
SECT193R1	SECT233K1
SECP256R1	SECP256R1
SECP384R1	SECP384R1
	SECP512R1

Table 6-3: The Fake TLS Scheme's Possible elliptic _ curves Sets

If the `elliptic_curves` extension is present, it is always followed by the `ec_point_formats` extension which defines a single format of uncompressed `(0)`. The probability of either elliptical curve set being used is defined by the same random selection algorithm used when determining if the `session_id` field will occur within the `ClientHello`.

After the client side of the communication sends the `ClientHello` packet, the client expects the next data received from the server to be a `ServerHello` packet. If the data that arrives from the server is not a `ServerHello`, the connection terminates. The `ServerHello` response may or may not have a `session_id` field, but the contents of this field are irrelevant to the client. The client will process the `ServerHello` packet only far enough to extract the selected cipher suite and then reads and disregards any incoming packets until the server sends the `ServerHelloDone` packet (up to 8 server packets).

After receiving the `ServerHelloDone` packet, the connection between the client and the server is complete. Further communication is encapsulated in what appears to be a legitimate TLS frame. The header for every datagram transmitted between the client and server (and vice versa) consists of a 5-byte header that specifies the type of datagram (typically set to `22`), the TLS version (set to `0x0301`), and the number of bytes within the datagram. Following the TLS frame header, the payload bytes are transmitted. The payload contains the data encrypted using the Caracachs encryption scheme (see Section 6.1.1).

4.3.3.2 C2 Connections

Several of the malware families under the Lazarus Group umbrella use a common function for connecting to a C2 server. While most malware that uses the Winsock API will use `socket` and `connect` to open a socket between two end points, what makes the C2 server connection function identifiable is the method by which the authors generate and test the connection (Figure 6-11).

```
int ConnectToHost(int dwIP, u_short wPort, signed int dwTimeout)
{
  __int32 actualTimeout; // edi@3
  SOCKET s; // esi@6
  u_long argp; // [sp+44h] [bp-120h]@1
  struct timeval timeout; // [sp+48h] [bp-11Ch]@8
  sockaddr_in endpt; // [sp+50h] [bp-114h]@6
  fd_set writefds; // [sp+60h] [bp-104h]@8
  argp = 1;
  if ( wPort && dwIP )
  {
    actualTimeout = dwTimeout;
    if ( dwTimeout <= 0 || dwTimeout > 60 )
      actualTimeout = 10;
    endpt.sin_family = 2;
    endpt.sin_addr.S_un.S_addr = dwIP;
    endpt.sin_port = htons(wPort);
    s = socket(2, 1, 0);
    if ( s != -1 && ioctlsocket(s, 0x8004667E, &argp) != -1 )// disable blocking
    {
      connect(s, (const struct sockaddr *)&endpt, 16);
      writefds.fd_array[0] = s;
      writefds.fd_count = 1;
      timeout.tv_sec = actualTimeout;
      timeout.tv_usec = 0;
      if ( select(s + 1, 0, &writefds, 0, &timeout) > 0 && __WSAFDIsSet(s, &writefds) )
      {
        argp = 0;
        ioctlsocket(s, 0x8004667E, &argp);      // enable blocking
        return s;
      }
      closesocket(s);
    }
  }
  return -1;
}
```

Figure 6-11: Common C2 Server Connection Function found in Lazarus Group Families

The authors perform the standard procedure of generating a virtual circuit between two end points by calling the socket API function to generate a socket object. Next the authors disable socket read blocking by calling **`ioctlsocket`** with the value **`0x8004667E`**. The code then proceeds to call **`connect`** to establish a virtual circuit between the Trojan and the C2 server. In order to test the validity of the channel, the code will call **`select`** followed by **`_WSAFDIsSet`** to determine if the Trojan can send data through the socket. If the socket is viable, read blocking is re-enabled via an **`ioctlsocket`** call, and the socket is returned to the caller of the function.

