IIR

Internet
Infrastructure
Review

May 2019
Vol. 42

Periodic Observation Report

SOC Report

Focused Research (1)

Deep-Learning for Log Analysis to Detect Malicious Communications

Focused Research (2)

Designing a Large-scale Email System

IIJ
Internet Initiative Japan

Internet Infrastructure Review

May 2019 Vol.42

Executive Summary ····· 3

1. Periodic Observation Report ····· 4
1.1 Introduction ····· 4
1.2 Observational Data ····· 4
1.2.1 Attacks Involving Cryptocurrencies ····· 4
1.2.2 SYN/ACK Reflection Attack ····· 6
1.2.3 Resurgence of Attacks Targeting Known Vulnerabilities ····· 7
1.3 Detecting Malicious Transmissions Using Machine Learning ····· 8
1.3.1 Application to DNS Query Data ····· 8
1.3.2 Application to Web Proxy Data ····· 8
1.4 Conclusion ····· 9

2. Focused Research (1) ····· 10
2.1 Introduction ····· 10
2.2 Background ····· 10
2.3 Detecting Communications with Malware C&C (C2) Servers ····· 11
2.4 Exploit Kit Detection ····· 15

3. Focused Research (2) ····· 18
3.1 Introduction ····· 18
3.2 Challenges and Goals of the System Overhaul ····· 18
3.2.1 Review of the Architecture ····· 18
3.2.2 Better Filtering ····· 19
3.2.3 Avoiding Over-reliance on Any One Vendor ····· 19
3.3 Single-stage MTA for Effective Use of Hardware Resources ····· 20
3.4 Making Antivirus/Antispam Engines Interchangeable ····· 20
3.5 Decision to Develop MTA In-house ····· 22
3.6 Overhaul Outcomes ····· 23
3.6.1 Achieving our Development Goals ····· 23
3.6.2 Secondary Benefits ····· 23

Executive Summary

Conventional DDoS attacks primarily used botnets consisting of a large number of malware-infected PCs to flood the attack target with connections. Then from around 2013, we began to observe large-scale DDoS attacks where the attacker had taken advantage of poorly configured IoT devices, such as home routers and internet cameras, exploiting their vulnerabilities to infect them with malware.

We expect the number of internet-connected IoT devices to continue rising strongly ahead, and preventing them from being used in DDoS attacks will be crucial to keeping the internet safe. In response to such threats, on February 1, 2019, Japan's Ministry of Internal Affairs and Communications, the National Institute of Information Communications and Technology (NICT), and telecommunications carriers unveiled a project to survey IoT devices and alert users called "NOTICE (National Operation Towards IoT Clean Environment)".

Through the NOTICE project, the NICT will survey IoT devices on the internet to identify devices vulnerable to cyberattacks and pass information about relevant devices onto the telecommunications carriers. The carriers will then identify the users of those devices and alert them to the issue. This is a collaboration between government and the private sector aimed at enhancing the safety of internet-connected IoT devices, and IIJ is an active participant.

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 discusses our SOC Report. IIJ's SOC collates and analyzes a range of logs, including those from security devices provided as IIJ services, on its Data Analytics Platform, and we release up-to-date information on threats observed in blog format via wizSafe Security Signal. In this edition, we draw from past wizSafe Security Signal posts and look at three types of notable activity revealed using our Data Analytics Platform and also describe the use of our Data Analytics Platform for machine learning.

Our first focused research report in Chapter 2, titled "Deep-Learning Analysis of Logs to Detect Malicious Communications", is a restructured version of the content of a presentation given by IIJ staff at Black Hat Europe 2018. The report looks at general-purpose methods for detecting threats using logs from commonly available servers and network devices, rather than using specialized equipment or security devices. These huge logs require complex processing, but our research has confirmed there is potential to make effective use of them if they are properly optimized for deep learning.

Our second focused research report in Chapter 3 discusses the overhaul of IIJ Secure MX Service, the email gateway service that IIJ provides. More than 10 years have passed since the service was first launched. It remains a key service for IIJ and continues to see strong growth in subscriptions to this day. That said, with the changes in the landscape over those 10 years, the system had grown obsolete and was saddled by various issues. The chapter presents a report from one of the engineers involved in development process, which included revising system architecture to solve those issues and the decision to develop the system in-house, and we hope this serves as a good reference.

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 announced its new security business brand wizSafe[*1] on October 31, 2016 and is constantly working to bring about a world in which customers can use the Internet safely. As part of such efforts, we release up-to-date information on security threats observed at our SOC in blog format via wizSafe Security Signal[*2]. This includes some information on threats identified through IIJ's Data Analytics Platform. For an overview of the Data Analytics Platform, see Internet Infrastructure Review (IIR) Vol. 38[*3].

Here, we give an overview of analysis using the Data Analytics Platform. The logs collected on the platform naturally include those from security devices such as firewalls, IPS/IDS, and antivirus solutions provided as IIJ services, as well as logs of DNS queries, Web access, incoming/outgoing email, and so forth. Characteristically, these logs contain only a tiny amount of abnormal traffic (threats) among a large amount of normal traffic. We therefore need to think about how to go about aggregating and visualizing the data so that we can identify threats clearly.

Section 1.2 describes information on threats revealed via the Data Analytics Platform in 2018, and Section 1.3 describes new initiatives using the Data Analytics Platform. The observations for 2018 are summarized in wizSafe Security Signal[*4].

1.2 Observational Data

First, we look at activity identified using the Data Analytics Platform that is particularly noteworthy. This information is taken from wizSafe Security Signal posts from last year.

1.2.1 Attacks Involving Cryptocurrencies

Attempts to monetize attacks using cryptocurrencies attracted attention in 2018. Analysis on IIJ's Data Analytics Platform also revealed several cases of attackers attempting to exploit cryptocurrencies.

