IIR

Internet
Infrastructure
Review

May 2020
Vol. 46

Periodic Observation Report

SOC Report

Focused Research (1)

Points to Watch when Acquiring Windows Memory Images

Focused Research (2)

Binary Program Analysis With No Prior Knowledge of Analysis Target

IIJ
Internet Initiative Japan

Internet Infrastructure Review

May 2020 Vol.46

Executive Summary

The year 2020 was to be the second time Tokyo hosted the Olympic and Paralympic Games. As preparations for the games were entering their final stages, the world was rocked by news of the novel coronavirus. The daily stream of media reports brings home just how hard it is to prevent the rapid spread of a virus like this in our modern world, where people and goods are constantly moving across the globe. Despite the efforts of WHO and government bodies around the world to stem the outbreak, the number of infections continues to rise daily.

Viral infections are not the only thing to propagate rapidly. Huge amounts of information spread through the Internet like wildfire, and we are constantly bombarded with all sorts of information about the coronavirus. That information is not always accurate. Some of it is partial to a particular perspective, some of it is geared to the interests of specific people, and some of it is even deliberately crafted to mislead. The great advances in information and communications technology we enjoy in our modern world afford us the freedom to obtain myriad information, yet at the same time, these recent world events have reaffirmed the need for individuals to verify the truthfulness of the vast sea of information available, and to think and act accordingly.

The IIR introduces the wide range of technology that IIJ researches and develops, comprising periodic observation reports that provide an outline of various data IIJ obtains through the daily operation of services, as well as focused research examining specific areas of technology.

As our periodic observation report for this edition, Chapter 1 contains our SOC Report. IIJ's SOC analyzes huge amounts of log data, including those from security devices provided as IIJ services, on its Data Analytics Platform, and we release up-to-date information on threats in blog format via wizSafe Security Signal. In this edition, we list the key security topics in 2019 that our SOC was focused on and look at three notable types of activity revealed by the Data Analytics Platform: information leaks via Elasticsearch servers, DDoS attacks, and Emotet. Our discussion incorporates IIJ's own observational data throughout, which we hope will be of keen interest.

The focused research report in Chapter 2 follows on from the forensic memory imaging discussion in Vol. 45. In that edition, we looked at acquiring memory images on Linux. This time around, we focus on Windows memory images. We go beyond simply introducing memory imaging tools and also discuss key points to be aware of when capturing and analyzing memory images. We also suggest methods of reliably dumping individual processes.

The focused research report in Chapter 3 looks at the IIJ Innovation Institute's research into binary program analysis technology that requires no special assumptions to be made about the program being analyzed. Program analysis technology is built into integrated development environments and helps to streamline development and reduce bugs. Investigating the behavior of suspected malware programs, meanwhile, at times requires us to analyze in-the-wild binary programs of unknown origins. Chapter 3 discusses research efforts to develop technology for the static analysis of binary programs that requires no prior knowledge or assumptions about the program's origins.

Through activities such as these, IIJ strives to improve and develop its services on a daily basis while maintaining the stability of the Internet. We will continue to provide a variety of services and solutions that our customers can take full advantage of as infrastructure for their corporate activities.

Junichi Shimagami

Mr. Shimagami is a Senior Executive Officer and the CTO of IIJ. His interest in the Internet led to him joining IIJ in September 1996. After engaging in the design and construction of the A-Bone Asia region network spearheaded by IIJ, as well as IIJ's backbone network, he was put in charge of IIJ network services. Since 2015, he has been responsible for network, cloud, and security technology across the board as CTO. In April 2017, he became chairman of the Telecom Services Association of Japan MVNO Council.

SOC Report

1.1 Introduction

IIJ launched the wizSafe security brand in 2016 and works constantly to create a world in which its customers can use the Internet safely. In our SOC Report in Vol. 38[*1] we examined the Data Analytics Platform at the core of wizSafe, and in Vol. 42[*2] we discussed threats that came to light in 2018 and new initiatives using the Data Analytics Platform. Here, we review key security topics for 2019 (Section 1.2) and discuss observations made on the Data Analytics Platform about threats related to those topics (Section 1.3).

1.2 2019 Security Topics

Key security topics that our SOC focused on in 2019 are summarized in Table 1.

*1 Internet Infrastructure Review (IIR) Vol.38 (https://www.iij.ad.jp/en/dev/iir/038.html).

*2 Internet Infrastructure Review (IIR) Vol.42 (https://www.iij.ad.jp/en/dev/iir/042.html).

Table 1: Key security topics in 2019

Month	Summary
January	**A personal information leak occurred on a file transfer service run by a Japanese internet services company due to unauthorized access by a third party. Roughly 4.8 million rows of member data were affected, and it was announced that the service would close on March 31, 2020.** "Closure of the Taku-File-Bin service (January 14, 2020) " (retrieved January 14, 2020) https://www.filesend.to/ (in Japanese)
February	**In February 2019, Japan's Ministry of Internal Affairs and Communications and the National Institute of Information and Communications Technology (NICT) launched a project called NOTICE (National Operation Towards IoT Clean Environment) to survey IoT devices, find those vulnerable to use in cyberattacks (e.g., due to weak passwords), and alert the users of those devices.** "The 'NOTICE' Project to Survey IoT Devices and to Alert Users" https://www.soumu.go.jp/main_sosiki/joho_tsusin/eng/Releases/Telecommunications/19020101.html "The 'NOTICE' Project to Survey IoT Devices and to Alert Users" https://www.nict.go.jp/en/press/2019/02/01-1.html
March	**The Coinhive service ended on March 8. The reason given was that factors such as repeated changes to cryptocurrency specifications and a decline in market value made it financially difficult to continue the service.**
March	**Servers run by a foreign PC manufacturer were subject to an APT (Advanced Persistent Threat). As a result, files containing malicious code were transmitted to some users who ran updates using the utilities bundled with the manufacturer's notebooks.** "ASUS response to the recent media reports regarding ASUS Live Update tool attack by Advanced Persistent Threat (APT) groups" http://www.asus.com/News/hqfgVUyZ6uyAyJe1
April	**It was discovered that an "ac.jp" domain (reserved for use in Japan by higher education institutions etc.) had been acquired by a non-qualified third party and that the domain had been used to host an adult website. The reported cause was inadequate checking of the registree's eligibility to register the domain. Given the need to ensure the credibility of highly public domains, the Ministry of Internal Affairs and Communications ordered that steps be taken to prevent a recurrence.** "Administrative action (order) relating to Japan Registry Services Co., Ltd.'s management of '.jp' domain names" https://www.soumu.go.jp/menu_news/s-news/01kiban04_02000152.html (in Japanese)
May	**A remote-code execution vulnerability in Remote Desktop Services, commonly known as BlueKeep, was revealed. As this was judged to have a serious impact on the spread of malware, an update was provided for end-of-life OS versions. Attacks actually using BlueKeep were also observed in November.** "CVE-2019-0708 | Remote Desktop Services Remote Code Execution Vulnerability" https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2019-0708
May	**It was announced that three IT security companies had been hacked and that confidential information including development documentation and antivirus source code may have been stolen. It was later concluded that one of the companies had not been impacted by the incident.** "Top-Tier Russian Hacking Collective Claims Breaches of Three Major Anti-Virus Companies" https://www.advanced-intel.com/post/top-tier-russian-hacking-collective-claims-breaches-of-three-major-anti-virus-companies
June	**A number of FreeBSD and Linux kernel vulnerabilities related to TCP were announced, including the vulnerability commonly known as SACK Panic (CVE-2019-11477), which could allow a kernel panic to be triggered by the receipt of deliberately crafted SACK packets.**
July	**It was announced that some accounts on a barcode-based payment service had been subject to unauthorized access and use by third parties. The reason given was inadequate restrictions against logging in on multiple devices and insufficient additional authentication, including two-step authentication. The service was terminated on September 30 in response.** "Notice of 7pay service termination, background, and response going forward" https://www.sej.co.jp/company/important/201908011502.html (in Japanese)
July	**It was announced that around 3 billion yen worth of cryptocurrency had been taken from a Japan-based cryptocurrency exchange. The funds taken were stored in "hot wallets", which are kept in online environments, and it is thought that the private keys had been stolen and used without authorization.** "(Update) Notification and Apology Regarding the Illicit Transfer of Crypto Currency at a Subsidiary of the Company (Third Report)" https://contents.xj-storage.jp/xcontents/AS08938/0bf3e2e9/7a8a/4e9f/97d5/0f0a146233de/20190802124804913s.pdf
July	**It was disclosed that an Elasticsearch (full-text search engine) database containing a Japanese automaker's internal information had been left open to unauthenticated access. The roughly 40GB of information included employees' personal information as well as information on the internal network and devices.** "Honda Motor Company leaks database with 134 million rows of employee computer data" https://rainbowtabl.es/2019/07/31/honda-motor-company-leak/
August	**Increase in attacks targeting a vulnerability in several SSL VPN products announced in April 2019 onward. Details on the vulnerability were revealed at Black Hat USA 2019 in August, and observations of PoC exploits and attacks using this vulnerability were also reported. Our SOC also observed attack traffic exploiting the Pulse Secure vulnerability (CVE-2019-11510).** "Over 14,500 Pulse Secure VPN Endpoints Vulnerable to CVE-2019-11510" https://badpackets.net/over-14500-pulse-secure-vpn-endpoints-vulnerable-to-cve-2019-11510/
September	**It was reported that an Elasticsearch (full-text search engine) database containing the information of over 20 million Ecuadorians had been left open to unauthenticated access.** "Report: Ecuadorian Breach Reveals Sensitive Personal Data" http://www.vpnmentor.com/blog/report-ecuador-leak/
September	**DDoS attacks were launched on Wikipedia, Twitch, and Blizzard servers. The attacks were staged by a botnet thought to be a Mirai variant.**
November	**JPCERT/CC issued an alert on the Emotet malware. The organization said that it had received multiple reports from late October 2019 of infections caused by Word files attached to forged emails purporting to be from actual organizations or people. And our SOC observed increased levels of such activity from end-September 2019.** "Alert Regarding Emotet Malware Infections" https://www.jpcert.or.jp/english/at/2019/at190044.html
December	**It was announced that several companies had been infected by the Emotet malware. Alerts were sent out saying that email addresses and email text saved on the infected devices may have been leaked and that people should not open attachments or URLs in suspicious emails purporting to be from any of the companies affected.**
December	**It was discovered that hard disks had been stolen from leased servers returned by a local government at the end of the lease before the data had been deleted from them. The hard disks were taken by an employee of the company hired by the leasing firm to erase the data and auctioned off on an online auction site.** "Theft of harddisks returned after lease expiry" http://www.pref.kanagawa.jp/docs/fz7/prs/r0273317.html (in Japanese)
December	**A report indicated that over 267 million user records on a foreign social networking service were left exposed on an Elasticsearch server that was publicly accessible without authentication.** "Report: 267 million Facebook users IDs and phone numbers exposed online" http://www.comparitech.com/blog/information-security/267-million-phone-numbers-exposed-online/

1.3 Observational Data

This section looks at notable activity in 2019 as revealed using the Data Analytics Platform.

1.3.1 Information Leaksfrom Externally Exposed Elasticsearch Servers

■ Elasticsearch and Information Leaks

Large-scale breaches of personal information were frequent in 2019. Particularly notable were information leaks due to poorly configured Elasticsearch (full-text search engine) servers. The security topics in Section 1.2 included three information leaks related to Elasticsearch. In addition to the cases listed there, an Elasticsearch server containing a U.S.-based cloud data management company's customer information was left externally accessible without authentication, according to a report[*3] in January 2019, and likewise for an Elasticsearch server containing information on roughly 90% of Panama citizens, per a May 2019 report[*4]. Large amounts of information were leaked in both cases, with the number of records exceeding several million and the volume of data exceeding several dozen GB.

Elasticsearch is an open-source, full-text search engine based on Apache Lucene developed primarily by Elastic[*5]. It employs parallel processing of massive datasets on distributed systems to achieve its high-speed search capabilities. This makes it a not-uncommon choice for large information banks, and is also why large amounts of information tend to be leaked when security incidents do occur. Elasticsearch provides a RESTful API and allows searches and data operations to be performed over the HTTP protocol. The default port for HTTP access is 9200/TCP.

In 2019, our SOC observed an increase in scanning traffic on port 9200/TCP that we believe indicates searches for Elasticsearch servers.

■ Observational Data

Figure 1 plots the number of 9200/TCP scans and source IP addresses observed over time on the IIJ Managed Firewall Service. The number of scans is normalized to a percentage of total 9200/TCP scans observed over the full year such that the overall total is 100%.