4.3.3.3 Socket Disconnect

Many of the RATs employed by the Lazarus Group have a unique method for closing active network socket connections. A typical solution to terminate a connection between two end points is to simply call the **`closesocket`** API function, which abruptly closes a socket channel. The authors responsible for the Lazarus Group's malware take a slightly more aggressive approach, however. The general form for disconnecting a socket employed by the Lazarus Group's malware consists of sending a WORD (2 byte) or DWORD (4 byte) value, usually equal to **`0x0001`** or **`0x00000001`**, to the other receiving end of the socket followed by calling the **`shutdown`** API function which instructs the WinSock API to close both directions of communication. The final step in terminating a socket connection is the call to **`closesocket`**. There are slight variations on this method exist where **`setsockopt`** is called to allow for lingering sockets or where a different DWORD value is transmitted to the receiving end, but the basic pattern of **`send/shutdown/closesocket`** remains consistent. Below are several example variations.

```
int __cdecl SendErrorAndCloseSocket(int skt)
{
  if ( skt == -1 )
  {
    return -1;
  }
  int v5 = 1;
  int val = 0x10001;
  setsockopt(skt, SOL_SOCKET, SO_LINGER, (const char *)&val, 4);
  send(s, (const char *)&v5, 2, 0);
  shutdown(skt, 2);
  closesocket(skt);
  return 0;
}
int __thiscall FlushAndShutdownSocket(void *pfSuccess, SOCKET s)
{
  DWORD buf = 0;
  char optval[4];
  strcpy(optval, "\x01");
  setsockopt(s, 0xFFFF, SO_LINGER, optval, 4);
  send(s, buf, 4, 0);
  shutdown(s, 2);
  result = closesocket(s);
  *pfSuccess = 0;
  return result;
}
int __cdecl ShutdownConnection(SOCKET s)
{
  __int16 v2 = 1;
  int v4 = 0x26380B;
  setsockopt(a1, 0xFFFF, 128, (const char *)&v2, 4);
  send(s, (const char *)&v4, 4, 0);
  shutdown(s, 2);
  return closesocket(s);
}
```

Figure 6-12: Common Forms of the Lazarus Group's Connection Disconnect Functions

4.3.3.4 Common Network Data Transmission and Receiving Function

The Lazarus Group uses a common structure for transmitting and receiving data over the network. For network communication that uses encryption, the developer(s) of the Lazarus Group's malware abstracts the data shuttling functionality that takes the burden of managing the encryption component of the communication channel off of the core code. The use of such a design pattern, a pattern that has been observed used more and more as the code within the Lazarus Group's code has matured, indicates a level of attention to modularity in design.

The design pattern used for the transmission of data to a remote end point takes the form seen in Figure 6-13. The prototype for the transmission function is consistent across a larger number of the malware families, with the first parameter being the socket, the second and third parameters defining the location and size of the data to transmit, and the final argument being a flag to encrypt the transmission (if non-zero).

```
int SendData(SOCKET skt, void *pvData, int dwSize, int fEncrypt)
{
  int dwXmitted;
  int dwBytesSent = 0;
  unsigned char* p = pvData;
  if ( fEncrypt )
  {
     /*
           Family specific encoding scheme
     */
  }
  if ( dwSize <= 0 )
    return 1;
  while ( 1 )
  {
    dwXmitted = send(skt, &pvData[dwBytesSent], dwSize - dwBytesSent, 0);
    if ( dwXmitted <= 0 )
      break;
    dwBytesSent += dwXmitted;
    if ( dwBytesSent >= dwSize )
      return 1;
  }
  return 0;
}
```

Figure 6-13: Common Form for Network Data Transmission with Encryption

The exact encryption scheme used varies from family to family. Regardless, the overall pattern remains the same with very few exceptions across the entirety of the Lazarus Group's collection.

There are two main reciprocal functions for receiving data from the network as Figure 6-14 and Figure 6-15 illustrate. The design pattern for the receiving of potentially encrypted data consists of reading the data from the network until the specified number of bytes has been received (or a timeout occurs, in the case of **RecvDataEx** variants) and if the decrypt flag is set to non-zero, apply the family-specific decryption scheme to the buffer.

```
int RecvData(SOCKET skt, void *pvData, int dwLength, int fDecrypt)
{
  int dwBytesRead = 0;
  if (skt == -1 )
    return 0;
  int dwBytesRemaining = dwLength;
  if ( dwLength > 0 )
  {
    do
    {
      int dwBytesRecv = recv(skt, &pvData[dwBytesRead], dwLength - dwBytesRead, 0);
      if ( dwBytesRecv <= 0 )
        return 0;
      dwBytesRead += dwBytesRecv;
    }
    while ( dwBytesRead < dwLength );
  }
  if ( fDecrypt && dwLength > 0 )
  {
     /*
           Family specific decoding scheme
     */
  }
  return 1;
}
```

Figure 6-14: Common Form for Receiving Network Data with Encryption

```
int RecvDataEx (SOCKET skt, void *pvData, int dwSize, int fDecode, int timeout)
{
  int dwBytesRemaining; // edi@1
  _ BYTE *p; // ecx@1
  int dwBytesRead; // esi@1
  int dwBytesRecv; // eax@3
  int v8; // eax@8
  signed int result; // eax@12
  int dwBytesRemaining = dwSize;
  int dwBytesRead = 0;
  if ( dwSize > 0 )
  {
    while ( WaitForRead(skt, timeout) )
    {
      int dwBytesRecv = recv(skt, &pvData[dwBytesRead], dwSize - dwBytesRead, 0);
      if ( dwBytesRecv <= 0 )
        break;
      dwBytesRead += dwBytesRecv;
      if ( dwBytesRead >= dwSize )
      {
        if ( fDecode && dwSize > 0 )
        {
           /*

              Family specific decoding scheme

           */
        }
        return 1;
      }
    }
  }
  return 0;
}
```

Figure 6-15: Common Form for Receiving Network Data with Encryption and Receive Timeout

The abstraction of the network data shuttling has the added benefit of allowing a malware family to use the same function call regardless of the underlying data format, encrypted or cleartext. The use of this behavior is found in several Lazarus Group families when the initial handshake to establish an encrypted channel requires sending cleartext followed by a switch to an encrypted mode after the handshake has been established. When such a use case occurs, the same send and receive abstract functions can be used, but their encrypted/decrypted mode flags will be the only change the programmers of the core code must concern themselves with.