The first example involves manipulating a website to embed a mining script. Our SOC's observations uncovered multiple cases of mining scripts embedded in websites that do not appear to have been put there intentionally by the website

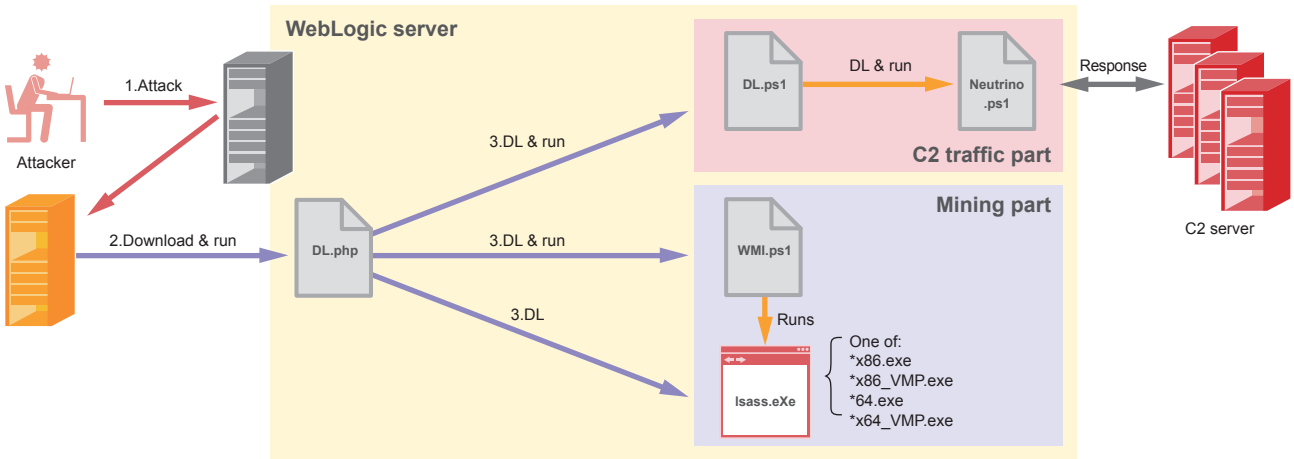

Figure 1: Overview of GhostMiner Attack

*1 IIJ announces new security business brand wizSafe (https://www.iij.ad.jp/en/news/pressrelease/2016/1031.html).
*2 wizSafe (in Japanese at https://wizsafe.iij.ad.jp).
*3 IIJ, Internet Infrastructure Review (IIR) Vol. 38 (https://www.iij.ad.jp/en/dev/iir/038.html).
*4 wizSafe, "wizSafe Security Signal 2018 Annual Summary" (in Japanese at https://wizsafe.iij.ad.jp/2019/03/601/).

administrator. By exploiting website vulnerabilities and so forth, attackers can embed mining scripts into Web pages. When a user views a tainted site, the user's computer runs the cryptocurrency mining script, and any mined proceeds go to the attacker.

The above is an example of an attack aimed at clients, but we have also observed cryptocurrency mining attacks on servers. One specific example is an attack campaign[*5] called GhostMiner (Figure 1). The GhostMiner campaign was observed in March 2018 and exploits a vulnerability (CVE-2017-10271) in Oracle WebLogic Server. The vulnerability allows the execution of remote code, so the attacker ultimately attempts to get the Web server to mine cryptocurrency. We have also observed several other attempts to use remote code execution vulnerabilities to get servers to mine cryptocurrency[*6][*7].

Yet another example involves not mining but attempts to illegally transfer funds. In December 2018, we observed scanning activity (Figure 2) targeting the JSON-RPC protocol used in an Ethereum client[*8]. The scanning activity was looking for Ethereum clients that are accessible via the Internet due to a misconfiguration. We note that a number of conditions must be met for the funds transfer to actually complete successfully.

Cryptocurrencies are appealing to attackers because attacks on them can be monetized directly and because, depending on the type, they offer a high degree of anonymity. Also, any environment that has computational resources can be attacked, as evidenced by the variety of attacks geared to cryptocurrency mining, which target both clients and servers. We expect attackers to continue to target cryptocurrencies as one means of monetizing attacks.

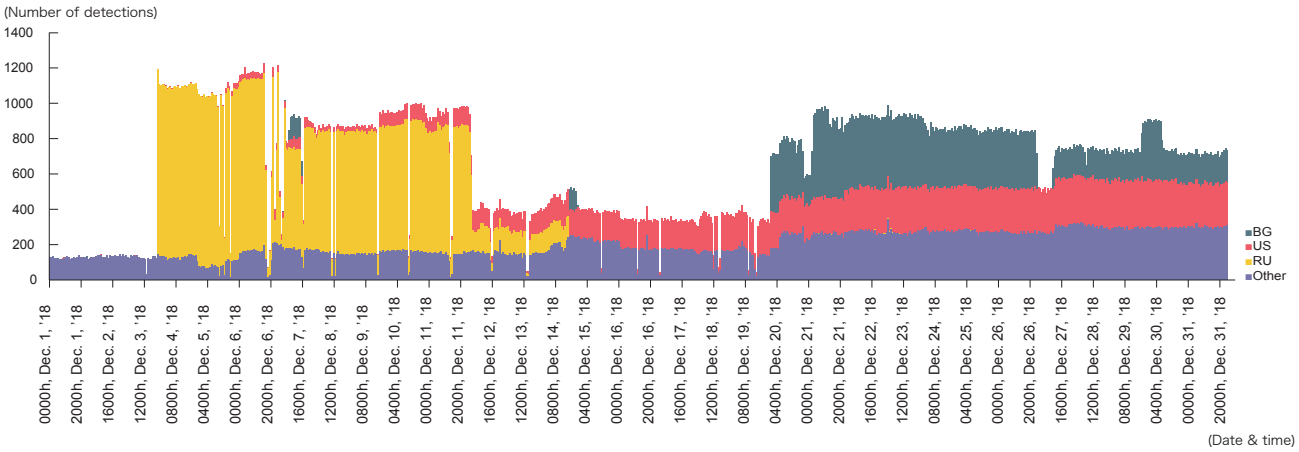

Figure 2: Scanning of 8545/TCP (Dec. 2018)

*5 wizSafe, "GhostMiner infections spreading" (in Japanese at https://wizsafe.iij.ad.jp/2018/04/323/).
*6 wizSafe, "wizSafe Security Signal January 2018 Observation Report" (in Japanese at https://wizsafe.iij.ad.jp/2018/02/247/).
*7 wizSafe, "wizSafe Security Signal February 2018 Observation Report" (in Japanese at https://wizsafe.iij.ad.jp/2018/03/286/).
*8 wizSafe, "Ethereum JSON-RPC scans observed" (in Japanese at https://wizsafe.iij.ad.jp/2019/01/541/).