Figure 1 shows a noticeable rise in scans over September 21 – October 31. Scans of 9200/TCP over these 41 days accounted for roughly 23.37% of all scans during 2019, and the number of source IP addresses per day rose to a peak of 30,394. This is about 98.68 times the IP address count for January 1 (308). No Elasticsearch vulnerabilities

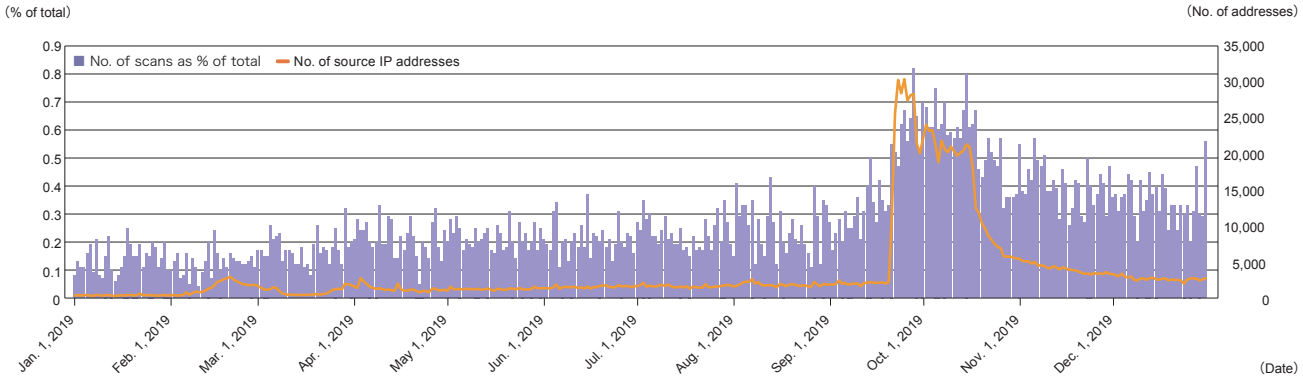

Figure 1: Scanning of 9200/TCP (January–December 2019)

*3 TechCrunch (https://techcrunch.com/2019/01/29/rubrik-data-leak/).

*4 Security Affairs, "Personally identifiable information belonging to roughly 90% of Panama citizens were exposed on a poorly configured Elasticsearch server" (https://securityaffairs.co/wordpress/85462/data-breach/panama-citizens-massive-data-leak.html).

*5 Elasticsearch (https://www.elastic.co/).

were announced around this time, and we have found no clear reason for the increased scanning activity.

On September 16, just before the spike in scans, it was reported that information on more than 20 million Ecuadorians had been exposed. While it is possible that this is what prompted the spike, we can find no clear evidence linking the two phenomena.

Temporary increases in scanning activity also occurred in mid-February and early April. The source IP count briefly rose to nearly 3,000 on February 20 and April 3. On February 19, Elastic announced an Elasticsearch vulnerability (CVE-2019-7611)[*6], and we surmise that the upticks represent searches for servers running Elasticsearch. CVE-2019-7611 is an access permission vulnerability that, if exploited, could allow information acquisition or tampering. Further, Talos, Cisco Systems' security arm, published a report on Elasticsearch attacks that occurred around this time[*7]. According to the report, attacks targeting previously revealed vulnerabilities (CVE-2014-3120, CVE-2015-1427) were observed.

The trend across the year also shows an overall rise in 9200/TCP scanning. The average number of scans per day in December rose to 0.35% from the January average of 0.14%, a roughly 2.46-fold rise, and the source IP address count increased roughly 7.02-fold, from 384.74 on average in January to 2702.10 in December. A report on observational data issued by Japan's National Police Agency (NPA)[*8] shows a similar trend and, like the Talos report, discusses attacks thought to have been targeted at CVE-2015-1427.

■ Countermeasures

Most of the widely reported information leaks involving Elasticsearch in 2019 were due to poorly configured servers allowing authentication-free access. Important basic countermeasures include using a firewall to exclude unnecessary traffic, including on 9200/TCP, if the system does not need to be accessible from the Internet and setting up appropriate authentication to only permit connections from trusted IP addresses. And as the Talos and NPA reports indicate, attacks targeting past vulnerabilities continue to be observed. When information on vulnerabilities relevant to your system is released, you need to determine what the impact on your system is and apply the relevant patches.

1.3.2 DDoS Attack Observations

IIJ observes and responds to DDoS attacks employing various methodologies. This section summarizes key topics in DDoS attacks in 2019. We start by looking at attacks detected by the IIJ DDoS Protection Service in 2019. Next, we look at attack methods that were much talked about in 2019. And finally we go over examples of damage caused by those attack methods in 2019, along with observational data.

*6 Elastic, "Security issues" (https://www.elastic.co/jp/community/security).

*7 Cisco Talos, "Cisco Talos Honeypot Analysis Reveals Rise in Attacks on Elasticsearch Clusters" (https://blog.talosintelligence.com/2019/02/cisco-talos-honeypot-analysis-reveals.html).

*8 National Police Agency, "Increase in online traffic aimed at Elasticsearch vulnerability" (in Japanese, https://www.npa.go.jp/cyberpolice/important/2019/201910021.html).

Summary of 2019 DDoS Attack Observations

DDoS attacks on Wikipedia, Twitch, and Blizzard created a stir in September 2019. Of the DDoS attacks IIJ responded to in 2019, here we summarize those detected by the IIJ DDoS Protection Service. Table 2 shows the number of attacks and traffic volume detected by the IIJ DDoS Protection Service.

Of the attacks in Table 2, the SYN Flood and SYN/ACK attacks use TCP, and the UDP Amplification and UDP Flood attacks use UDP. A number of application protocols are used in UDP Amplification attacks, including DNS, NTP, and LDAP.

Table 2 shows the daily average number of attacks for each month. No month in 2019 was a particular standout for DDoS attacks. May recorded the highest number of packets per second, and the longest attack occurred in January. The maximum number of packets was relatively large in May, July, and December, but the longest attacks in those months were under one hour. UDP Amplification attacks using LDAP and DNS feature prominently in the maximum traffic and maximum attack duration listings.

Key DDoS Attack Topics for 2019

A number of new methodologies suited to DDoS attacks other than those appearing in Table 2 also popped up in 2019. Three keywords stood out on the DDoS landscape in 2019.

- Web Services Dynamic Discovery (WSD)
- Apple Remote Management Service (ARMS)
- SYN/ACK reflection

The first, WSD, is a protocol that uses the Simple Object Access Protocol (SOAP) to locate services and enable data exchanges in specific network ranges. It uses port 3702/UDP, and it is known to be used on printers and PCs that run on Windows Vista and up. The possibility of DDoS attacks using this protocol has been discussed by zeroBS GmbH[*9]. It has been observed that there are roughly 630,000 IP addresses online that respond on 3702/UDP[*10]. Our SOC observed an increase in 3702/UDP scanning activity in August 2019[*11]. Figure 2 shows scanning activity on this port observed at the SOC in 2019. Note that the

Table 2: Summary of Observational Data on DDoS in 2019

Month	No. of incidents (daily avg.)	Approx. max. packets/sec. (x10,000)	Maximum traffic		Maximum attack duration	
			Bandwidth	Method	Duration (h:mm)	Method
1	13.58	~179	17.38Gbps	DNS Amplification	3:20	SYN Flood
2	15.75	~284	27.89Gbps	LDAP Amplification	1:18	LDAP Amplification
3	14.00	~652	19.30Gbps	SSDP Amplification	2:32	SSDP Amplification
4	22.96	~97	9.21Gbps	DNS Amplification	0:41	DNS Amplification
5	16.16	~886	39.29Gbps	LDAP Amplification	0:41	DNS Amplification
6	10.93	~148	8.11Gbps	SSDP Amplification & SYN/ACK reflection	0:30	SSDP Amplification & SYN/ACK reflection
7	16.41	~738	75.67Gbps	DNS Amplification	0:38	NTP Amplification
8	18.10	~91	8.77Gbps	LDAP & DNS Amplification	1:35	UDP Flood
9	19.20	~130	11.71Gbps	LDAP & DNS Amplification	0:43	NTP Amplification
10	22.09	~310	23.09Gbps	Amplification: LDAP, DNS, NTP, etc.	1:56	LDAP Amplification
11	13.36	~70	8.24Gbps	UDP Flood	0:25	UDP Flood
12	10.38	~607	61.34Gbps	LDAP & DNS Amplification	0:38	NTP Amplification

*9 zeroBS, "Analysing the DDOS-Threat-Landscape, Part 1: UDP Amplification/Reflection" (https://zero.bs/analysing-the-ddos-threat-landscape-part-1-udp-amplificationreflection.html).

*10 zeroBS, "New DDoS Attack-Vector via WS-Discovery/SOAPoverUDP, Port 3702" (https://zero.bs/new-ddos-attack-vector-via-ws-discoverysoapoverudp-port-3702.html).

*11 wizSafe, "wizSafe Security Signal August 2019 Observational Report" (in Japanese: https://wizsafe.iij.ad.jp/2019/09/746/).

number of scans is normalized to a percentage of total 3702/UDP scans observed over the full year such that the total is 100%.

Figure 2 shows that scans on this port increased from around August 13. The number of source IP addresses scanning the port also rose from August 19 through end-August. The observations in Figure 2 generally match those in BinaryEdge reports. The reason for the increase in scanning on the port on February 17 is unclear, but Baidu, Inc. reported on February 19 that a DDoS attack using WS-Discovery had occurred[*12]. Hence, it appears that DDoS attacks exploiting WS-Discovery had been in use since at least February. But it was September 2019 when they came into focus in Japan. And a US-Cert document on UDP Amplification Factors was updated in December to cite a September article on this type of attack[*13]. So it seems that it was actually a few months after WS-Discovery was first used in attacks that attackers started to use the protocol for DDoS attacks in earnest.

The second keyword, ARMS, is a service used on Apple Remote Desktop (ARD). ARD is an application for remotely controlling macOS devices. ARMS receives commands from the control console via 3283/UDP. It was found that there are around 40,000 devices on which ARMS is reachable via the Internet[*14]. Figure 3 shows scanning activity on the port observed by our SOC in 2019. Note that the number of scans is normalized to a percentage of total 3283/UDP scans observed over the full year such that the total is 100%.

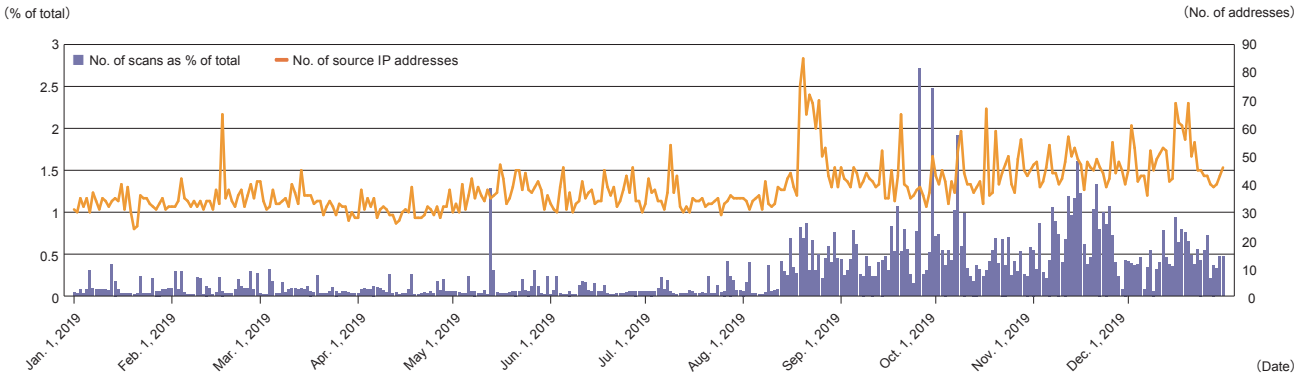

Figure 2: Scanning of 3702/UDP and Number of Source IP Addresses (Jan.–Dec. 2019)

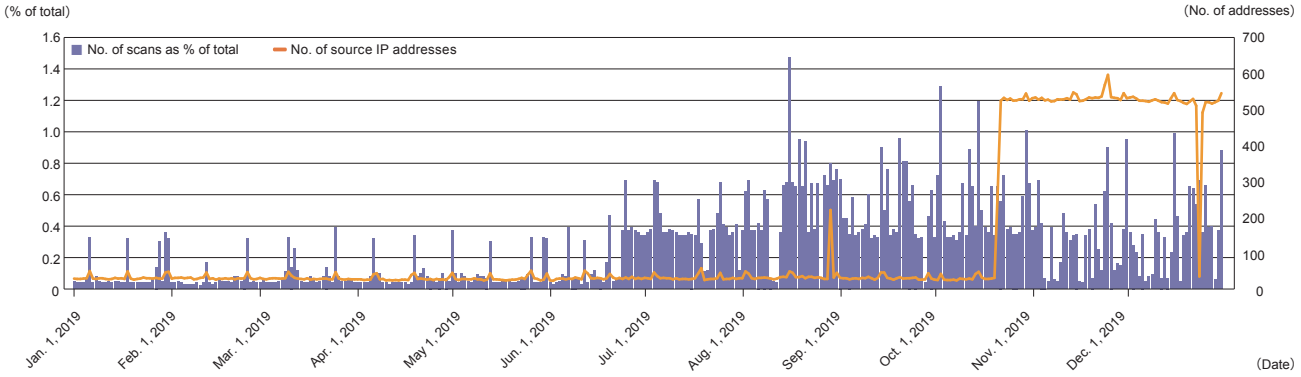

Figure 3: Scanning of 3283/UDP and Number of Source IP Addresses (Jan.–Dec. 2019)

*12 Baidu Security Index,"基于ONVIF协议的物联网设备参与DDoS反射攻击"(in Chinese, https://bsi.baidu.com/article/detail/128).
*13 CISA, "Alert (TA14-017A)" (https://www.us-cert.gov/ncas/alerts/TA14-017A).
*14 ZDNet, "macOS systems abused in DDoS attacks" (https://www.zdnet.com/article/macos-systems-abused-in-ddos-attacks/).