4.3.3.5 Suicide Scripts

A suicide script is a method by which a running executable can ensure, upon termination, that its presence is removed from a host system. As running executable are locked by Windows, it is necessary for malware binaries to deploy suicide scripts in order to remove themselves from a victim's machine. The typical suicide script consists of a Windows batch file that enters an infinite loop attempting to delete the source executable over and over until it is finally successful (after the running program terminates).

While many unrelated malware families use suicide scripts, there are times when a suicide script can give away a common author or library. This is the case with the Lazarus Group's suicide scripts. Novetta has observed five distinct suicide scripts that span across multiple malware families attributed to the Lazarus Group. These observed suicide scripts largely follow the same pattern: a short label (a single letter with an option single number), a file deletion attempt, a file check, a conditional loop, and finally a file delete to remove the suicide script.

```
:L1
del “<source binary filename>”
if exist “<source binary filename>” goto L1
del “<suicide script filename>”
```

```
@echo off
:R1
del /a “<source binary filename>”
if exist “<source binary filename>” goto R1
del /a “<suicide filename>”
```

```
:R
IF NOT EXIST <source binary filename> GOTO E
del /a <source binary filename>
GOTO R
:E
del /a d.bat
```

```
:Hello
del /a <source binary filename>
if exist <source binary filename> goto Hello
del /a <suicide filename>
```

```
@echo off
:D1
del /a <source binary filename>
if exist %1 goto D1
del /a <suicide filename>
```

```
@echo off
:Loop
del /a H “<source binary filename>”
if exists “” goto Loop
del “<suicide filename>”
```

```
:Repeat1
del “<source binary filename>”
if exist “<source binary filename>” goto
Repeat1
del “<suicide script filename>”
```

Figure 6-16: Suicide Script Forms Found within Lazarus Group Families

A common design pattern for generating many of the suicide scripts is to construct each line one at a time. When decompiled in Hex-rays, a typical suicide script construction function takes the following form:

```
strcat(szSuicideScriptFilename, “PM0D4.bat”);
fp = fopen(szSuicideScriptFilename, “wb”);
fprintf(fp, “:Repeat1\r\n”);
fprintf(fp, “del \”%s\”\r\n”, szSourceFileName);
fprintf(fp, “if exist \”%s\” goto Repeat1\r\n”, pszSourceFileName);
fprintf(fp, “del \”%s\”\r\n”, szSuicideScriptFilename);
fclose(fp);
```

or

```
strcpy(szScript, “@echo off\r\n”);
strcpy(szScript, “:Loop\r\ndel /a H \””);
strcat(szScript, szSourceFileName);
strcat(szScript, “\”\r\nif exist \””);
strcat(szScript, szSourceFileName);
strcat(szScript, “\” goto Loop\r\ndel \””);
strcat(szScript, szSuicideScriptFilename);
strcat(szScript, “\””);
WriteFile(fp, szScript, strlen(szScript), &NumberOfBytesWritten, 0);
CloseHandle(fp);
```

The other design pattern for generating suicide scripts is a more streamlined approach in which the entire content of the suicide script is constructed and then written to file as follows:

```
  fp = fopen(&Buffer, “wt”);
  if ( fp )
  {
    fprintf(fp, “:L1\r\ndel \”%s\”\r\nif exist \”%s\” goto L1\r\ndel \”%s\”\r\n”,
szSourceFileName, szSourceFileName, szSuicideScriptFilename);
    fclose(fp);
```

4.3.4 Directory Hierarchy Verification and Generation

From time to time it is necessary to verify the existence of a particular file path and, if the path fails to exist, create the file path. The Lazarus Group uses a specific function for this task in several of its family members. What makes the code distinguishable is the fact that the function will take a file's full path (e.g. **`C:\temp\folder1\folder2\malware.exe`**) and traverse the entire path. At each level of the directory hierarchy, the code will ensure that the directory exists. At the same time, the code allows the caller of the function to specify if the highest level of the hierarchy is a directory name or a filename. The ability to allow the caller to specify this means the function was originally designed to accommodate both file paths and directory paths.