1.2.2 SYN/ACK Reflection Attack

One peculiar example of a DDoS attack that the SOC observed in 2018 is a SYN/ACK reflection attack using 80/TCP. This was included in wizSafe Security Signal for September 2018[*9] (Figure 3). The attack sends TCP SYN packets with a spoofed source address to many addresses simultaneously, thereby effectively recruiting the resulting SYN/ACK packet responses to perform a DDoS attack on the source address.

This SYN/ACK reflection attack was observed by the SOC on September 26, 2018, but it has also been observed on a small scale since October, and attacks of the same type are detected daily via the Data Analytics Platform. One feature of the DDoS attack observed on September 26 is that the source invokes the attack by sending a small amount of SYN packets to servers on which the 80/TCP port is open to the Internet. If an attacker sends a high volume of SYN packets to a single server, the administrator of the receiving server is liable to think that a TCP SYN flood attack[*10] is underway and block further traffic. If this happens, the attacker may not be able to realize an attack of the scale envisaged. Also, because only a small volume of SYN packets is received per server, one can infer that the attacker is probably sending out SYN packets far and wide.

The attack described above uses port 80/TCP, and servers on which 80/TCP is open can generally be considered to be Web servers. Hence, normal Web access traffic on such servers do contain a small amount of SYN packets, the type of packet used in a SYN/ACK reflection attack, and it is thus difficult to determine whether any of those packets are being used in an attack. In this example, we detected the attack by cross-analyzing the multiple customer firewall logs present on our Data Analytics Platform.

Because firewall logs reveal information about internal and external access, such attacks can be detected when multiple firewall logs indicate that 80/TCP responses are being generated for a single specific IP address (the spoofed IP address being attacked). However, this feature could indicate scanning activity rather than a DDoS attack. We therefore

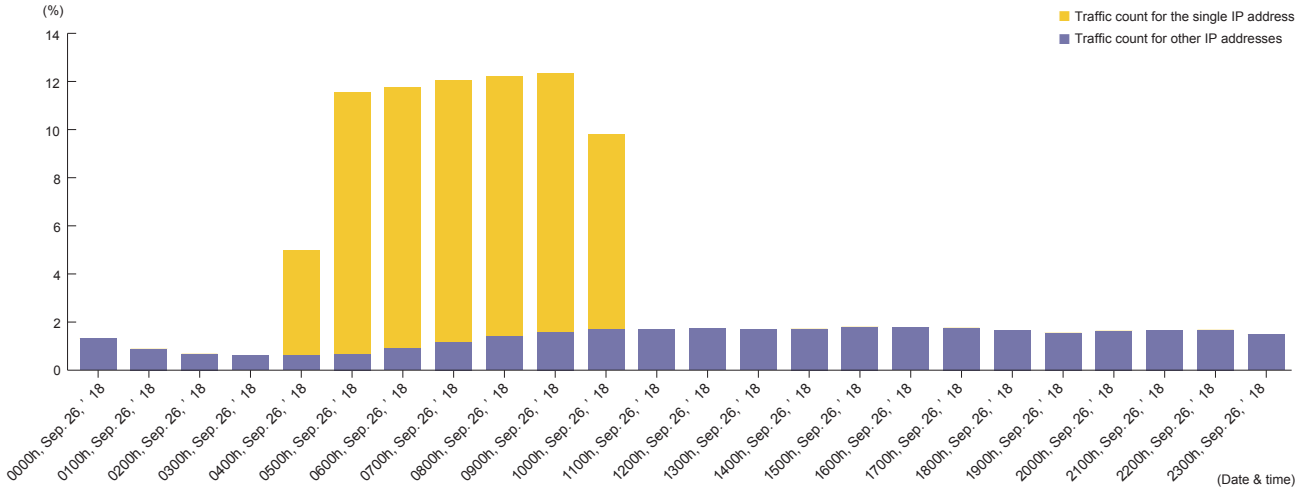

Figure 3: Increase in 80/TCP Traffic from a Single IP Address

*9 wizSafe, "wizSafe Security Signal September 2018 Observation Report" (in Japanese at https://wizsafe.iij.ad.jp/2018/10/470/).

*10 In a TCP SYN flood attack, the attacker sends a large amount of SYN packets—requests used to establish a TCP connection—to the target system, causing it to prepare for a large number of connections and thereby wasting processing power, memory, etc.

differentiate between DDoS attacks and scanning activity based on total bytes sent/received, duration, and so on as calculated from the firewall logs.

The SYN/ACK reflection attack that we wrote about in September 2018 was also observed in IIJ's honeypots. We reported about this in detail in an IIJ-SECT blog post (in Japanese): "SYN/ACK reflection attack using IoT devices as a springboard"[*11]. The post describes changes in the ports used in the attack and reveals that it is a complex DDoS attack that uses the UDP protocol, so we encourage you to read through it.

1.2.3 Resurgence of Attacks Targeting Known Vulnerabilities

One notable of the 2018 analysis performed on the Data Analytics Platform is that some attacks targeting vulnerabilities that have already been disclosed and for which patches have been made available have re-emerged after being dormant for some time. One example is malware that exploits a vulnerability (CVE-2017-11822) in the Microsoft Office Equation Editor.

The vulnerability that the malware exploits is a buffer overflow issue with the Microsoft Office Equation Editor that allows remote code execution. Microsoft issued a patch that fixes this vulnerability in November 2017. As a workaround, users can also disable the Equation Editor as a means of avoiding this attack without applying the patch.