Figure 3 indicates that scans of the port increased from around June 24. And the number of source IP addresses scanning the port increased from around October 22. So it is evident that scanning activity was increasing a few days before the release of the NetScout Systems, Inc. report[*15].

Our third keyword is SYN/ACK reflection attacks. This attack takes place in the TCP three-way handshake. SYN packets with a spoofed source address are sent to many addresses simultaneously, thereby effectively recruiting the resulting SYN/ACK packet responses to perform a DDoS attack on the source address. Figure 4 gives an overview of a SYN/ACK reflection attack.

Below, we describe the flow of events from the launch of a SYN/ACK reflection attack through to the damage it inflicts on the victim. Refer to Figure 4 as you read through.

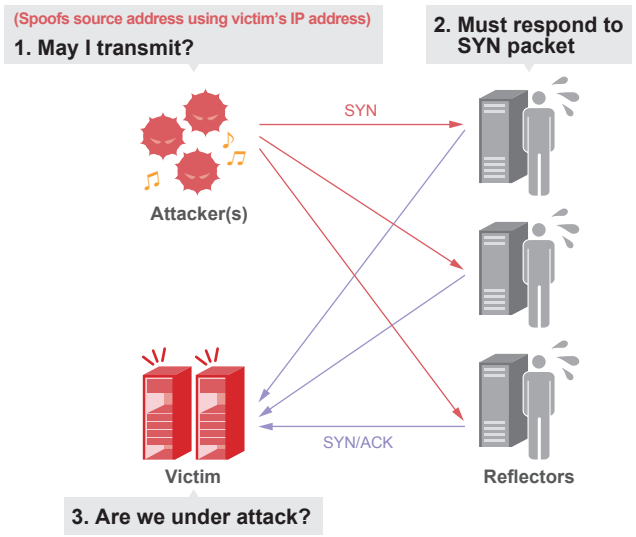

Figure 4: Overview of a SYN/ACK Reflection Attack

1. To generate the SYN/ACK packets used in the attack, the attacker spoofs the source address to match the attack target and sends the SYN packets with that spoofed source address to the reflectors.
2. During the three-way handshake, the reflectors send SYN/ACK packets in response to those SYN packets.
3. Because the source address on the SYN packets is spoofed, the SYN/ACK packet responses from the reflectors are delivered to the attack target's IP address, thus consummating the attack.

This type of attack was observed by our SOC in 2018 and is explained in Section "1.2.2 SYN/ACK Reflection Attack" of Vol. 42[*16]. This SYN/ACK reflection attack uses a TCP Amplification attack technique that was known around 2006[*17]. In 2014, researchers discovered devices on the Internet with protocol implementations that result in more SYN/ACK packets, RST packets, or PSH packets being retransmitted than is common[*18]. It is not clear whether the devices found in 2014 are actually being used, but the attack principles are the same. At our SOC, TCP Amplification attacks that use SYN/ACK packets are termed SYN/ACK reflection attacks, and they were observed frequently from around July through November.

A distinctive feature of these three attack methods that featured prominently in 2019 is that they spoof the packet source address to match the target and recruit reflectors to mount the DDoS attacks. DDoS attacks like this are called Distributed Reflection Denial of Service (DRDoS). To perform a DRDoS attack, the attacker first looks for hosts and ports that can be used as reflectors and attempts to exploit them. So if ports that can be used for DRDoS are made accessible to anyone on the Internet, they are at risk of being recruited as reflectors in DRDoS attacks. With DRDoS attacks like WSD and ARMS, countermeasures are needed not only on the sender and target but also on the reflectors. In DRDoS attacks, the administrators of the reflector servers are not being targeted, but they are unintentionally participating in attacks on the targeted servers or networks. So it is important to make sure you do not unnecessarily leave

*15 NETSCOUT, "A Call to ARMS: Apple Remote Management Service UDP Reflection/Amplification DDoS Attacks" (https://www.netscout.com/blog/asert/call-arms-apple-remote-management-service-udp).
*16 Internet Infrastructure Review (IIR) Vol. 42 (https://www.iij.ad.jp/en/dev/iir/042.html).
*17 RFC 4732, "Internet Denial-of-Service Considerations" (https://tools.ietf.org/html/rfc4732#section-3.1).
*18 USENIX, "Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks" (https://www.usenix.org/system/files/conference/woot14/woot14-kuhrer.pdf).

ports open on the Internet and configure hosts to allow only the intended access. Not only does this reduce unauthorized access, it also helps reduce the number of reflectors available for DDoS attacks and thus makes it more likely that we can limit attackers' options for staging DDoS attacks. There is a DRDoS attack, however, for which it is not easy to restrict access to reflectors. This is the SYN/ACK reflection attack. The reason for this is explained in the next section on the SOC's observations.

Our SOC's Observations

DDoS attacks using the three methods described targeted organizations and services in Japan in 2019. Here, we look at some of the more prominent DDoS attacks that occurred in Japan in 2019, together with information observed by our SOC. DDoS attacks using WSD and ARMS aimed at organizations in Japan were highlighted in a JPCERT/CC alert in October 2019[*19]. It is known that in these cases, not only were WSD and ARMS used for DDoS attacks but extortion emails demanding cryptocurrency payments were also received. Attempts apparently motivated by monetary gain and involving messages threatening to launch DDoS attacks like this are called Ransom Denial of Service (RDoS) attacks. RDoS attacks caused a stir not only in 2019 but in 2017 as well[*20]. Whether the actors behind the attacks were the same in both years is unclear, but it is at least true that DDoS attacks using WSD and ARMS, which had not been disclosed in 2017, were used in the 2019 cases. Considering this in conjunction with Figures 2 and 3, it appears that DDoS attack infrastructure is being progressively adapted to exploitable protocols.

An example of a SYN/ACK reflection attack being used in a DDoS attack aimed at companies in Japan is that listed for maximum traffic volume and attack length for June in Table 2. A key feature of SYN/ACK reflection attacks is that they use any TCP port as the reflector and thus do not exploit services tied to specific ports like WSD or ARMS. This is why ports commonly used by Web servers, such as 80/TCP and 443/TCP, are used. It is important, for example, that the content of Web servers on the Internet be accessible from anywhere if it is to be made available to a wide audience. In this scenario, the firewall will be configured to allow anyone to access the server. And as such, it will be difficult to deny access on the server side if the server is used as a reflector in a SYN/ACK reflection attack. Figure 5 shows the percentage breakdown of TCP ports used as reflectors in SYN/ACK reflection attacks observed by our SOC in 2019.

As Figure 5 shows, the TCP ports used in SYN/ACK reflection attacks are 80/TCP, 443/TCP, and 25/TCP, which is used for the Simple Mail Transfer Protocol (SMTP). These account for over 95% of the total. The "Others" slice represents many ports including 21/TCP, 22/TCP, and 587/TCP. So it is evident that the ports recruited to stage SYN/ACK reflection attacks are TCP ports that are relatively openly accessible on the Internet. As Figure 5 shows, TCP ports on which services that permit external access are running appear prone to exploitation, making it difficult to deal with SYN/ACK reflection attacks by implementing access controls on the reflectors.

Yet this is not the only challenge in dealing with SYN/ACK reflection attacks. SYN/ACK reflection attacks are tough to identify unless you basically have an overview of the entire network, encompassing all the devices, the attack target, and so on. Since a slew of SYN packets from the attacking device actually arrives at each host recruited as a reflector, the reflector host administrators are liable to conclude that a SYN Flood attack is underway. In that case, if the source address in the SYN packets is permanently blacklisted on reflector hosts, it will not be possible to reach those hosts from the

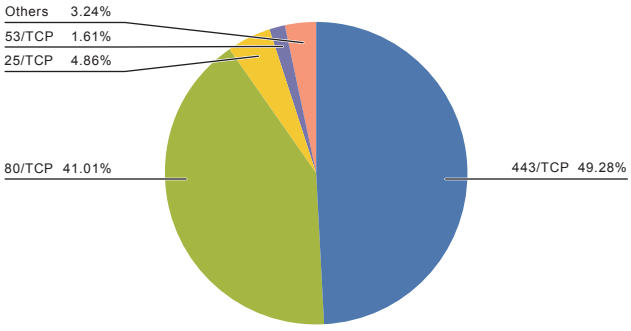

Figure 5: Breakdown of Reflector Ports Used in SYN/ACK Reflection Attacks

*19 JPCERT/CC, "Extortion emails threatening DDoS attacks and demanding cryptocurrency" (in Japanese, https://www.jpcert.or.jp/newsflash/2019103001.html).
*20 Radware, "Fancy Bear DDoS for Ransom" (https://security.radware.com/ddos-threats-attacks/threat-advisories-attack-reports/fancybear/).

attack target's IP address once the DDoS attack is over. This is the collateral damage of SYN/ACK reflection attacks.

Examples of devices in Japan being used as reflectors in SYN/ACK reflection attacks are available on our SOC's reporting site, wizSafe Security Signal[*21][*22][*23][*24]. Note that because these are SYN/ACK reflection attacks observed from the reflector's point of view, not the target's, the information does not indicate the full scale of SYN/ACK reflection attacks.

1.3.3 Emotet

■ Overview of Emotet

A malware program called Emotet, which infects hosts by exploiting emails, came to the fore in the latter half of 2019. This malware was first reported[*25] in 2014 by Joie Salvio, then working at Trend Micro. Emotet was initially active as a banking trojan targeting information from financial institutions but bit by bit morphed into a botnet. It also acquired worm capabilities by adopting a modular framework, giving it the ability to spread various malware and ransomware payloads. It has thus morphed in recent years and gained the ability to download malware (Trickbot, ZeuS, etc.) that steals not only financial institutions' information but other confidential information as well. It has also been reported that malware with information stealing capabilities downloaded by Emotet can infiltrate target systems and eventually deploy a ransomware payload called Ryuk. There have been reports of activity dubbed a triple threat[*26] involving a multistage attack in which information stolen by these malware programs is used to infiltrate target systems, on which a ransomware payload called Ryuk is then deployed. As these changes have unfolded, the range of attack targets has also shifted to public institutions and private companies.

Internationally, it was observed[*27] that C2 servers used by Emotet went inert from June 2019, but the hiatus did not last long. It was reported at the end of August 2019 that the servers had resumed activity, and from September on IIJ's email gateway service, the IIJ Secure MX Service, we detected an increase in malicious emails designed to spread Emotet infections.

Our SOC observed a lot of infection activity exploiting Microsoft Word (doc) format attachments. Subsequently, there was an increase in the number of emails representing a separate infection vector, namely that the body text contained a URL that downloads a doc file that then infects the host with Emotet.

*21 wizSafe, "wizSafe Security Signal July 2019 Observation Report" (in Japanese, https://wizsafe.iij.ad.jp/2019/08/717/).
*22 wizSafe, "Observation of DDoS attacks targeting Servers.com" (in Japanese, https://wizsafe.iij.ad.jp/2019/10/764/).
*23 wizSafe, "Examples of TCP SYN/ACK Reflection Attack Observations for October 2019" (in Japanese, https://wizsafe.iij.ad.jp/2019/12/820/).
*24 wizSafe, "Examples of TCP SYN/ACK Reflection Attack Observations for November 2019" (in Japanese, https://wizsafe.iij.ad.jp/2019/12/839/).
*25 Trend Micro, "New Banking Malware Uses Network Sniffing for Data Theft" (https://blog.trendmicro.com/trendlabs-security-intelligence/new-banking-malware-uses-network-sniffing-for-data-theft/).
*26 Cybereason, "Research by Noa Pinkas, Lior Rochberger, and Matan Zatz" (https://www.cybereason.com/blog/triple-threat-emotet-deploys-trickbot-to-steal-data-spread-ryuk-ransomware).
*27 Bleeping Computer, "Emotet Botnet Is Back, Servers Active Across the World" (https://www.bleepingcomputer.com/news/security/emotet-botnet-is-back-servers-active-across-the-world/).