The function that the Lazarus Group uses for ensure a directory hierarchy is as follows:

```
void GenerateDirectoryPath(char *pszPath, int fLastEntryIsDir)
{
  char *p;
  const char *pn;
  char *v4;
  char *v5;
  char szDirPath[260];
  if ( pszPath)
  {
    p = strchr(pszPath, '\\');
    pn = p + 1;
    if ( p != (char *)-1 && strchr(pn, '\\') )
    {
      do
      {
        memset(szDirPath, 0, 260);
        v4 = strchr(pn, '\\');
        strncpy(szDirPath, pszPath, v4 – szDirPath);
        v5 = strchr(pn, '\\');
        pn = v5 + 1;
        if ( v5 == (char *)-1 )
          break;
        if ( GetFileAttributesA(szDirPath) == -1 )
          CreateDirectoryA(szDirPath, 0);
      }
      while ( strchr(pn, '\\') );
    }
    if ( fLastEntryIsDir )
      CreateDirectoryA(pszPath, 0);
  }
}
```

The traversal function begins at the first directory separator (the backslash) and verifies that the path up to that particular point exists by calling **`GetFileAttributesA`** to determine if the path if valid or not. If the path to that point is not valid, **`CreateDirectoryA`** is called to generate the folder. The process is repeated for each of the additional directories in the path until the final directory separator character is found. If the **`fLastEntryIsDir`** flag is set to non-zero by the caller, then the full path is supplied to **`CreateDirectoryA`** to attempt to create the final directory. This call will fail, however, if the directory already exists or a file with the same name exists, but the result of this behavior is ignored by the function.

4.3.5 Secure File Delete

The Lazarus Group goes to great lengths to destroy content, not only in their destructive malware but also in their RATs and installers as well. Securely deleting a file (or files) from a victim's machine has practical applications when viewed from the perspective of forensic recovery. When a file is deleted using standard operating system deletion functions, the file's contents remain on the hard drive but the file's space is marked as available. For a recently deleted file, a forensic analysis has a high probability of recovering the original file. A secure deletion function, however, not only deletes the file by marking the space available, but it also overwrites the data on the disk in order to destroy the content.

Many of the families within the Lazarus Group's collection use a similar methodology for the destruction of files on a victim's computer. While there are variations on a theme when it comes to destroying files, the most common method that the Lazarus Group employs to ensure a file is securely deleted is as follows:

1. Generate a buffer of random data
2. Overwrite the targeted file with the random data until the entirety of the file has been replaced
3. Rename the file with random letters (replacing each letter in the filename, without adding additional letters)
4. Delete the file

Some variations observed in Lazarus Group families include replacing the file name with `TMP`{number}`.tmp` and changing the size of the file (via _ **`chsize`** or **`SetEndOfFile`**) to 0.

4.3.6 Target File Identification

```
BOOL IsTargetFileExtension(wchar _ t *Str1)
{
  return Str1
      && (!wcsnicmp(Str1, L".doc", 4u)
       || !wcsnicmp(Str1, L".docx", 5u)
       || !wcsnicmp(Str1, L".docm", 4u)
       || !wcsnicmp(Str1, L".wpd", 4u)
       || !wcsnicmp(Str1, L".wpx", 4u)
       || !wcsnicmp(Str1, L".wri", 4u)
       || !wcsnicmp(Str1, L".xls", 4u)
       || !wcsnicmp(Str1, L".xlsx", 5u)
       || !wcsnicmp(Str1, L".mdb", 4u)
       || !wcsnicmp(Str1, L".ppt", 4u)
       || !wcsnicmp(Str1, L".pptx", 5u)
       || !wcsnicmp(Str1, L".pdf", 4u)
       || !wcsnicmp(Str1, L".hwp", 4u)
       || !wcsnicmp(Str1, L".hwp", 4u)
       || !wcsnicmp(Str1, L".hna", 4u)
       || !wcsnicmp(Str1, L".gul", 4u)
       || !wcsnicmp(Str1, L".kwp", 4u)
       || !wcsnicmp(Str1, L".eml", 4u)
       || !wcsnicmp(Str1, L".pst", 4u)
       || !wcsnicmp(Str1, L".alz", 4u)
       || !wcsnicmp(Str1, L".gho", 4u)
       || !wcsnicmp(Str1, L".rar", 4u)
       || !wcsnicmp(Str1, L".php", 4u)
       || !wcsnicmp(Str1, L".asp", 4u)
       || !wcsnicmp(Str1, L".aspx", 5u)
       || !wcsnicmp(Str1, L".jsp", 4u)
       || !wcsnicmp(Str1, L".java", 4u)
       || !wcsnicmp(Str1, L".cpp", 5u)
       || !wcsnicmp(Str1, L".h", 5u)
       || !wcsnicmp(Str1, L".c", 5u)
       || !wcsnicmp(Str1, L".zip", 4u));
}
```

Several of the destructive malware samples identified during Operation Blockbuster use a common function to identify target files by their extension. The function is straightforward in its operation: it takes a single wide character string **`(wchar _ t*)`** and performs a series of string compares to determine if the supplied string matches any of the targeted file extensions. While this may not seem like a particularly strong artifact to tie together multiple malware families, the function has two distinct characteristics that make it a suitable artifact for cross-family correlation, both shown in the source code in Figure 6-17. First, the order of the extensions is constant. Second, the function has a typo where the file extension **`.hwp`** is checked for twice in a row.