We observed an attack targeting this vulnerability via the Data Analytics Platform in September 2018, almost a year after the fix had been issued (Figure 4)[*12]. The attackers sent malware that exploits the vulnerability as an email attachment. We think this was a deliberate attempt to target systems in which a fix was never applied or in which the Equation Editor had only temporarily been disabled as a means of avoiding the attack, and that it was timed for when awareness of this vulnerability had faded somewhat.

In addition to the Microsoft Office vulnerability discussed here, we have observed multiple other similar cases via the Data Analytics Platform[*13]. A lesson to be learned from these observations is that whether one implements a fundamental

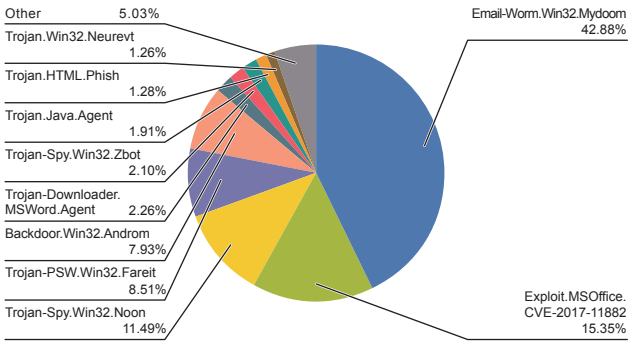

Figure 4: Breakdown of Malware Types Detected in Received Emails (Sep. 2018)

*11 IIJ-SECT Security Diary, "SYN/ACK reflection attack using IoT devices as a springboard" (in Japanese at https://sect.iij.ad.jp/d/2019/02/128021.html).
*12 wizSafe, "wizSafe Security Signal September 2018 Observation Report" (in Japanese at https://wizsafe.iij.ad.jp/2018/10/470/).
*13 wizSafe, "wizSafe Security Signal November 2017 Observation Report" (in Japanese at https://wizsafe.iij.ad.jp/2017/12/184/).

fix to a vulnerability or, depending on the circumstances, any of the various workarounds available, it is crucial that such measures are kept in place indefinitely.

1.3 Detecting Malicious Transmissions Using Machine Learning

Characteristically, the data analyzed on our Data Analytics Platform include only a tiny amount of abnormal traffic (threats) among a large amount of normal traffic. Efforts are being made to use machine learning to discover such threats. The main task handled is that of detecting anomalies from imbalanced data. Here, we describe two such projects that are underway, along with the challenges they face.

1.3.1 Application to DNS Query Data

The domain names of the C2 (command & control) servers used by malware may be generated algorithmically using a DGA (domain generation algorithm). The domain names generated by DGAs differ widely depending on the type of algorithm and the parameters used when running it. This can make it difficult to blacklist the malware's servers ahead of time or to create an expression for the detection signature.

So in this project, we aim to solve the problem by combining the Data Analytics Platform's DNS query data with machine learning. We take this approach because tasks that humans find difficult to construct rules for can be amenable to machine-learning solutions. A desirable property of machine learning algorithms is that they can autonomously acquire the ability to classify anomalies when provided with data containing features that are effective in identifying those anomalies. For example, in IIR Vol. 41, we looked at URL strings and described an approach to identifying rogue sites using neural networks[*14].

We know of several attempts to use machine learning to detect DGAs, including some that have already been put into real-world use. Currently, we are engaged in research that follows on from the FANCI (Feature-based Automated NXDomain Classification and Intelligence)[*15] system announced at USENIX Security '18. As the conference paper on FANCI explains, the system combines domain features inspired by those used in past research with a machine learning algorithm known as random forests, and it generalizes very well.

As the first step in our follow-up research, we aim to stick to the methodology described in the paper as much as possible and use the DNS query data available from our Data Analytics Platform. This first step is intended to assess whether the methodology can be applied unmodified to the Data Analytics Platform's data. We do this because any given methodology will not necessarily produce the same results when different data are used. Next, if we determine that the methodology cannot be applied as is, and that it is possible to investigate why and implement a solution, we intend to work toward a practical implementation that may include additional performance enhancements. Potential performance enhancements could, for example, come from the use of gradient boosting decision trees, a popular method in recent years, or ensemble learning that combines undersampling and bagging.

The volume of data passing through and processed by the Data Analytics Platform is large, however, so as a matter of practicality, the model needs to have high throughput. We aim to strike a balance between the increasing computational load that results from the use of more complicated models and workflows and the performance enhancements that can be obtained, and with some fine tuning, we ultimately aim to build the system into our Data Analytics Platform to provide one means of detecting these anomalies.

1.3.2 Application to Web Proxy Data

One other project we are pursuing aims to detect communications sent to C2 servers by malware in Web proxy data. We are currently running validation tests with the objective of applying the methodology presented by IIJ engineers at Black Hat Europe 2018[*16] to our Data Analytics Platform's logs. The methodology presented at Black

*14 IIJ, Internet Infrastructure Review (IIR) Vol. 41 (https://www.iij.ad.jp/en/dev/iir/041.html).

*15 USENIX, "FANCI: Feature-based Automated NXDomain Classification and Intelligence" (https://www.usenix.org/conference/usenixsecurity18/ presentation/schuppen).

*16 Black Hat, "Deep Impact: Recognizing Unknown Malicious Activities from Zero Knowledge" (https://www.blackhat.com/eu-18/briefings/schedule/#deep-impact-recognizing-unknown-malicious-activities-from-zero-knowledge-12276).

Hat Europe 2018 is described later in this edition under "Focused Research (1): Deep-Learning Analysis of Logs to Detect Malicious Communications".

The project uses convolutional neural networks, which are commonly applied to image recognition tasks, to discern trends in normal traffic and anomalous traffic (with a C2 server). The key here is learning model performance and its evaluation.