Opening the Emotet-infecting doc file with Word's default settings produces a message asking you to "Enable Content", as in Figure 6. Enabling this results in a macro being executed. If Word is already configured to enable macros, the user does not see a screen like that in Figure 6 and the macro simply runs automatically. Once executed, the macro downloads Emotet from a malware distribution server, which infects your device.

Once it has infected a device, Emotet tries to make the infection persistent by copying itself to new services, configuring them to run automatically. It then steals information from the infected PC and communicates with its C2 server. The information stolen includes email text and addresses, and some of Emotet's malicious emails exploit this information to disguise themselves as replies to past emails threads. This is one factor behind Emotet's spread. As the multi-stage attack (triple threat) example demonstrates, Emotet serves as an entry point for other malware, so the type of damage it ultimately causes is likely to continue to morph ahead.

Observational Data

Below, we report on our SOC's observations on Emotet.

The stacked bar graph in Figure 7 divides attacks detected over September–December 2019 into those related to Emotet and those related other attacks. Date is on the horizontal axis. The vertical axis represents the total number of detections normalized to a percentage of total detections over the entire period, such that the overall total is 100%.

The first prominent Emotet detection in the graph is on September 27. Following that, it was also detected prominently on October 16, 17, 23, and 24. In November onward, it was detected on more days and more frequently than in the preceding months. And the detections tended to be concentrated on weekdays. Detections reached the overall peak for the period over December 3–4. This was followed by a spike on December 16, and then detections on the IIJ Secure MX Service settled down through the rest of December.

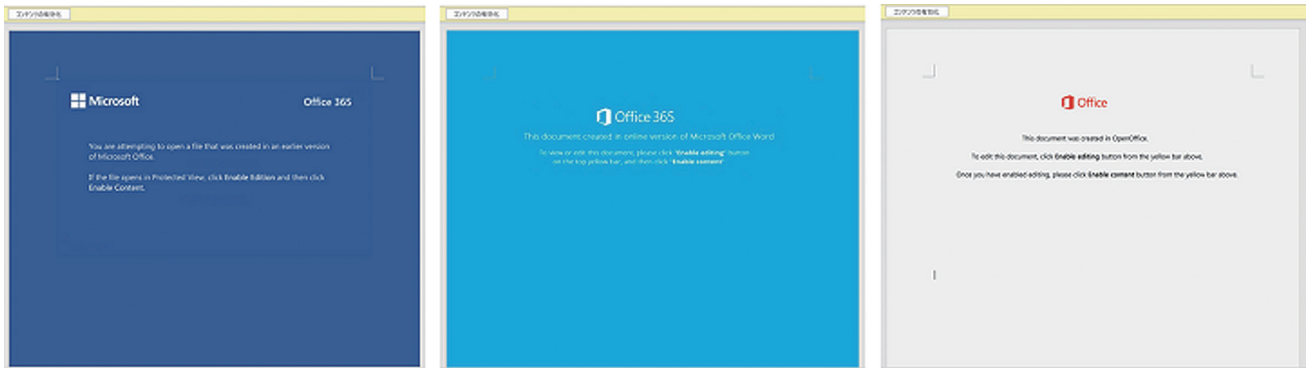

Figure 6: Examples of Screens Asking User to Enable Content

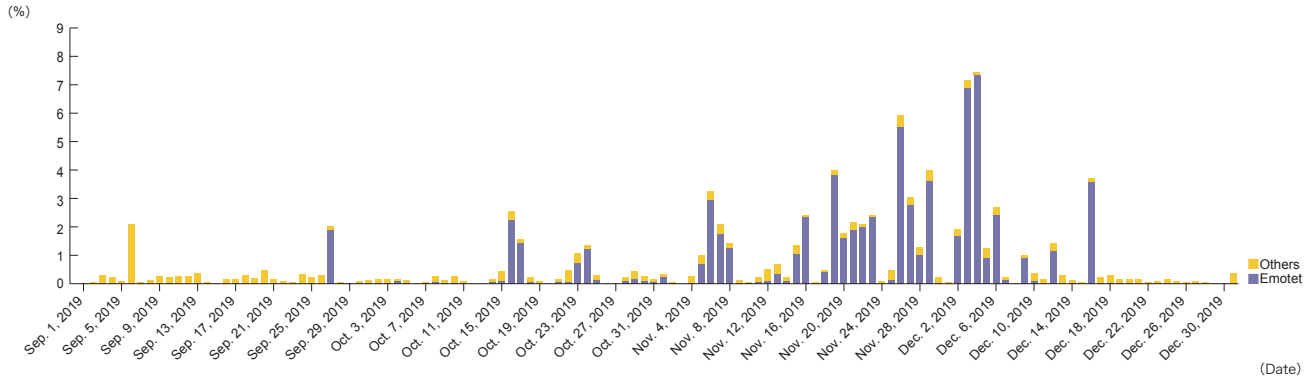

Figure 7: Malware Detections in Received Emails (Sep.–Dec. 2019)

But as if substituting for this, Emotet-related detections on the IIJ Secure Web Gateway Service then increased. In Figure 8, date is on the horizontal axis, and the vertical axis represents the number of detections normalized to a percentage of total in December 2019, such that the total is 100%.

These Emotet-related detections in Figure 8 increased for a few days starting December 17, right after the email detections in Figure 7 eased off. We have determined that this traffic represents attempts to download Emotet-infecting doc files. Japan's Information-technology Promotion Agency also issued an alert[*28] stating that Japanese emails containing links to malicious URLs that cause Emotet infections had been observed from around December 10, which matches the start of detections in Figure 8. Hence, the traffic detected in Figure 8 is likely accessing URLs in the text of emails designed to spread Emotet.

Next, of the emails thought to be Emotet propagators observed by our SOC, Table 3 summarizes those that contain Japanese text in the subject line. Note that Table 3 only shows the main examples and is not comprehensive.

As Table 3 shows, the subject lines are varied. Some are just a single word, like "Realize", "Help", or "Information", and others purport to be invoices/receipts. Also, around

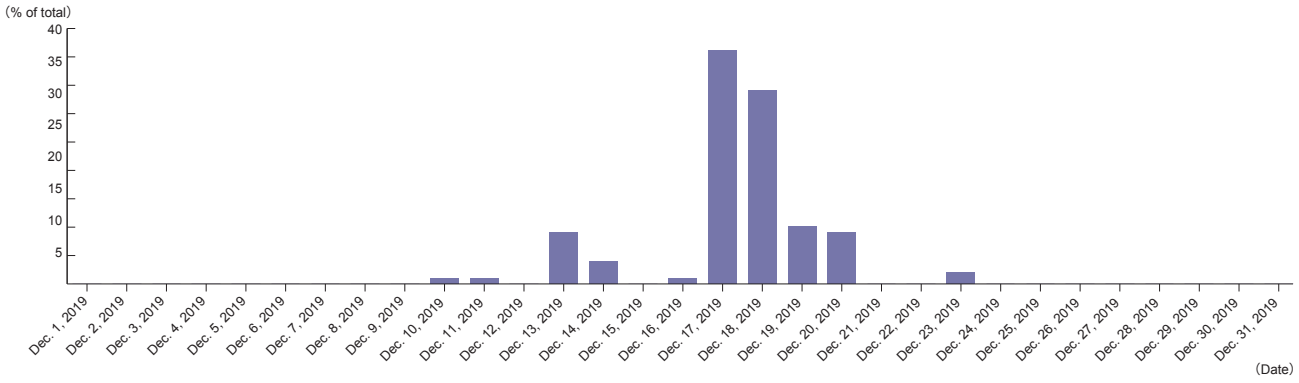

Figure 8: HEUR: Trojan.MSOffice.SAgent Detections as Percentage of Total (Dec. 2019)

Table 3: Suspicious Emails Designed to Spread Emotet with Japanese Text in Subject Line

Subject lines	Attachment filenames	Notes
December bonus	<date>.doc	<date> is the date of receipt in YYYYMMDD format Date, names of people or organizations are appended to the subject line in some cases
[Valid till 23:59 today] Renewal discount coupon issued on amazon.com	<date>_<random alphanumeric string>.doc	
Account credited		
Please issue invoice	<random alphanumeric string> <date>.doc	
Document	<random alphanumeric string>_<date>.doc	
Resending message	<random alphanumeric string>-<date>.doc	
Reminder		
Realize	Bonus payment advice.doc	
Final option	December bonus.doc	
Payment advice	Winter 2019･performance bonus payment.doc	
Help		
Information	Please send invoice <random alphanumeric string>-<date>.doc	
New version	Merry Christmas <date>.doc	
Please attach invoice		
Receipt		

*28 Information-technology Promotion Agency, "Emails designed to propagate a virus called 'Emotet'" (in Japanese, https://www.ipa.go.jp/security/announce/20191202.html#L11).

Black Friday, some subject lines tout discount coupons for online shopping, and attachment filenames contain words to match the season, like “Bonus” or “Christmas”.

■ Countermeasures

As mentioned earlier, Emotet uses information stolen from infected devices to create emails—fake replies etc.—designed to propagate its spread. This may make it difficult for recipients to judge that something is amiss or suspicious based on the sender address or email text. To prevent infections and minimize damage, you should first check your Word settings and disable automatic macro execution if it is on. It is also important not to inadvertently open any attachments or manually enable any macros contained in the attachments that you cannot vet as clean. US-Cert also states that a policy blocking emails with attachments that have filename extensions used by malware or file formats that antivirus software cannot scan is an effective way defend against entry[*29]. It also recommends the use of appropriate permission settings, sender authentication, and the like.

1.4 Conclusion

In this report, we covered prominent security incidents in Japan in 2019 and looked at a number of examples alongside our SOC’s observations. Various security threats beyond these examples are also observed everyday. It is important to properly understand the landscape and address threats, and this effort should not be limited to the incidents and events discussed in Sections 1.2 and 1.3. Some can be addressed with ACL, such the Elasticsearch issues in 1.3.1, while others can be defended against at the individual level by applying vulnerability patches and not casually enabling macros, as discussed in 1.3.3. Our SOC will continue to periodically publish information on security incidents and threats via wizSafe Security Signal (https://wizsafe.iij.ad.jp), and we hope these updates will prove useful in your ongoing security efforts.

Shun Morita
Data Analyst, Security Operations Center, Security Business Department, Advanced Security Division, IIJ

Eisei Honbu
Data Analyst, Security Operations Center, Security Business Department, Advanced Security Division, IIJ

Junya Yamaguchi
Data Analyst, Security Operations Center, Security Business Department, Advanced Security Division, IIJ

*29 CISA, “Increased Emotet Malware Activity” (https://www.us-cert.gov/ncas/current-activity/2020/01/22/increased-emotet-malware-activity).

Points to Watch when Acquiring Windows Memory Images

2.1 Acquiring Memory Images on Windows

In Vol. 45, we discussed the acquisition of forensic memory images on Linux[*1]. In this edition, we discuss the acquisition of memory images on Windows.

We use tools such as those listed in Table 1 to acquire full memory images of Windows systems. We use the Volatility Framework[*2] and Rekall Memory Forensic Framework[*3] to analyze the images.

Windows version upgrades, however, can come with changes to the memory management framework to improve security and performance. So you need to use tools compatible with the new specifications to acquire and analyze memory images. Not only do we cover memory image acquisition tools here, we also discuss some key points to watch when actually acquiring and analyzing images. We also suggest reliable ways of acquiring individual process dumps.

2.2 Points to Watch when Acquiring/Analyzing Memory Images

In this edition, we explain how to deal with the three features: the paging files, memory compression, and Virtual Secure Mode. Paging files existed in Windows prior to version 10, but other features were added in updates after the Windows 10 release.

■ Paging Files

Windows saves a process's paged-out virtual memory pages in a paging file called C:\pagefile.sys. Figure 1 shows the result of extracting notepade.exe from a Windows 10 1809 memory image using Volatility's procdump tool. This was done right after notepad.exe started, so the process dump was successful. Figure 2, meanwhile, shows the result of trying to dump notepad.exe from a memory image taken on the same system after memory usage had jumped. This was unsuccessful because of a page-out. The memory

Table 1: Examples of Memory Image Acquisition Tools

Tool	Vendor
WinPmem memory imager	https://winpmem.velocidex.com/
Comae Technologies	https://www.comae.com/
MAGNET RAM Capture	https://www.magnetforensics.com/resources/magnet-ram-capture/
Belkasoft RAM Capturer	https://belkasoft.com/ram-capturer

```
>vol.py --profile Win10x64_17763 -f "Windows 10 1809 Feb x64 en-Snapshot82.vmem" procdump --pid=4596 -D procdump
Volatility Foundation Volatility Framework 2.6.1
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xffffaf86ab950480 0x00007ff7413c0000 notepad.exe          OK: executable.4596.exe
```

Figure 1: A Successful Volatility procdump

```
>vol.py --profile Win10x64_17763 -f "Windows 10 1809 Feb x64 en-Snapshot83.vmem" procdump --pid=4596 -D procdump
Volatility Foundation Volatility Framework 2.6.1
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xffffaf86ab950480 0x00007ff7413c0000 notepad.exe          Error: ImageBaseAddress at 0x7ff7413c0000 is unavailable (possibly due to paging)
```

Figure 2: An Unsuccessful Volatility procdump

*1 Internet Infrastructure Review (IIR) Vol. 45, Focused Research (2): Acquiring Forensic Memory Images on Linux (https://www.iij.ad.jp/en/dev/iir/045.html).
*2 The Volatility Foundation (https://www.volatilityfoundation.org/).
*3 Rekall Forensics (http://www.rekall-forensic.com/).

images used for analysis in Figures 1 and 2 are VMware Workstation memory snapshots. We used these when preparing this article to make it a bit easier to acquire memory images in the state that we wanted.