Figure 6-17: Common Target File Extension Identification Function with Duplicate Entries for **`.hwp`**

CHAPTER

FIVE

5. CONCLUSION

Using the hashes of the malware used in the November 2014 SPE attack, Novetta was able to identify more than 45 malware families due to shared code, encryption keys, and other features across a diverse set of tools. This set of malware has been attributed to a threat actor we have dubbed the Lazarus Group. The Lazarus Group's malware variants have been under active development since at least 2009 and can be tied to publicly related attacks as early as 2007.

Despite the fact that many of the malware variants are not as sophisticated as many tools attributed to other APT groups, the corpus of malware used by the Lazarus Group is extremely effective and, in multiple cases, responsible for targeted cyber espionage, data theft, and destructive attacks. Notably, as the attack against SPE and other targets have shown, efficient, long-term, and destructive cyber attacks can be orchestrated and executed by this group.

5. Conclusion (continued)

In Operation Blockbuster, Novetta and industry partners have begun working together to understand and devise ways to degrade the Lazarus Group's malware toolset, eroding the group's ability to use these tools for further harm.

While no effort can completely halt malicious operations, Novetta believes that these efforts can help cause significant disruption and raise operating costs for adversaries, in addition to profiling groups that have relied on secrecy for much of their success.

It is our hope that private industry will not only continue to illuminate various threat actors' toolsets and operations, but also work with other industry partners and law enforcement agencies as able to affect positive change on the safety of network environments worldwide.

5.1 Remediation Suggestions

Given the nature of the Lazarus Groups tool set and its well-resourced operations, this section of the report is not intended to provide in-depth remediation suggestions for every possible scenario and environment. Rather, we highlight general methods that can be of use to organizations who are concerned about mitigating these types of general threats. For organizations who feel like their own internal cyber security capabilities are immature or non-existent, Mitre has released a high quality book on this topic[60] for public consumption.

With the help of operation partners, Novetta has pushed AV, IDS and YARA signatures to identify associated Lazarus Group tools and traffic. In addition to checking against these signatures, an up-to-date antivirus tool reporting to a central, monitored location is highly recommended. Other freely available tools, such as Microsoft's EMET, are also valuable defensive measures in conjunction with following suggestions for securing endpoints, servers, and network infrastructure.

On top of the provided signature-based detections, scrutinizing network traffic, and storing raw network traffic (i.e., pcap) for as long as is economically feasible can function as a tremendous aid in the investigation of alerts or identification of anomalous or malicious traffic. When considering the SPE attack, there is clear evidence to suggest that the attackers had access to corporate networks and were exfiltrating data long before the destructive malware was downloaded and executed.

Network segregation, i.e., preventing workstations from talking to each other, could also help mitigate attacks; malware used by the Lazarus Group takes advantage of such configurations between machines for lateral movement to spread within the network, deploying malware that spreads via P2P or via SMB bruteforcing using built-in Windows shares. Similarly, remote access to machines should be restricted and only allowed on a case-by-case basis where needed. Administrator-level permissions should also be restricted, as attackers with an initial foothold into a system can use or elevate to administrator privileges to gain access to entire networks. Wherever possible, two-factor authentication is strongly recommended as well as proper ageing of account passwords and strong password complexity requirements and associated testing.

Like many other attackers, the Lazarus Group appears to rely on social engineering as an initial attack vector. Educating employees as to the dangers of spear phishing both in email as well as a its use in a social media context is crucially important, as an attacker can easily gain access to sensitive information that can be used to social engineer remote access

60 "Ten Strategies of a World-Class Cybersecurity Operations Center." MITRE. 2014. https://www.mitre.org/sites/default/files/publications/pr-13-1028-mitre-10-strategies-cyber-ops-center.pdf

or in the worst case gain direct remote access to a target network. One way to attempt to minimize these types of attacks is to ensure that end users are applying software updates and patches to their home machines prior to connecting via VPN, as well as mount internal awareness campaigns that promote patching as well as suspicion of links and files sent via social media.

In addition to the above steps, regular backups of servers are recommended including continual testing and verification of your backup process and DRP plans can aid in recovery from failures or DDoS attacks. Furthermore, as the Lazarus Group does not solely concentrate on destructive attacks, but also cyber espionage and data theft, encryption of sensitive data, including emails, is highly recommended.

It is worth noting that automated solutions, tools, and other procedures outlined above and elsewhere are no substitute for having a well-funded and dedicated security team. As breaches have become the new normal, with increasing fallout, a thorough security policy and empowered team is necessary.

For more information, including guidelines for restoration of targeted systems, see the National Security Agency report "Defensive Best Practices for Destructive Malware"[61] and US-CERT's "Handling Destructive Malware."[62]

5.2 Additional Resources and Reporting

Novetta has released additional technical reports detailing the capabilities of identified Lazarus Group malware, detailing the RATs and attack staging and content distribution tools, the data exfiltration tools,the destructive malware "wipers" and DDoS bots, other identified network tools, and the installers, uninstallers, loaders.