If there were, say, a model capable of producing 95% accuracy or better, this would generally be regarded as good performance. But because the volume of logs collected on the Data Analytics Platform is enormous, 1% of this data is not the sort of volume that a human could process by eye. Even if false positives do arise, the model needs to provide a level of accuracy that is tolerable when put into operation. This, of course, is the case when only machine learning is used, and approaches that reduce false positives through non-machine-learning systematic processing are also conceivable.

Aside from accuracy, we also need to be aware of differences in the distributions of the datasets we are dealing with. It is quite possible that the distributions of datasets used by reportedly well-performing machine learning models presented at conferences, academic events, and the like are characteristically different from the dataset distributions encountered by our SOC, so follow-up research is needed.

In view of the above, the SOC takes the overall design and operation of systems that use machine learning models into consideration, conducting follow-up research and working to improve the accuracy of machine learnings models, and is focused on building systems that improve quality without putting any additional load on current security operations.

One attempt to improve accuracy entails feature engineering. There is a strong perception that feature engineering involves adding features expected to be effective on the basis of data analysis, but other approaches also exist. For example, various statistics can be calculated from existing features, combined with the data from which they were derived, and then used for learning and evaluation. We will also use various other methods to repeatedly add and evaluate features as we work to enhance model.

1.4 Conclusion

In this edition, we provided an overview of analysis using the Data Analytics Platform, went over some actual observations from 2018, and described our efforts with respect to machine learning. The final machine learning approach that we described has the potential to further expand the detectable range for threats where detection with conventional methods is difficult or subject to limitations. We are able to pursue these efforts entirely because we are able to use traffic logs received from customers on the Data Analytics Platform, subject to customer consent. Machine learning approaches require large volumes of data in particular, so it is fair to say that we are only able to pursue these efforts because we have access to these traffic logs via the Data Analytics Platform. We will continue to provide up-to-date information on threats through wizSafe Security Signal and the IIJ-SECT blog, and we will continue striving to bring about a world in which the Internet is even safer for customers to use.

Satoshi Kobayashi
Data Analyst, Security Operations Center, Security Business Department, Advanced Security Division, IIJ

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

Deep-Learning for Log Analysis to Detect Malicious Communications

2.1 Introduction

Deep learning can be used to discover malicious communications. Here, we describe two methods of detecting malware communications caused by malware and by exploit kits in the huge volumes of logs generated by commonly used devices such as firewalls and Web proxy servers.

This chapter is a retelling of a presentation titled "Deep Impact: Recognizing Unknown Malicious Activities from Zero Knowledge"[*1] given as part of the Briefings sessions at the Black Hat Europe 2018 international security conference.

2.2 Background

In most cases at present, the following methods are used to detect malicious activities, including malware infections.

- Pattern matching (including blacklists and whitelists)
- Behavioral analysis
- Event correlation

However, sophisticated and unknown attacks can circumvent these solutions. And even with attacks that are not particularly sophisticated or unknown, detection rules for pattern matching, for instance, need to be changed in response to even small changes in an attack's pattern. This is because the existing detection methods are based on information that an attacker can easily alter, such as the C2 server domain name, IP address, and the executable's binary pattern. Hence, if it is possible to use detection criteria that do not rely on these existing methods and that apply to essential aspects of an attack that are difficult for attackers to alter, we can combine such criteria with existing methods to achieve an ever greater level of security.

Some of the solutions described earlier are also very expensive and thus not necessarily something that all organizations can deploy. The aim of our work, therefore, was to develop a general-purpose solution that would enable many organizations to detect threats based on the logs created by common types of servers and network devices, such as Web proxy servers, routers, and firewalls, rather than specialized equipment and security devices. These logs have rarely been used effectively in the past, with their use being limited to cases such as the following.

- Anomaly detection based on communication volume, frequency, etc.
- SIEM event correlation
- Detection using IoCs (indicators of compromise) when they are available[*2]

If we can make use of these sorts of logs, which consume valuable disk space, many organizations may be able to achieve greater levels of security without making large changes to network structure or additional investments.

One possible reason why these sorts of logs have not been put to effective use is that, although somewhat dependent on system and organizational scale, the logs themselves are very large, preventing effective analysis that involves high computational complexity. Deep learning, however, is known to be suitable for big data analysis; for example, it is capable of processing hundreds of millions of images each on the order of tens to hundreds of kilobytes in size. So if such logs can be optimized for deep learning, it may be possible to solve this problem.

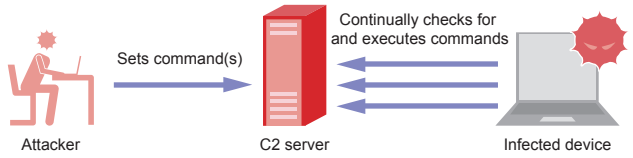

Figure 1: Bot or RAT Continually Communicating with a Command & Control (C2) Server

*1 Deep Impact: Recognizing Unknown Malicious Activities from Zero Knowledge (https://www.blackhat.com/eu-18/briefings/schedule/index.html#deep-impact-recognizing-unknown-malicious-activities-from-zero-knowledge-12276).

*2 For example, IoCs may be obtained from host names discovered via anomaly detection or analysis of suspicious devices reported by users, as well as from external reports, etc.

2.3 Detecting Communications with Malware C&C (C2) Servers

Some types of malware such as bots and RATs continually connect to their C2 servers to receive commands from attackers, which they then execute (Figure 1). Typically, they use polling intervals that range from several dozen seconds to several minutes or so. The longer the interval, the longer the program waits for any single command, which makes it difficult for the attacker to take action. Conversely, the shorter the interval, the easier it is for the attacker to act, but the easier defenders can detect the activity, since the activity will appear toward the top in a simple analysis of communication frequency broken down by destination hosts[*3]. What this boils down to is that adjusting the frequency of communication presents both an important task and a tough decision for the attacker. Meanwhile, when ordinary users within an organization communicate with external networks, such as when accessing the Web, it is rare for those communications to occur frequently or persist over a long time. Figure 2 illustrates what communication frequency looks like over the course of an hour when a user accesses harmless Web servers (left) and when malware is continually communicating with a C2 server (right). Different communication patterns almost always arise, so we thought that if our system could learn the differences, it would be able to detect malware communications. This method does not rely on DNS name, IP address, URL, and the like, so it should detect malware that existing detection methods miss.