The paging file stores paged-out memory data, so we want to use it in our analysis if possible. But many memory image acquisition tools do not collect this file. With WinPmem, the memory image and paging file can be obtained almost simultaneously by specifying pagefile.sys with the "-p" option (Figure 3). It can also be obtained using The Sleuth Kit or FTK Imager, but keeping the time interval between the memory image and pagefile.sys as short as possible averts discrepancies.

Obtaining pagefile.sys is pointless if the memory image analysis tool does not support it. Rekall can analyze memory images and pagefile.sys seamlessly as a single, complete memory image. Immediately after the outcome in Figure 2, we used the command in Figure 3 to acquire a memory image and pagefile.sys using WinPmem, and then using Rekall's procdump plugin, we were able to dump the notebook.exe process from those files (Figure 4). A look at the first part of the file shows that it is an MZ header (the procdump plugin dumps the specified process in PE format).

Volatility 2, on the other hand, cannot analyze pagefile.sys files like Rekall. Presentation slides released for OSDFCon 2019[*4], however, contain hints that the currently in-development Volatility 3 will support paging files as well as memory compression as described below, so it looks like Volatility will support paging file analysis in future.

The WinPmem version used in Figure 3 is 2.1 post4. The latest WinPmem as of this writing (Feb. 2020) is 3.3 rc3, but analyzing the generated AFF4 file with Rekall produced the error in Figure 6. We have not inspected all of the relevant

```
>winpmem-2.1.post4.exe -p c:\pagefile.sys -o memdump.aff4
```

Figure 3: Acquiring a Memory Image and pagefile.sys with WinPmem

```
$ rekal -f memdump.aff4 procdump --proc_regex="notepad*" --dump_dir="./"
Webconsole disabled: cannot import name 'webconsole_plugin'
                    _EPROCESS                       Filename
------------------------------------------------------ --------
0xaf86ab950480 notepad.exe                    4596 executable.notepad.exe_4596.exe
```

Figure 4: Running procdump with pagefile.sys

```
$ hexdump -C executable.notepad.exe_4596.exe | head -10
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ..............|
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 f8 00 00 00  |................|
00000040  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  |........!..L.!Th|
00000050  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  |is program canno|
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  |mode....$.......|
00000080  15 48 94 65 51 29 fa 36  51 29 fa 36 51 29 fa 36  |.H.eQ).6Q).6Q).6|
00000090  58 51 69 36 4f 29 fa 36  34 4f f9 37 52 29 fa 36  |XQi6O).640.7R).6|
```

Figure 5: File Dumped by Rekall's procdump

```
$ rekal -f winpmem3.aff4 pslist
Traceback (most recent call last):
  File "/home/user01/Downloads/rekall/rekall-core/rekall/addrspace.py", line 519, in read_partial
    data = self._cache.Get(chunk_number)
  File "/home/user01/Downloads/rekall/rekall-lib/rekall_lib/utils.py", line 147, in NewFunction
    return f(self, *args, **kw)
  File "/home/user01/Downloads/rekall/rekall-lib/rekall_lib/utils.py", line 336, in Get
    raise KeyError(key)
KeyError: 0
(snip)
  File "/home/mkobayashi/envs/win10_rekall2/lib/python3.6/site-packages/pyaff4/aff4_image.py", line 432, in _ReadChunkFromBevy
    "Unable to process compression %s" % self.compression)
RuntimeError: Unable to process compression https://tools.ietf.org/html/rfc1951
```

Figure 6: Rekall Throws an Error when Passed an AFF4 File Acquired with WinPmem 3.x

*4 Volatility 3 Public Beta: The Insider's Preview (https://www.osdfcon.org/events_2019/volatility-3-public-beta-the-insiders-preview/).

source code, but it looks like the error is due to the default compression format for saved data as of WinPmem 3.3 rc2 having been changed to deflate, which Rekall does not support (the default in WinPmem 2.x is zlib). WinPmem's "-c" option specifies compression format, but we had the same processing error even when using zlib. Also, zlib was the default compression format in WinPmem 3.x versions before WinPmem 3.3 rc1, but using these versions to generate AFF4 files that contain the paging file and running them through Rekall produced a different error (AFF4 files without the paging file do work).

So when using Rekall as the analysis tool, WinPmem 2.1 post4 can create memory images that are less likely to cause problems during analysis. That said, we do not recommend using WinPmem 2.x because it is no longer being developed, it can cause forced shutdowns on Windows 10, and it does not support Virtual Secure Mode, which we discuss below. Note that Rekall development is effectively on hold; a new version has not been released since December 2017.

Hopefully, if development of Volatility and Rekall moves forward, they will eventually be able to analyze AFF4 files generated by WinPmem 3.x, but until such time, acquiring memory images and paging files using WinPmem 3.x and exporting the image as shown in Figure 7 is probably the better option. The exported memory image is in RAW format, so you can use either Volatility or Rekall for analysis.

Memory Compression

Paging of a process's virtual memory involves paging file reads and writes, which inevitably degrades system performance. SSDs have become widespread in recent years, so latency is not what it was with HDDs, but performance still unmistakably drops. But page-in and page-out performance can be improved by creating a dedicated area in memory to store paged-out pages in compressed form. The size of the compressed pages can be viewed in the Memory section of the Task Manager's Performance tab (red box in Figure 8). This framework was adopted from Windows 10 1511. Similar frameworks exist in macOS and Linux.

Analyzing memory images containing compressed memory data naturally requires a tool that can cope. Unfortunately, Volatility 2 and Rekall cannot currently analyze compressed memory data from any OS along with other memory pages in a seamless fashion.

```
>winpmem_v3.3.rc3.exe -dd -e */PhysicalMemory -D <export_dir> <image_file>.aff4
```

Figure 7: Exporting a Memory Image from an AFF4 File Generated by WinPmem 3.x

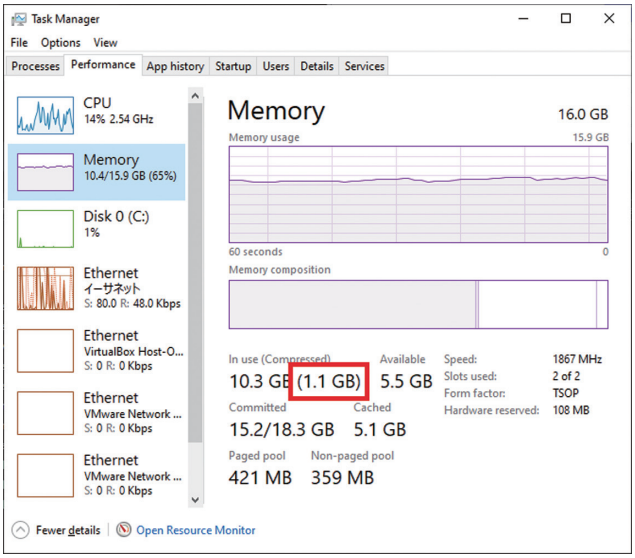

Figure 8: Size of the Compressed Memory

However, at SANS DFIR Summit Austin 2019[*5] and BlackHat USA 2019[*6], FireEye's Omar Sardar and Dimiter Andonov announced implementations of Volatility[*7] and Rekall[*8] that support Windows 10 memory compression.

Note that as both implementations only support Windows 10 1607 through 1809, memory images from Windows 10 1903 or later cannot be analyzed. As of this writing, the announced capability does not seem to have been incorporated into the developers' source code, but as mentioned earlier, Volatility is set to add support in the new version.

Figures 9 and 10 show the results of running the hashdump plugin on the original Volatility and Volatility with support for memory compression. The hashdump plugin retrieves a user's password hash from the registry hive read into memory. Since the user password hash is stored in compressed memory, original Volatility gives no output, but Volatility with support for memory compression is able to print out the hash.

Virtual Secure Mode

The Enterprise and Education editions of Windows 10 1511 and later, and Windows Server 2016 and later, introduce a virtualization-based security (VBS) isolation mechanism that uses virtual machines. Security mechanisms such as Device Guard and Credential Guard are implemented using VBS by executing virtual machines for specific functions in what is called Virtual Secure Mode (VSM). These features allow you to run integrity checks when loading drivers, place execution restrictions on applications, and protect credentials.

Running a tool without support for VSM, such as WimPmem 2.x, yields a BSoD as shown in Figure 11.

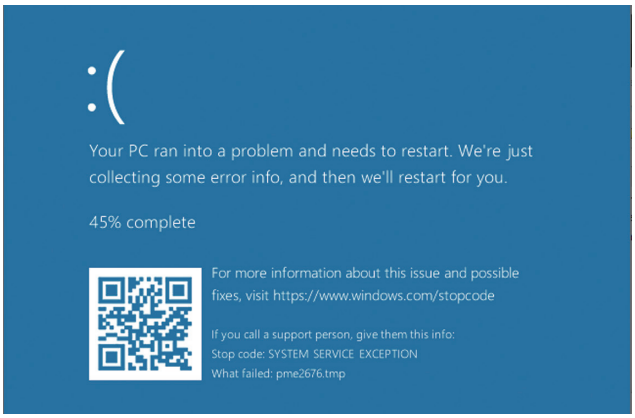

Figure 11: Running Memory Image Acquisition Tools with no VSM Support Yields a BSoD

```
> vol.py --profile Win10x64_17763 -f "Windows 10 1809 Feb x64 en-Snapshot64.vmem" hashdump
Volatility Foundation Volatility Framework 2.6.1
```

Figure 9: Result of Running hashdump Plugin on Original Volatility

```
> vol.py --profile Win10x64_17763 -f "Windows 10 1809 Feb x64 en-Snapshot64.vmem" hashdump
Volatility Foundation Volatility Framework 2.6.1
localuser01:1001:8492e81f418ee4da82b19ef1f27d39af:17e26cdcb1cf786246e7cf8373a540ca:::
```

Figure 10: Result of Running hashdump Plugin on Volatility with Memory Compression Support

*5 SANS DFIR Summit 2019 (https://www.sans.org/event/digital-forensics-summit-2019/summit-agenda).
*6 Paging All Windows Geeks – Finding Evil in Windows 10 Compressed Memory (https://www.blackhat.com/us-19/briefings/schedule/#paging-all-windows-geeks--finding-evil-in-windows--compressed-memory-15582).
*7 win10_volatility (https://github.com/fireeye/win10_volatility).
*8 win10_rekall (https://github.com/fireeye/win10_rekall).

By default, a BSoD causes the host to automatically reboot. The entire contents of memory will of course be erased, so you need to determine beforehand whether the tool you are using is compatible with VSM. Note that the latest versions of the tools in Table 1 do not trigger a BSoD, so consider updating if you are using an older version.

■ Which Memory Image Analysis Tool Should You Use?

Table 2 summarizes the types of data supported by the memory image analysis tools we have discussed. Of these, only Volatility 2 remains in active development and is thus essentially our recommendation. But Volatility with memory compression support should be used when analyzing memory images from Windows 10 1809 or earlier.

You need to use Rekall if you want to analyze paging files as well, but as development has stopped and it has compatibility issues with WinPmem 3.x, it's probably not a go-to tool for many situations. Unfortunately, none of the tools currently available cater to every case. But the situation looks set to improve with the release of Volatility 3.

2.3 Reliable Process Dumpings

So far, we have discussed precautions and strategies for acquiring memory images. But even with these measures, ensuring the integrity of captured memory images is difficult. If the host being analyzed is a VM, you can obtain a complete memory image by taking a snapshot. But on live systems, various processes will be running when you

Table 2: Comparison of Memory Image Acquisition Tools

Tool	Memory images		Paging file	Memory compression	Notes
	RAW	AFF4			
Volatility 2	✓				
Rekall	✓	✓*1	✓*2		Development has stopped
Volatility with memory compression support	✓			✓	Supports up to Windows 10 1809
Rekall with memory compression support	✓	✓*1	✓*2	✓	Supports up to Windows 10 1809

*1 Cannot parse AFF4 files generated by WinPmem 3.3 rc2 and later
*2 Cannot parse AFF4 files containing paging files generated by WinPmem 3.3 rc2 and later

capture the memory image, so the data in memory can change and page-outs can occur. So even if you analyze the memory image, you may find that the contents of process memory are not internally consistent.