YARA Rules

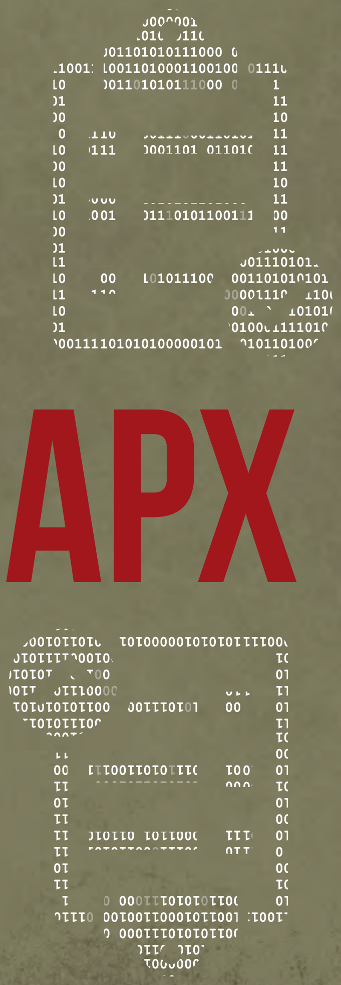
<link to microsite of yara rules>

Hashes

<link to microsite listing of hashes>

61 "Defensive Best Practices for Destructive Malware." National Security Agency/Central Security Service. January 16, 2015. https://www.nsa.gov/ia/_files/factsheets/Defending_Against_Destructive_Malware.pdf

62 "Handling Destructive Malware." US-CERT. November 4, 2013. https://www.us-cert.gov/ncas/tips/ST13-003



6. APPENDIX

The following table expands on the evidence shown earlier in the report, including further notes on the malware variants. The appendix table depicts the Lazarus Group code relationships and detections to further demonstrate the connection between variants observed in the SPE attacks, and other earlier publicly reported attacks. For more information on the code relationships, contact **trig@novetta.com.**

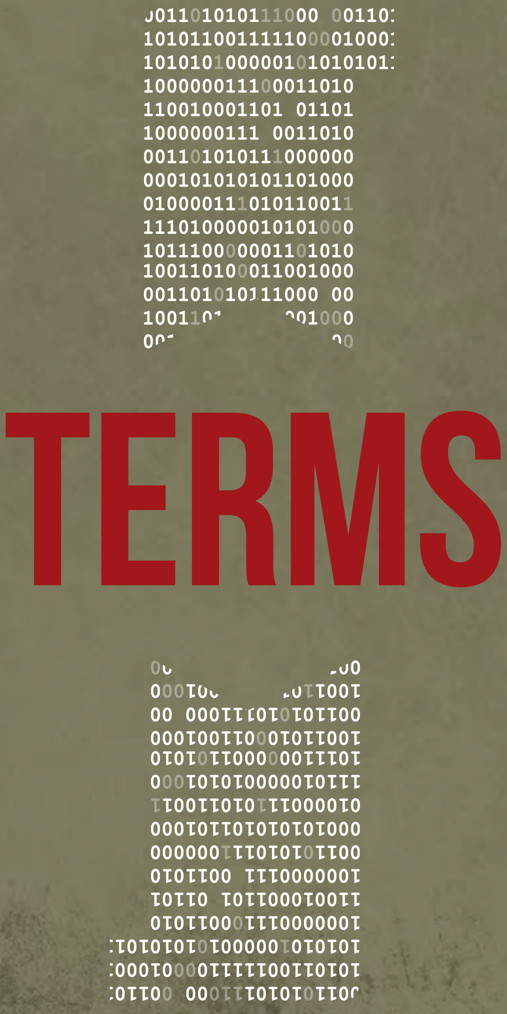
6. Appendix (continued)