Here, we divide the logs up by client and server and count communications in 1-minute buckets for each 1-hour period. We thus convert the logs into pseudo 60-dot images on which we perform image recognition using CNNs (convolution neural networks), a class of neural network used in deep learning. It is known that, depending on the model used, CNNs can outperform human recognition accuracy, and we therefore transform the logs into images and use this class of network in the hopes that this will prove more effective than other deep learning models.

Our training dataset is constructed as follows. We use 1.5 million "images" created from Web proxy logs as our benign sample set. For our malicious samples, we use no actual (in-the-wild) malware communication patterns and instead emulate patterns of periodic communications. With this approach, simply by generating a range of conceivable malicious patterns and training our model on them, we should be able to detect malware even for which no real-world samples are available. This is what we mean by the term "zero

	0	1	2	3	4	5	6	7	8	9	(min.)
0	9	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	8	1	0	0	0	0	
20	0	0	0	0	0	0	0	0	0	0	
30	0	0	0	0	0	0	0	0	0	0	
40	0	0	0	0	0	0	0	0	0	0	
50	0	0	0	0	0	0	0	0	0	0	

(1) To a legitimate Web server

	0	1	2	3	4	5	6	7	8	9	(min.)
0	1	0	1	0	1	0	1	0	1	0	
10	1	0	1	0	1	0	1	0	1	0	
20	1	0	1	0	1	0	1	0	1	0	
30	1	0	1	0	1	0	1	0	1	0	
40	1	0	1	0	1	0	1	0	1	0	
50	1	0	1	0	1	0	1	0	1	0	

(2) To a C2 server

Figure 2: Illustration of Communication with a Legitimate Web Server (left) and Continual Communication with a C2 Server (right)

*3 Some recent malware samplesreceive sleep times from a C2 server. Only when the attacker is active, sleep times are short and communication is frequent. At other times, the programs sleep for long periods. These sorts of techniques can make it difficult to detect the anomalous communication based on, for example, daily average times.

knowledge" in the subtitle of our Black Hat presentation. We emulate patterns for a wide range of intervals, from 3 seconds up to 12 minutes (Figure 3). As a special case, we also generate patterns that include sleeps of several minutes following several minutes of continuous activity (Figure 4). Additionally, to account for patterns that differ only slightly from the ones we have come up with and to better resist CNN attacks[*4], we also (a) apply rotations that shift each dot in the generated patterns along a number of intervals and (b) randomly set existing values to zero. We thus generate a total of around one million patterns.

Moving on to our test dataset, we use around 4.5 million images constructed from Web proxy logs (in the same manner as for our training dataset) as our benign samples. For our malicious samples, we use images constructed from logs of malware communications taken from in-the-wild incidents to see if our system can detect these. Our investigation covers the following malware families[*5].

- PlugX
- Asruex
- xxmm
- himawari/ReadLeaves
- ChChes
- Elirks
- Logedrut
- ursnif/gozi
- Shiz/Shifu
- Vawtrak
- KINS

Using the model we built, we were able to detect all of these malware families. And as shown below, the rates of false

	0	1	2	3	4	5	6	7	8	9	(min.)
0	20	20	20	20	20	20	20	20	20	20	
10	20	20	20	20	20	20	20	20	20	20	
20	20	20	20	20	20	20	20	20	20	20	
30	20	20	20	20	20	20	20	20	20	20	
40	20	20	20	20	20	20	20	20	20	20	
50	20	20	20	20	20	20	20	20	20	20	

Emulation of communication once every 3 seconds

...

	0	1	2	3	4	5	6	7	8	9	(min.)
0	1	0	0	0	0	0	0	0	0	0	
10	0	0	1	0	0	0	0	0	0	0	
20	0	0	0	0	1	0	0	0	0	0	
30	0	0	0	0	0	0	1	0	0	0	
40	0	0	0	0	0	0	0	0	1	0	
50	0	0	0	0	0	0	0	0	0	0	

Emulation of communication every 12 minutes

Figure 3: Emulation of Malicious Communications

	0	1	2	3	4	5	6	7	8	9	(min.)
0	1	1	0	0	1	1	0	0	1	1	
10	0	0	1	1	0	0	1	1	0	0	
20	1	1	0	0	1	1	0	0	1	1	
30	0	0	1	1	0	0	1	1	0	0	
40	1	1	0	0	1	1	0	0	1	1	
50	0	0	1	1	0	0	1	1	0	0	

2 min. of continual communication followed by 2-min. sleep

	0	1	2	3	4	5	6	7	8	9	(min.)
0	1	1	1	0	0	0	1	1	1	0	
10	0	0	1	1	1	0	0	0	1	1	
20	1	0	0	0	1	1	1	0	0	0	
30	1	1	1	0	0	0	1	1	1	0	
40	0	0	1	1	1	0	0	0	1	1	
50	0	0	0	0	1	1	1	0	0	0	

3 min. of continual communication followed by 3-min. sleep

...

Figure 4: Emulation of Malicious Communications (2)

*4 Simple Black-Box Adversarial Perturbations for Deep Networks (https://arxiv.org/abs/1612.06299).

*5 We do not obtain patterns simply by acquiring and running malware samples in a closed environment; instead, we use patterns obtained from actual incidents in which malware was connected to a C2 server. This is because the sleep times observed when malware is connected to an in-the-wild C2 server may differ from those observed when it is simply run in a closed environment (see footnote *3).

positives for our benign samples were also low. So we believe our system will be effective if we filter out these FQDNs using a whitelist.

- Benign sample set 1
  Accuracy: 1,565,139/1,566,109 (99.94%)
  False positive FQDNs: 64/246,190
- Benign sample set 2
  Accuracy: 1,540,419/1,541,050 (99.96%)
  False positive FQDNs: 72/243,106
- Benign sample set 3
  Accuracy: 1,528,936/1,529,617 (99.96%)
  False positive FQDNs: 65/243,185

That said, it is conceivable that Web pages that frequently reload, such as Web mail interfaces and sports sites, could register as false positives. So to reduce false positives when the system is in operation, we can take steps like excluding such sites by whitelisting them or regarding communication with a Web server as legitimate when alerts from many users are raised for that same destination.