Figure 12 compares the .text sections of the files dumped in Figure 1 (left) and Figure 4 (right). As Figure 5 showed, the MZ header was extracted correctly from the file dumped by Rekall, but a look at the .text sections of both files shows that the data values in the file dumped by Rekall are all 0x00 (red area in Figure 12). As discussed, this kind of situation is unavoidable, but it can hinder process analysis. In cases like this, dumping each process from userland individually can give internally consistent process dumps (dumping a process triggers memory access, causing the OS to page-in anything that has been paged out, enabling you to capture all of the process's virtual memory pages).

Several tools for dumping processes exist. Windows Sysinternals ProcDump*9 is a common one. Note that it is different from the plugins of the same name that exist for Volatility and Rekall. Also, Volatility's and Rekall's procdump generate PE format files, whereas Sysinternals ProcDump uses the crash dump format.

Running this with the command in Figure 13 will dump the process with an ID of 4596. You can also specify process name instead of process ID. It's also useful to pass in the

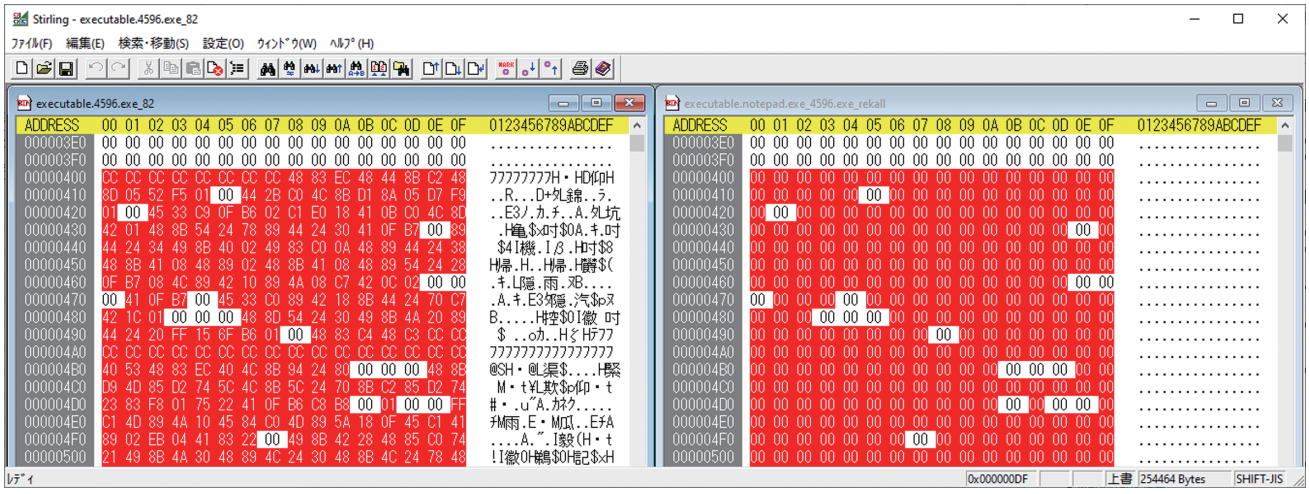

Figure 12: Process Dump Discrepancies

```
>procdump64.exe -ma 4596

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[12:05:29] Dump 1 initiated: C:\Users\localuser01\Desktop\tools\notepad.exe_200206_120529.dmp
[12:05:29] Dump 1 writing: Estimated dump file size is 107 MB.
[12:05:32] Dump 1 complete: 107 MB written in 3.8 seconds
[12:05:33] Dump count reached.
```

Figure 13: Dumping a Process Using the ProcDump Command

*9 ProcDump - Windows Sysinternals (https://docs.microsoft.com/en-us/sysinternals/downloads/procdump).

"-ma" option to dump all of the memory used by the process. The crash dump generated can be read using tools like WinDbg[*10][*11]. Figure 14 is a screenshot of WinDbg reading in the crash dump generated by ProcDump after Figure 4. The code that was missing with Rekall's procdump is now present (red area in Figure 14). Using ProcDump like this ensures a reliable process dump.

2.4 Scripted Process Dumps and Order of Steps to Preserve Artifacts

Creating a script in PowerShell or the like makes it easy to run ProcDump for all processes. Dumping all processes, however, will yield tens of GB or more. This is no problem if there is enough space on the storage destination, but if you want to keep the data on a small USB memory stick or external SSD, compressing the data with a tool like 7-Zip[*12] after each dump is the way to go. You also need to specify the right compression command option to ensure that the temporary files created when compressing files are not written to the disk being analyzed. With 7-Zip, the "-w" option specifies the working folder, so you should use this to specify the disk on which you will be preserving the artifacts.

When running ProcDump, we skip protected processes like System and Registry (trying to dump those results in an error). These scripts need to run in various system environments, so they are often created using PowerShell (installed on Windows by default) or as a batch file, but there are some key points to note.

Launching the PowerShell prompt (powershell.exe) or command prompt (cmd.exe) will also launch the console

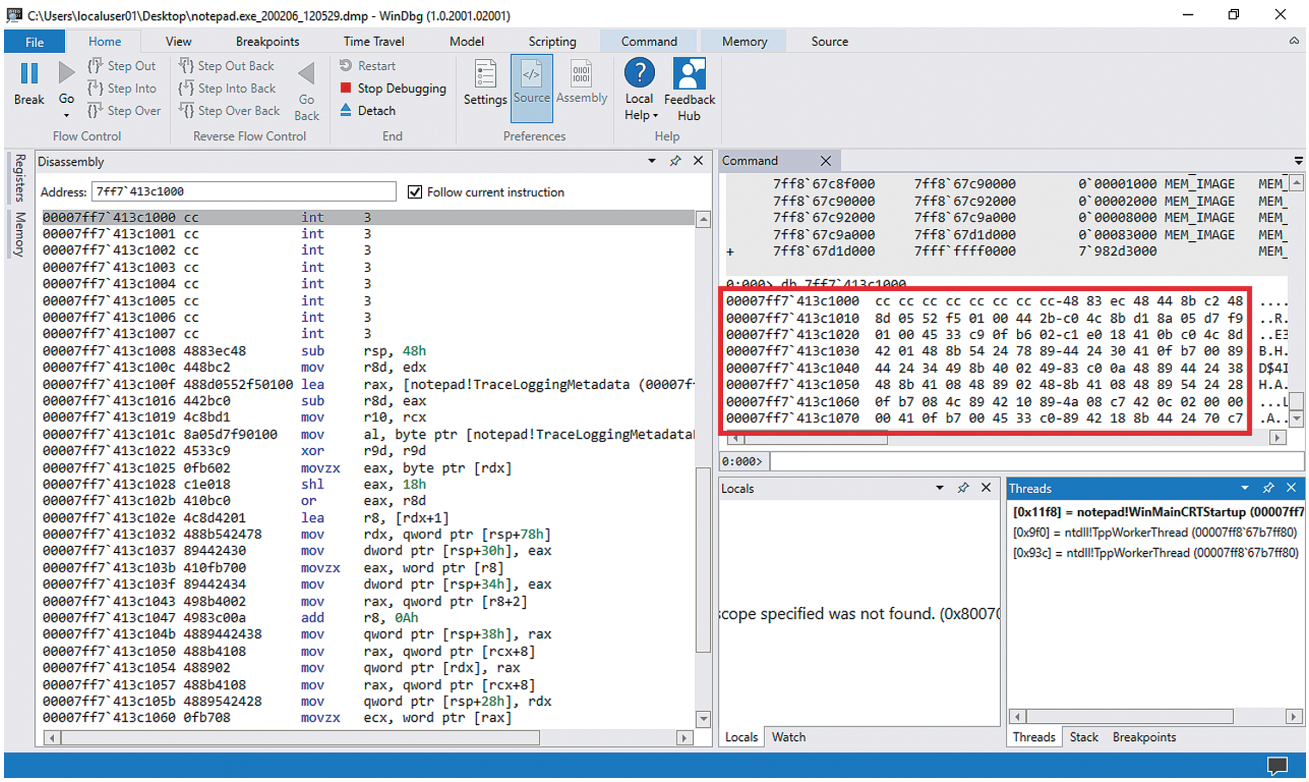

Figure 14: Inspecting the Process Dump in WinDbg

*10 Download Debugging Tools for Windows - WinDbg (https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugger-download-tools).

*11 Debugging Using WinDbg Preview (https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugging-using-windbg-preview).

*12 7-Zip (https://www.7-zip.org/).

window host (conhost.exe). If you try to dump the conhost.exe hosting the prompt that launched the PowerShell script or batch file that runs ProcDump, the ProcDump process will stop. This is because ProcDump suspends a process when dumping it. Since conhost.exe is the process handling the console's IO buffer and display, suspending this process also stops the ProcDump instance running in the console (Figure 15). If necessary, you can analyze conhost.exe by acquiring a memory image using WinPmem.

Depending on the system environment, executing the script can result in several hundred, or more, process dump and file compression cycles. So from a forensics point of view, it may be good practice to preserve the artifacts according to ordered steps such as those in Figure 16.

2.5 Conclusion

We have discussed key points to note when acquiring and analyzing memory images on Windows. We also looked at process dumping with a view to aiding image integrity when acquiring memory images. Many articles dealing with memory forensics imply that the preservation task can be completed by acquiring a memory image using WinPmem or the like, but we hope it is now evident that this is not always sufficient. That said, dumping and compressing all processes is time consuming, so you need to decide on whether to do this in accord with the situation and policies in effect when the incident response is initiated.

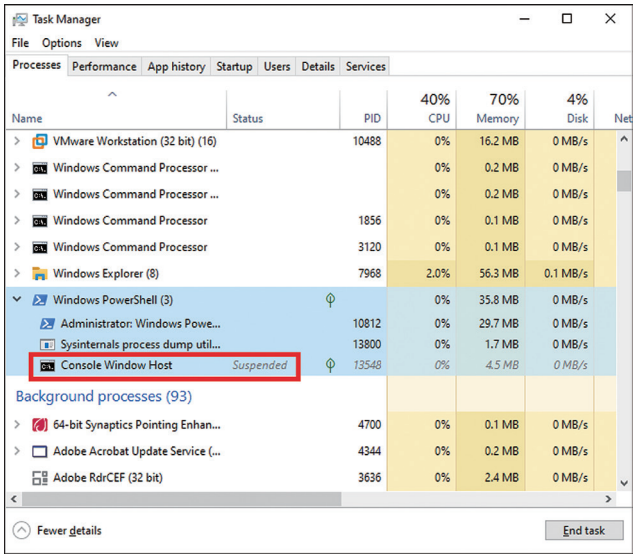

Figure 15: Dumping conhost.exe Causes ProcDump to stop

1. Memory image
2. Artifacts that can be preserved as individual files from disk, such as MFT, Prefetch, and event logs
3. Process dumps
4. Disk image

Figure 16: Example of Ordered Steps for Artifact Preservation

Minoru Kobayashi

Forensic Investigator, Office of Emergency Response and Clearinghouse for Security Information, Advanced Security Division, IIJ
Mr. Kobayashi is a member of IIJ-SECT, mainly dealing with digital forensics. He works to improve incident response capabilities and in-house technical capabilities.
He gives lectures and training sessions at security events both in Japan and abroad, including Black Hat, FIRST TC, JSAC, and Security Camp events.

Binary Program Analysis With No Prior Knowledge of Analysis Target

3.1 Introduction

This article describes binary program analysis technology that is assumption-free in that it requires no prior knowledge of the analysis target, which is being developed at the IIJ Research Laboratory.

Program analysis is a set of techniques for analyzing how programs behave. It can be broadly divided into dynamic analysis, whereby the program is actually run and its behavior is observed, and static analysis, in which the program's structure is reviewed without executing it.

Examples of dynamic analysis include unit tests for checking the integrity of program features, and fuzzing tools that feed a program random input to test its behavior. Examples of static analysis include optimization analysis—which seeks to enhance runtime computational efficiency by eliminating unnecessary code, precomputing operations, and so on—and static type checking to ensure data type consistency so as to avoid runtime errors caused by the program handling data in unintended ways. Whether dynamic or static, these sorts of program analysis techniques are incorporated into integrated development environments (IDEs), helping to streamline development and reduce bugs.

So these program analysis techniques are intended primarily for developers. Meanwhile, you may want to analyze the behavior of programs (binary programs) that are already in the wild. For instance, you may want to know how a suspected malware program behaves or check if firmware from a third party does anything suspicious. In such cases, the person seeking to analyze the program will not always have access to the source code or information on what compiler was used to create it. Dynamic analysis is still possible here. For example, quarantine systems that run suspected malware in a sandboxed environment to analyze its behavior are used in practice. But a problem with dynamic analysis is that only the control path actually executed can be analyzed. So dynamic analysis can be difficult in the case of anti-analysis malware that alters its behavior depending on the execution environment or firmware that has a backdoor enabling it to change its behavior according to specific input.