MALWARE VARIANT	LAZARUS CODE RELATIONSHIPS	OTHER NOTES	OTHER AV DETECTIONS/ NAMES
DeltaAlfa	N/A	Used in Ten Days of Rain attacks, identified as part of Operation Troy, dropped by IndiaGolf	DDoS-KSig, Fibedol, Koredos
DeltaBravo	Suicide Script	Dropped by IndiaFoxtrot	
DeltaCharlie	RSA Transform, Space-Dot Encoding, Dynamic API Loading		
HotelAlfa	N/A	GOP server in the SPE attack	Destover, DestoverServ, Nukesped, NukespedServ
IndiaAlfa	Suicide Script	Installs RomeoAlfa	Escad, Destover "Messagethread," Destover "BasicHwp," Mdrop
IndiaBravo	Dynamic API Loading, Basic XOR with Constant 0xA7, Space-Dot Encoding	Installs RomeoBravo, RomeoCharlie, and PapaAlfa	Escad, Destover "Messagethread"
IndiaCharlie	Directory Hierarchy Verification and Generation, Suicide Script	Installs RomeoFoxtrot	
IndiaDelta	Dynamic API Loading	Installs LimaAlfa and WhiskeyCharlie	
IndiaEcho	Suicide Script	Installs LimaBravo, RomeoGolf, and IndiaBravo-RomeoBravo	Escad
IndiaFoxtrot	Dynamic API Loading, Space-Dot Encoding, DNSCALC-style Encoding	Installs RomeoWhiskey	Escad, Winsec, Destover, Gamarue
IndiaGolf	Directory Hierarchy Verification and Generation, Suicide Script	Installs RomeoMike and DeltaAlfa, Loads RomeoGolf, identified in Operation Troy	Koredos, DDoS-KSig, QDDOS, Fibebol
IndiaHotel	N/A	Installs WhiskeyBravo and RomeoLima, identified in Operation Troy	Wiper.C
IndiaJuliett	N/A	Installs SierraJuliett-MikeOne and SierraBravo, IndiaJuliett signatures matched several Operation Troy hashes	Escad, Joanap.d

MALWARE VARIANT	LAZARUS CODE RELATIONSHIPS	OTHER NOTES	OTHER AV DETECTIONS/ NAMES
IndiaKilo	N/A	Dropped by SierraJuliett-MikeOne during campaign	
IndiaWhiskey	Dynamic API Loading, Space-Dot Encoding, Suicide Script	Installs RomeoWhiskey	Escad, KorDllbot backdoor service installer
UniformAlfa	Suicide Script	Uninstalls RomeoBravo	
UniformJuliett	Directory Hierarchy Verification and Generation, Suicide Script	Uninstalls SierraJuliett-MikeOne	
KiloAlfa	Suicide Script, DNSCALC-style encoding	It is believed that RomeoDelta is responsible for collecting the keystroke log files generated by KiloAlfa, KiloAlfa's suicide script contains similar strings as that of Dozer, the malware used in a July 2009 DDoS attack	
LimaAlfa	Secure File Delete, Suicide Script	Loads WhiskeyCharlie	
LimaBravo	N/A	Loads RomeoGolf	BZub
LimaCharlie	Space-Dot Encoding, Dynamic API Loading	Loads RomeoHotel	Escad
LimaDelta	Suicide Script	Loads IndiaGolf, identified in Operation Troy	Koredos, DDoS-Ksig, QDDOS, Fibebol, Npkon
PapaAlfa	Space-Dot Encoding, Dynamic API Loading, Opening Windows Firewall Method	Acts as a proxy for traffic specific to the Romeo-CoreOne based RATs	Escad
RomeoAlfa	FakeTLS, Caracachs	Shares a common core, Romeo-CoreOne	Escad, Destover, NukeSped
RomeoBravo	DNSCALC-style Encoding	Shares a common core, Romeo-CoreOne	Escad
RomeoCharlie	DNSCALC-style Encoding, Opening Windows Firewall Method	Shares a common core, Romeo-CoreOne	Escad
RomeoDelta	Dynamic API Loading, Space-Dot Encoding, DNSCALC-style Encoding	Uses the same CRSA code (specifically the custom RSATransform function) found in SierraJuliet-MikeOne and RomeoWhiskey	Escad, Destover "Windows updatetracing," NukeSped
RomeoEcho	DNSCALC-style Encoding, Datagram Format		Escad, Darpapox
RomeoFoxtrot	Common Send/Recv. Functions	Dropped by IndiaCharlie	
RomeoGolf	Fake TLS	Loaded by LimaBravo, Dropped by IndiaEcho	
RomeoHotel	FakeTLS, Caracachs	Shares a common core, Romeo-CoreOne	Escad

MALWARE VARIANT	LAZARUS CODE RELATIONSHIPS	OTHER NOTES	OTHER AV DETECTIONS/ NAMES
RomeoMike	N/A	The C2 component seen in the "Ten Days of Rain" attacks	
RomeoNovember	DNSCALC-style Encoding	Shares a common core, Romeo-CoreOne	Escad
RomeoWhiskey	Socket Disconnect, Common Network Data Transmission and Receiving Function, Datagram Format	One variant uses the same public key found in SierraJuliett-MikeOne	KillFW, Escad, Winsec, KorDllbot, KillFW, Destover
SierraAlfa		Built specifically for the SPE attack, responsible for the distribution and activation of WhiskeyAlfa	Destover, NukeSped, Escad, Wiper
SierraBravo	Suicide Script	Uses the same public key as SierraJuliett-MikeOne as well as one same library	Escad, Brambul, Joanap.c, Joanap.d
SierraCharlie	Suicide Script	Shares a certificate with SierraJuliett-MikeOne, uses the same random IP generator as SierraBravo	Escad
SierraJuliett-MikeOne	N/A	Shares a public key with SierraBravo, RomeoWhiskey; shares a certificate with SierraCharlie; loads TangoCharlie	Escad, Joanap
SierraJuliett-MikeTwo	Caracachs		
TangoAlfa	Opening Windows Firewall Method	Network Tester	
TangoBravo	Suicide Script	Domain Redirector, identified in Operation Troy	Koredos
TangoCharlie	SierraJuliett-MikeOne payload	Windows Firewall Disabler	
TangoDelta	Suicide Script	Antivirus Suite Killer	Escad, Destover, NukeSped, Wiper
WhiskeyAlfa	Suicide Script	One variant associated with the SPE attack also drops an additional malware family, HotelAlfa. Another variant associated with the SPE attack includes a spreading mechanism specific to SPE infrastructure and an option to drop TangoDelta	Destover, Escad, NukeSped, Wiper, KillFiles
WhiskeyBravo	Shares code with other malware families during the file destructive process	Also profiled by McAfee's analysis of the "Ten Days of Rain" incident[1]	KillFiles, DDoS-KSig, Fibebol, Koredos, QDDOS
WhiskeyCharlie	Secure File Delete		
WhiskeyDelta	DNSCALC-style Encoding		KillDisk, HDDKill, MBRKiller, KillMBR, Basutra