Figures 5–9 show examples of communication patterns successfully detected from actual malware communications. It is evident that communications from the actual samples are not perfectly periodic, but the system takes the differences in stride and detects the patterns using deep learning. Logedrut

	0	1	2	3	4	5	6	7	8	9	(min.)
0	6	6	0	3	6	3	0	6	6	0	
10	3	6	3	0	6	6	0	3	7	2	
20	0	6	6	0	3	7	2	0	6	6	
30	0	3	8	1	0	6	6	0	3	8	
40	1	0	6	6	0	3	8	1	0	6	
50	6	0	4	8	0	0	7	5	0	5	

	0	1	2	3	4	5	6	7	8	9	(min.)
0	96	95	92	96	95	97	96	97	101	95	
10	97	93	95	96	95	98	92	93	95	101	
20	96	95	94	93	88	98	95	97	97	96	
30	97	88	94	96	94	101	98	97	97	96	
40	95	95	91	93	91	101	96	100	97	89	
50	92	94	96	98	94	98	98	92	94	95	

Figure 5: PlugX Communication Pattern

	0	1	2	3	4	5	6	7	8	9	(min.)
0	0	0	0	0	0	0	0	0	0	4	
10	0	2	0	2	0	2	0	0	2	0	
20	0	2	2	0	2	0	0	2	0	2	
30	0	2	0	0	2	0	2	0	2	0	
40	0	2	0	0	2	0	0	2	0	2	
50	2	0	2	0	2	0	2	2	0	2	

Figure 6: Asruex Communication Pattern

	0	1	2	3	4	5	6	7	8	9	(min.)
0	1	0	1	1	0	1	1	0	1	1	
10	0	1	1	0	1	0	1	0	1	0	
20	1	0	1	0	1	0	0	1	1	0	
30	0	1	1	0	0	1	1	0	0	1	
40	1	0	0	1	0	1	0	1	0	1	
50	0	1	0	1	0	0	1	1	0	0	

Figure 7: Elirks Communication Pattern

	0	1	2	3	4	5	6	7	8	9	(min.)
0	0	0	0	0	0	0	1	0	0	0	
10	0	0	0	0	0	0	0	1	0	0	
20	0	0	0	0	0	0	0	0	0	1	
30	0	0	0	0	0	0	0	0	0	0	
40	1	0	0	0	0	0	0	0	0	0	
50	0	1	0	0	0	0	0	0	0	0	

Figure 8: Logedrug Communication Pattern

	0	1	2	3	4	5	6	7	8	9	(min.)
0	1	0	0	0	1	1	0	0	0	1	
10	1	0	0	1	1	0	0	0	1	1	
20	0	0	0	1	1	1	0	0	1	0	
30	1	0	0	1	0	1	0	0	1	0	
40	1	0	0	1	0	0	0	0	0	0	
50	0	0	0	0	0	0	0	0	0	0	

Figure 9: Vawtrak Communication Pattern

(Figure 8) communicates infrequently, only once every 12 minutes, but it was still detected as being distinct from the benign sample. In the case of Vawtrak (Figure 9), no communications occur in the final 16 minutes, but the system detects the malware in cases like these as well.

For space reasons, we do not show all malware families and model details here. Further details can be found in our slides[*1] on the Black Hat Europe 2018 website. Since we are not building complicated models here, we believe the models can be trained on CPU-based systems.

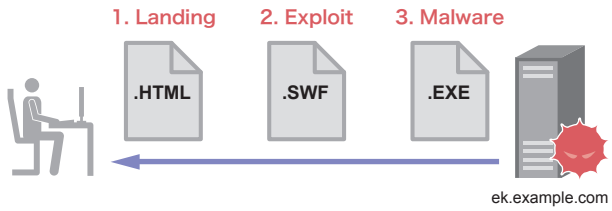

Figure 10: Sequence of Content-Types Sent by an Exploit Kit Server

Substance	Content-Type	URL path & parameters
Landing page	text/html	/? NTI0OTU5&RCDUIv&oJhtJNm=dGFraW5n&wouMDc=Y2FwaXRhbA==&JgtXjOEttlAHrl=Y2FwaXRhbA==&TKCcodYFxdiy=dGhpbmdz&tNDodvGjF=Y2FwaXRhbA==&pHtonQrvp=bG9jYXRlZA==&kl345dfdfg234fsd=UDQTpjkGELQNmyN9ZAF1G9P2s3EeBzhWZiMHT-RTZZA4QrZSQR7Rt3VzyxrckQPskg1TH6ml&pWjLICBUIUSRlw=Y2FwaXRhbA==&nR45dsgd54lsCs=xXrQMvWfbRXQDJ3EKvjcT6NAMVHRGUCL2YqdmrHXefjaf1WkzrfFTF_3ozKATASG6_ZtdfJ

Substance	Content-Type	URL path & parameters
Flash Exploit	application/x-shockwave-flash	/? NTQ0NjEw&zWuWFX&lskPeVWn=dW5rbm93bg==&NCDmQdmxCxapA=dW5rbm93bg==&eLCxfNVxDhHqBH=Y29uc2lkZXI=&nzZHzkCNdL=cmVwb3J0&HZELKhjPUenym=cG9wdWxhcg==&nR45dsgd54lsCs=wnrQMvXcKxXQFYbDKuXDSKZDKU7WG0aVw4-dhMG3YpjNfynz1ezURnL1tASVVFiRrbMdKL&kl345dfdfg234fsd=VYOQfk20LUKgEzm9sJVFhBo66tjUmDmBCd1JLX-UeLMg9DqZOSHbIL0Vz0zLMRQIgigECy&rZpDUeqxIDnMQL=bG9jYXRlZA==&LENxPZQZ=cmVwb3J0