So to comprehensively analyze the behavior of binary programs, we need to perform static analysis in addition to dynamic analysis. The difficulty involved in the static analysis of binary programs can depend on how much prior knowledge you have about how the program was created. For example, if you can deduce what compiler was used, it may be possible to reconstruct the original source code based on the code patterns the compiler produces. This technique is called decompilation. If you can decompile a program, you can then use existing static analysis techniques on the reconstituted source code to analyze the program's behavior.

That said, you will not always have such prior knowledge available, or know whether you can trust it if it is available. The IIJ Research Laboratory is developing static analysis technology that can be applied to binary programs about which you have almost no prior knowledge, or in other words, when the program's origins are an enigma.

In the following sections, we discuss the difficulty of binary program static analysis and why the difficulty can increase depending on whether you have prior knowledge.

3.2 Binary Program Analysis

T.A program written in a language such as C/C++ (the source code) is converted by software called a compiler into a series of simple machine instructions a CPU can execute. This sequence of machine instructions is encoded into byte data according to the encoding method specified for the CPU architecture. A program expressed as a string of byte data like this is called binary code.

The binary code is embedded in a file with a specific format that makes it an executable file. In addition to the binary code, the executable also contains metadata specifying where to position the program in memory at runtime, where in the program to start execution from (the entry address), and what external library functions are called during execution. In this article, we treat these sorts of executables as binary programs. Static analysis of binary programs in the absence of source code is called binary code analysis.

A familiar example is standard virus detection software, which looks for malware signatures in binary code using a database of such signatures—distinctive byte data strings extracted from known malware. Systems that use machine learning on other metadata contained in executable files to detect unknown malware have also appeared in recent years.

Methods of analyzing binary code as mere byte data like this are suitable for automatic program classification applications such as malware detection, but to learn about how a program will behave in detail, you need to analyze it as a program and not just as a sequence of data. In this article, we call this binary program analysis.

With binary program analysis, you need to extract and reconstruct the program control structure from the binary code. A disassembler (Figure 1) is the first step. As mentioned earlier, machine instructions are converted into byte data according to the rules for each architecture. Doing this in reverse—converting byte data back into machine instructions—is called disassembly.

Simple disassemblers use linear sweeping, which involves disassembling instructions in sequence from the beginning of the binary code. If it encounters non-program data, a linear sweeping disassembler may not be able to correctly disassemble the program from that point onward. Advanced assemblers like IDA Pro, on the other hand, use recursive descent to recursively follow direct jump instructions (a program control instruction for which the destination address is specified as part of the instruction) starting from the entry address (Figure 2). Once a recursive descent disassembler encounters an indirect jump instruction (control instruction for which the destination address is stored in a register or memory), it can proceed no further. This is because, in general, the destination address of an indirect jump instruction presents an undecidable problem for static analysis, so it is, in theory, difficult to proceed.

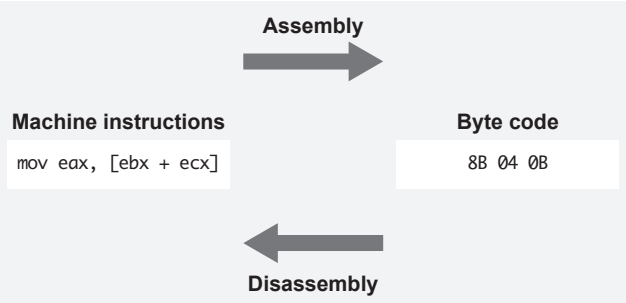

Figure 1: Disassembler

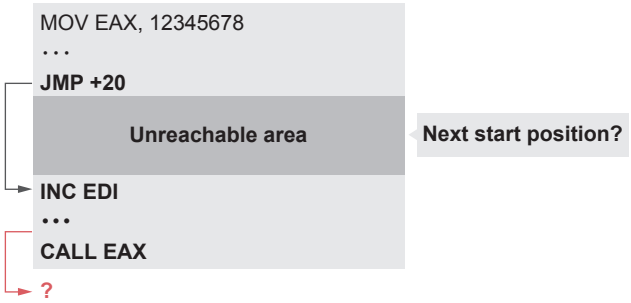

Figure 2: Recursive Descent Disassembler

Recursive descent, therefore, will result in unreachable areas—areas that cannot be reached by direct jump instructions alone. Disassemblers like IDA Pro use various heuristics to identify potential starting positions within these unreachable areas and restart the recursive-descent process. One such heuristic is a method is function identification.

Program development generally involves breaking the program logic into separate functional units, referred to as functions or procedures, to enhance code reusability and development efficiency. How parameters are passed to functions, and the way in which return values (the result of a function's calculations) are received, is determined according to the calling conventions of the CPU, operating system, etc. When functions are compiled, the compiler inserts the necessary preamble and post-amble code according to the calling conventions. This processing produces specific patterns depending on the compiler, so by finding these patterns, you can infer the location of functions. This type of analysis is called function identification.

The entire program can be disassembled by using function identification to break a program into functions and then recursively descending through each of them (Figure 3).

The use of function identification to determine the starting position of functions is premised on the assumption that the program was compiled according to the calling conventions. If this assumption does not hold, the disassembler will not be able to correctly reconstruct the program structure. It has also been reported that identifying the position of functions can be problematic even with programs generated by a compiler if the preprocessing and post-processing code patterns have been omitted due to heavy optimization[*1].

Indirect jump instructions are one reason it is not possible to recreate program structures using disassemblers alone. Using static data analysis to statically resolve indirect jump instruction destination addresses to the extent possible is called control flow reconstruction. As discussed, attempts to statically resolve the destination of indirect jump instructions run up against an undecidable problem, so a complete solution is not possible. Previous research such as CodeSurfer/x86[*2] and Jakstab[*3] has used abstract interpretation to seek approximate solutions to the destination address problem.

There is one more difficulty with control flow reconstruction, however. As discussed, programs consist of several functions. If function identification is first performed to divide a program into separate functions, intra-procedural program analysis, which analyzes each function independently, can be used. If you have insufficient prior knowledge to perform function identification accurately, you will need to use whole program analysis. Even if function positions cannot be identified in advance, programs are actually divided into a number of functions. So with whole program analysis, context-dependency must be taken into account.

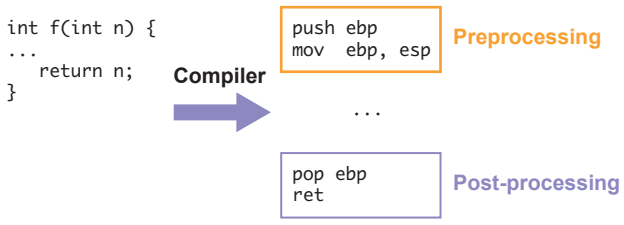

Figure 3: Preprocessing and Post-processing of Functions

*1 Andriesse2016: Andriesse, Dennis, et al. "An in-depth analysis of disassembly on full-scale x86/x64 binaries" 25th {USENIX} Security Symposium ({USENIX} Security 16). 2016.

*2 Balakrishnan2005: Balakrishnan, G., Gruian, R., Reps, T., & Teitelbaum, T. (2005, April). CodeSurfer/x86—A platform for analyzing x86 executables. In International Conference on Compiler Construction (pp. 250-254). Springer, Berlin, Heidelberg.

*3 Kinder2008: Kinder, J., & Veith, H. (2008, July). Jakstab: A static analysis platform for binaries. In International Conference on Computer Aided Verification (pp. 423-427). Springer, Berlin, Heidelberg.

For example, if a function is called from multiple program locations, control is transferred from each function call to the function, and once the function's activity is complete, control returns to the return point. After control has returned, if you want to refer to what the program state was before the function call, the call and return paths must match. This dependence of the analysis on paths is called context-dependency. If this context-dependency is not accounted for, distinguishing between multiple calls is not possible, so irrelevant contextual information muddies the mix when program state is being analyzed after a function call returns, significantly reducing analysis accuracy. Additional processing is needed to resolve context-dependency, such as the use of the stack to ensure consistent handling of function calls and returns. This sort of analysis is called inter-procedural program analysis.

So to enable highly accurate binary program analysis when function positions cannot be identified, you need to estimate function call/return positions during the analysis process. With existing static analysis methods, function positions are inferred by assuming that minimum calling conventions are followed (on Intel x86 architecture, for example, the CALL instruction is used to call functions and the RET instruction is used for returns).

Even the minimum calling conventions, however, cannot be guaranteed if you have no prior knowledge of the target program. For instance, the CALL/RET instructions may be used for purposes other than function calls/returns, or conversely, they could be replaced by other instructions. So to apply existing static analysis methods to binary program analysis, you need to have prior knowledge that guarantees the "goodness" of the target program.

The difficulties in binary program analysis discussed so far can be summarized as follows.

1. Disassemblers do not know the destination of indirect jump instructions.
2. In order to determine the destination of indirect jump instructions using existing static analysis methods, the program must first be divided into functions.
3. To identify the location of functions in a binary program, you need to make assumptions about, e.g., what sort of compiler was used and whether calling conventions are followed.

As such, existing binary program analysis methods require that you have prior knowledge about the conditions under which the program being analyzed was created, and that this knowledge is reliable.

Our method[*4] uses analysis of an intermediate representation that we propose to identify parts of a program as functions during the control flow reconstruction process, the aim being to make static program analysis applicable even in the absence of prior knowledge (i.e., even if the program is "bad").

*4 Izumida2018: Izumida, T., Mori, A., & Hashimoto, M. (2018, January). Context-Sensitive Flow Graph and Projective Single Assignment Form for Resolving Context-Dependency of Binary Code. In Proceedings of the 13th Workshop on Programming Languages and Analysis for Security (pp. 48-53).

3.3 Binary Program Analysis Using the Projective Single Assignment Form

In this section, we describe the method we are working on. As a working example, we use the 32-bit Intel x86 architecture program shown in Figure 4.

[Working example]

```
00401000:   xor ecx, ecx
00401002:   mov ebx, 0x40100c
00401007:   jmp 0x401017
0040100c:   mov ebx, 0x401016
00401011:   jmp 0x401017
00401016:   hlt

00401017:   inc ecx ; (A)
00401018:   jmp ebx
```

Figure 4: Working Example on 32-bit Intel x86 Architecture

In the example in Figure 4, the function code (A) is called twice, but instead of using the CALL/RET instructions, the code stores the return address in the EBX register and then jumps to (A), at which point it increments the ECX register by 1 and then jumps to the address stored in EBX, which takes it back to the instruction following the call. Existing analysis tools like IDA Pro cannot recognize code like this as function calls because it does not use the standard CALL/RET style (Figure 5).

In our research, we convert each machine instruction into simple assignment forms, and then further convert this into static single assignment (SSA) form. SSA is an internal representation format used in compiler optimization analysis. It changes variable names so that each variable definition is unique. This clarifies the definition-and-use (def-use) relationship of each variable, making it easy to understand the

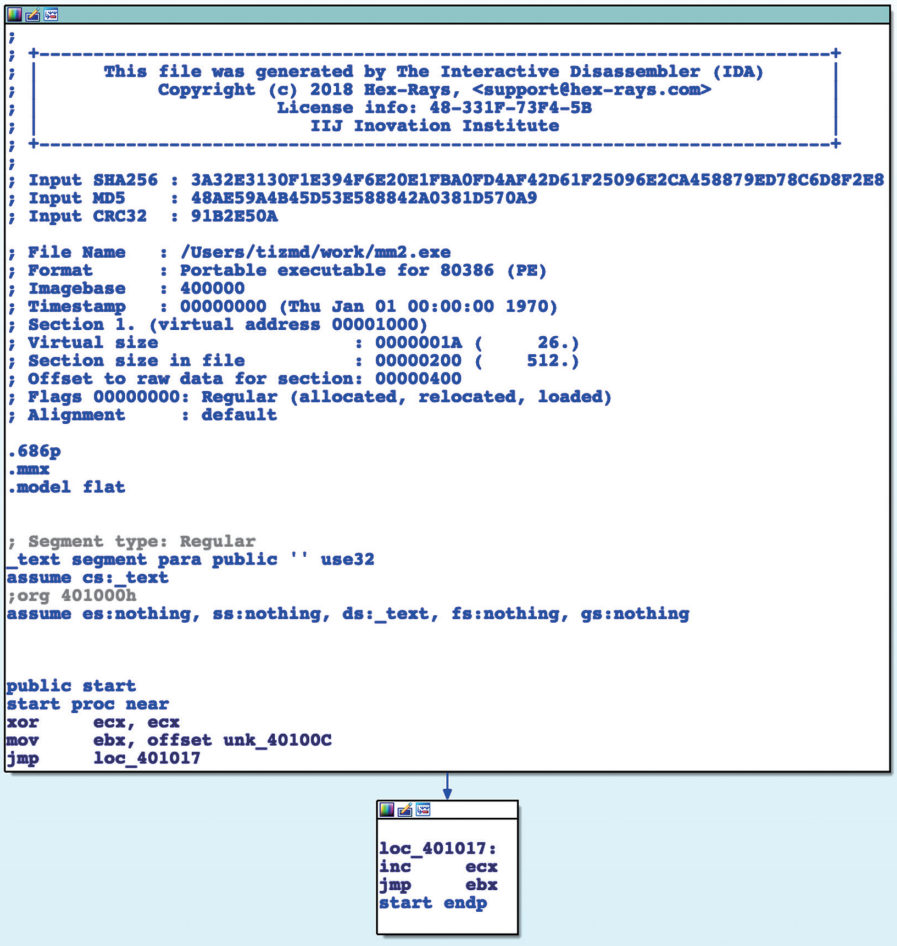

Figure 5: Example of Disassembly in IDA Pro

flow of information. For example, in Figure 6, the two ECX assignments are differentiated as ECX_1 and ECX_4.