1 Ten Days of Rain: Expert analysis of distributed denial-of-service attacks targeting South Korea." McAfee. 2011. http://www.mcafee.com/us/resources/white-papers/wp-10-days-of-rain.pdf



7. GLOSSARY OF TERMS

Operation Blockbuster frequently uses technical terminology and abbreviations that may be unfamiliar to certain audiences. Therefore, we have compiled the following glossary of terms to serve as a reference for readers. For further information on terminology or report details, please contact **trig@novetta.com**.

7. Glossary

API (Application Programming Interface)

Set of routines and tools for creating software and applications.

C2 (Command and Control)

Infrastructure used to control malware.

CNO (Computer Network Operations)

Intentional actions taken to improve networks and user compatibility.

DDoS Attack (Distributed Denial-of-Service)

A type of attack where many compromised systems target a single system making it unavailable to the intended user.

DNSCALC

Malware used by several APT groups and first profiled in 2010. Known for the use of DNS lookups for domain names that would return specific IP addresses used to calculate the listening port number for the C2 server.

Guardians of Peace (GOP)

The hacker group who claimed to use destructive malware to attack Sony Pictures Entertainment by releasing confidential information.

Hangul Word Processor (HWP)

Word processing application created by the South Korean company Hancom Inc.

IDS (Intrusion Detection Signatures)

A pattern that allows identification of signatures.

Installers

Software that allows applications to run on a computer.

International Civil Aviation Organization (ICAO)

UN organization that promotes security and aviation regulation.

JoongAng Attack

The June 2012 attack on conservative media organization JoongAng carried out by hacker group IsOne using two North Korean servers and 17 servers in 10 other countries.

Keylogger

Someone who tracks and notes each keystroke made on a computer, usually without permission from the user.

Master Boot Record (MBR)

The information located in the first sector of a hard disk. This identifies where the system is located so that it can be loaded into the main storage.

Microsoft EMET (Enhanced Mitigation Experience Toolkit)

A tool that helps prevent software from being exploited by hackers.

P2P (Peer-to-Peer)

An application that distributes tasks between peers.

PDB (Program Database) Path

A path for storing data about how to identify and remove information from a program.

Proxy Trojan

A type of Trojan designed to use the victim's computer as a proxy server. This allows the attacker to commit illegal activities from a separate host.

RATs (Remote Access Trojans)

A malware program that includes an entry point for administrative control over a computer. These are usually invisible to users and are downloaded through platforms such as online games and email attachments.

RSA (Rivest, Shamir, and Adelman)

An algorithm developed to better factor large numbers.

SMB (Server Message Block)

Used for enabling shared access to files between users on a network.

Sony Pictures Entertainment (SPE)

An American Entertainment Incorporation and a supplementary piece of media conglomerate Sony.

Spreaders

Those who try to cause other computers to become infected with viruses.

Ten Days of Rain Attacks

Attacks that targeted South Korea's media, financial, and critical infrastructure targets.

TLS (Transport Layer Security)

Protocols created to provide communications security over a network.

Totem

An open-source Novetta developed framework for large-scale file analysis and triage.

TTPs (Tools, Techniques and Processes)

The extensive and varied toolset which effectively combines a number of methods for delivering additional malicious tools, exfiltrating data, and launching destructive attacks.

Uninstallers

Various utility software that is created to remove parts from a computer.

VPN (Virtual Private Network)

A network that is created by using the internet to connect to a private network as a platform for transporting data.

Wipers

A security measure taken to completely erase the data from a hard disk.

YARA

A tool used by researchers to identify malware samples based on various patterns and rules.

NOVETTA

McLean, Virginia – Headquarters
7921 Jones Branch Drive
5th Floor
McLean, VA 22102
Phone: (571) 282-3000
www.novetta.com

www.OperationBlockbuster.com