Here, the code ends with an indirect jump to the return address stored in the EBX register (the jump instruction is expressed as an assignment to the program counter [EIP]). If you trace the definition of EBX_2 on the the right-hand side of the assignment, you can easily see that it is 0x40100c. So the destination address of this indirect jump expands to 0x40100c.

Figure 7 graphs the point up to where the second (A) call is completed. SSA expresses information merge points by introducing a pseudo-function called the Φ-function. For example, in the assignment statement $EBX_8 \leftarrow \Phi_8$ (3: EBX_2, 7: EBX_6), the EBX register values EBX_2 from node 3 and EBX_6 from node 7 merge and are newly assigned to the variable EBX_8. As before, if we trace the definition of EBX_8, it is expressed with a Φ-function as Φ_8 (3:0x40100c, 7:0x401016).

This means that if control reaches node 8 from node 3, the EBX register takes the value of 0x40100c, whereas if control comes from node 7, it takes the value 0x401016, so there is a merging of information. In our research, we refer to this changing of destination addresses due to information merging at certain points as context-dependency. Here, the code in the range from 0x401017 to 0x401018 is reused by multiple contexts, so we can infer that this is a function.

If context dependency is detected in this way, our method inserts a pseudo-function called the Π-function (Figure 8). The Π-function acts as a projection function for the Φ-function. For example, the expression $\Pi_{3 \to 8}$ (...) means that the information coming from node 3 is extracted from the information merged at node 8, so it is evaluated as $\Pi_{3 \to 8}(\Phi_8$ (3: X, 7: Y)) ⇒ X. This extension of the SSA form with Π projective functions is our own novel approach, which we call the projective single assignment (PSA) form.

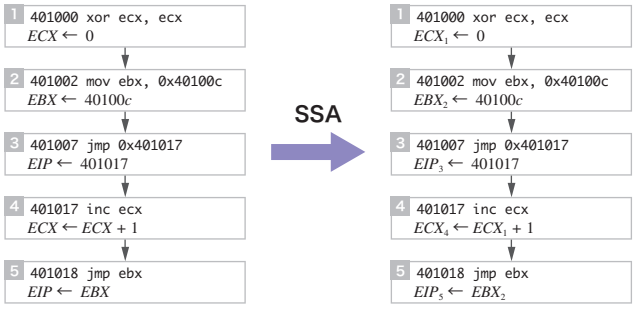

Figure 6: SSA Form: Up to the end of the first (A) call

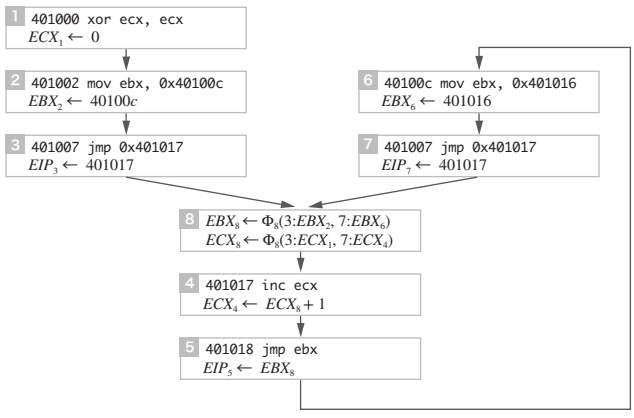

Figure 7: SSA Form: Up to the end of the second (A) call

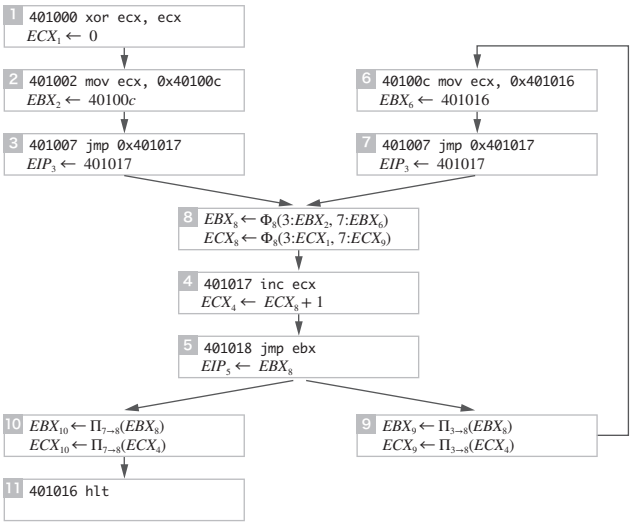

Figure 8: PSA Form: Insertion of Π-functions

Using Π-functions, the value of the ECX register when the program ends (node 11), for example, can be derived as follows.

$$ECX_{10} \Rightarrow \Pi_{7\rightarrow 8}(ECX_4) \Rightarrow \Pi_{7\rightarrow 8}(ECX_8) + 1 \Rightarrow \Pi_{7\rightarrow 8}(\Phi_8(3{:}ECX_1, 7{:}ECX_9)) + 1$$
$$\Rightarrow ECX_9 + 1 \Rightarrow \Pi_{3\rightarrow 8}(ECX_4) + 1 \Rightarrow \Pi_{3\rightarrow 8}(ECX_8) + 2 \Rightarrow \Pi_{3\rightarrow 8}(\Phi_8(3{:}ECX_1, 7{:}ECX_9)) + 2$$
$$\Rightarrow ECX_1 + 2 \Rightarrow 2$$

So by extracting context-dependency during the reconstruction process, we are able to resolve programs even if they do not follow calling conventions..

3.4 Application: Verifying Buffer-Overflow Safety

NThe projective single assignment adds not only projective Π-function but also conditional Γ-functions, which record the branching condition at each conditional branch. If loops are used within a function to rewrite data on the stack, the use of Γ-functions makes it possible to determine whether it is possible for the return address on the stack to be overwritten.

For example, converting the program in Figure 9 to PSA form and simplifying it results in Figure 10.

Here, Ld (M, A, N) denotes an N-byte value being read from address A in memory state M, and St(M,A,X,N) denotes the N-byte value X being set at address A in memory state M. In this example, when the function finishes, the program jumps to the 4-byte return address stored in the stack location expressed as $EBP_1 + 4$. The condition under which this area will be overwritten during the for loop is expressed in the PSA form as follows.

$$\Gamma(i_2 < 10, EBP_1 + 4 \leq EBP_1 - 10 + i_2 \leq EBP_1 + 8)$$

This means whether $EBP_1 + 4 \leq EBP_1 - 10 + i_2 \leq EBP_1 + 8$ is satisfied under the condition that $i_2 < 10$. Since i_2 is defined as Φ (i_1, i_4) and i_4 is defined in the loop, i_2 will change within the loop. If an i_2 value that satisfies this condition exists, there is a possibility of the return address value being overwritten during the loop. In this case here, however, it is easy to see that no such i_2 value exists. In other words, it is guaranteed that this loop will not change the return address area.

In practice, loop conditions and the memory overwrite conditions appear in more complicated forms. Finding loop invariants (conditions that do not change within a loop) is important in determining whether conditions such as these that vary within loops are satisfiable. Research on analysis methods for automatically evaluating such loop invariants has advanced in recent years and has been implemented in SMT solvers such as Z3. In our research, our objective is to extract loop conditions and memory overwrite conditions from binary programs and automatically calculate loop invariants using an SMT solver. This analysis will not always determine a solution within a specific timeframe, but if it can determine there to be no possibility of an overwrite, it

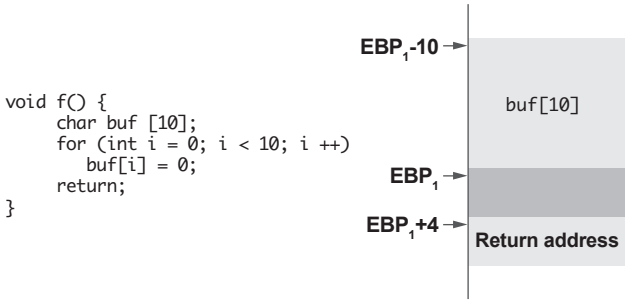

Figure 9: Example Program

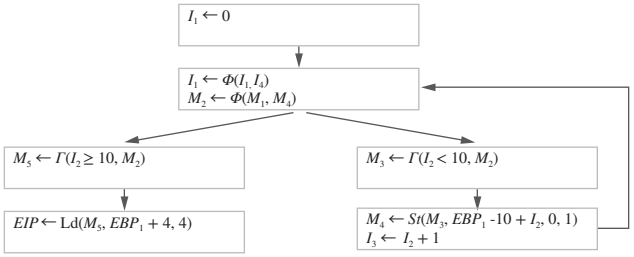

Figure 10: Loop

can guarantee that no buffer overflow will occur. And if it does identify the possibility of an overwrite, you can investigate the conditions under when overwrites can occur.

We are currently studying the application of this method to technology that detects vulnerabilities, such as buffer overflows, and the presence of backdoors, such as Trojan horses, in embedded firmware in AI edge devices and the like.

3.5 Conclusion

This article has described the assumption-free binary program analysis technology being developed at the IIJ Research Laboratory. Existing static analysis methods require that a program is first divided into separate functions, but doing this requires prior knowledge of or assumptions about how the program was generated. Using an extension of the SSA form, our method makes it possible to identify the location of functions by evaluating the destination of indirect jump instructions while also extracting context-dependency. This means that program analysis can be performed even on "bad" programs about which no prior knowledge can be obtained.

However, static binary program analysis involves undecidable problems, so no analysis method can provide a complete solution. Even with our method, we halt the evaluation and generate an approximate solution when the evaluated form becomes bloated and it appears that finding a static solution will be difficult.

Another method for binary program analysis is symbolic execution. This method of analysis sits somewhere between static and dynamic analysis. In 2016, mayhem[*5] and angr[*6], analysis tools that use symbolic execution, were among the leaders in the Cyber Grand Challenge, an IT security automation contest created by DARPA. Symbolic execution can detect if a program might produce dangerous states such as buffer overflows. But proving that a program is safe—meaning that it cannot produce any dangerous states at all—requires exhaustive execution, which is not something that symbolic execution is suited to. In this respect, we believe symbol execution and our method can complement each other.

Looking ahead, we aim to develop binary program analysis tools that integrate our method with other analysis techniques such as symbolic execution and dynamic analysis.

Acknowledgment
This research is carried out as part of "Research & Development on Fundamental Technologies Required for Comprehensive Security Evaluation of AI Edge Devices" work commissioned by Japan's New Energy and Industrial Technology Development Organization (NEDO).

Tomonori Izumida
Researcher, IIJ Innovation Institute (since 2015). PhD (information science).

*5 Cha2012: Cha, S. K., Avgerinos, T., Rebert, A., & Brumley, D. (2012, May). Unleashing mayhem on binary code. In 2012 IEEE Symposium on Security and Privacy (pp. 380-394). IEEE.

*6 Wang2017: Wang, F., & Shoshitaishvili, Y. (2017, September). Angr-the next generation of binary analysis. In 2017 IEEE Cybersecurity Development (SecDev) (pp. 8-9). IEEE.

May 2020 Vol.46

Internet Initiative Japan

About Internet Initiative Japan Inc. (IIJ)

IIJ was established in 1992, mainly by a group of engineers who had been involved in research and development activities related to the Internet, under the concept of promoting the widespread use of the Internet in Japan.
IIJ currently operates one of the largest Internet backbones in Japan, manages Internet infrastructures, and provides comprehensive high-quality system environments (including Internet access, systems integration, and outsourcing services, etc.) to high-end business users including the government and other public offices and financial institutions.
In addition, IIJ actively shares knowledge accumulated through service development and Internet backbone operation, and is making efforts to expand the Internet used as a social infrastructure.

IIJ-MKTG020-0044

Internet Initiative Japan Inc.

Address: Iidabashi Grand Bloom, 2-10-2 Fujimi, Chiyoda-ku,
Tokyo 102-0071, Japan
Email: info@iij.ad.jp URL: https://www.iij.ad.jp/en/