Substance	Content-Type	URL path & parameters
Malware	application/x-msdownload	/? MjEwNzA1&mTONXmiGJttk&nR45dsgd54lsCs=wXrQMvXcJwDQDobGMvrESLtGNknQA0KK2lv2_dqyEoH9fWnihNzUSkr16B2aCm3W&UEiQzsUEYQeeS=Y2FwaXRhbA==&jeeGWAgbhZSFoHh=bG9jYXRlZA==&KRssZN=bG9jYXRlZA==&BWeciQaXKEgAey=bG9jYXRlZA==&SOymAmL=cG9wdWxhcg==&uLNyyCiGt=cG9wdWxhcg==&wlNBeZFOQXgP=dW5rbm93bg==&kl345dfdfg234fsd=_fcpKeRXaVKziULVLwczyllbUVJFpqj6i0SAmxDPhcGD_hKEUQ1M-5KREYFmmF7F

Figure 11: Example of Rig Exploit Kit's Content-Type Sequence

Substance	Content-Type	URL path & parameters
Landing page	text/html	/sm/

Substance	Content-type	URL path & parameters
Flash loader script	application/x-javascript	/sm/swfobject.js

Substance	Content-Type	URL path & parameters
IE Exploit	text/html	/sm/main.html

Substance	Content-Type	URL path & parameters
Java Exploit	application/java-archive	/sm/NeIsFp.jar

Substance	Content-Type	URL path & parameters
Malware	application/octet-stream	/dwm.exe

Figure 12: Example of KaiXin Exploit Kit's Content-Type Sequence

2.4 Exploit Kit Detection

When a PC accessing the Web is redirected to an exploit kit, the exploit kit server sends content in the order shown in Figure 10.

1. Landing page: Identifies the PC's Web browser environment and loads the next stage of exploit content. May also include an exploit(s) for the browser itself. The Content-Type is text/html.
2. Exploit content: Content file containing exploit(s) for the browser and its plug-ins. Content-Types include application/x-shockwave-flash, application/x-java-archive, application/x-silverlight-app, application/pdf.
3. Malware: If the previous stage's exploit(s) succeeds, malware that infects the PC itself is loaded. In most cases, the Content-Type is application/octet-stream or application/x-msdownload.

Figures 11 and 12 show examples of the Content-Type sequences when, respectively, Rig Exploit Kit and KaiXin Exploit Kit are observed. The figures show that their Content-Type transitions are indeed as described above.

On the other hand, we can think of almost no cases in which normal Web browsing would produce these sorts of sequences in Content-Type sent by a server. In large-scale Web services, for example, dedicated servers are typically set up to handle each Content-Type, so content that is subject to exploits, such as Flash and Java, and HTML content like landing pages tend to come from different servers, as shown in Figure 13. And in cases where a single server hosts all of a service's content, data like images and CSS, which recent exploit kits do not use all that much, come from the same server, as shown in Figure 14.

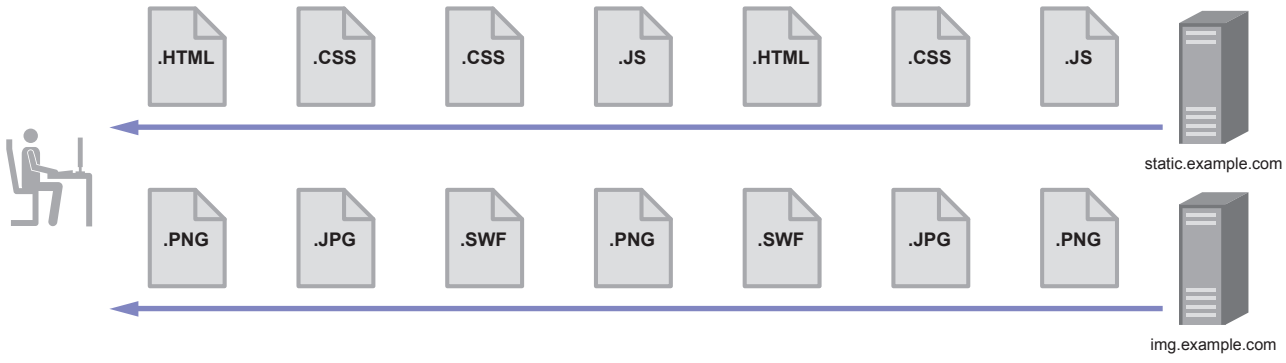

Figure 13: Example Content-Type Sequences for a Web Service with Separate Servers for Content-Types

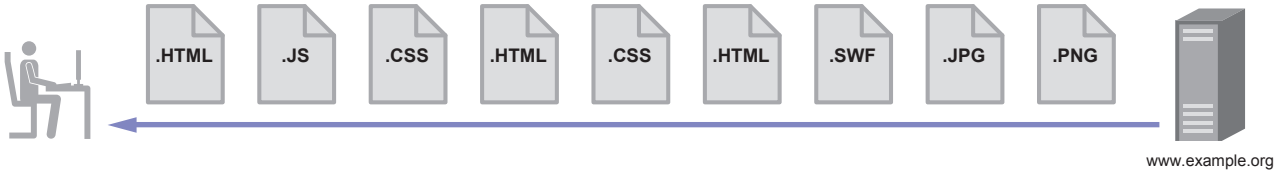

Figure 14: Example Content-Type Sequence for a Service Hosted on a Single Web Server

Given the above, we thought that if analysis of Web proxy logs could differentiate between the sequences for exploit kit servers and the sequences for normal Web server connections, it might be possible to detect exploit kits without relying on techniques like pattern matching. The Content-Type sequences peculiar to the exploit kits mentioned above are strongly related to the fundamental functions by which exploit kits force a Web browser to run exploits and infect a PC with malware. We should, therefore, be able to detect unknown exploit kits that operate in a similar manner. For the same reason, it should not be easy for an exploit kit author to evade detection by altering the sequence.

To differentiate between sequences, we use a class of neural networks called RNNs (recurrent neural network), which are used in natural language processing and the processing of time-series data such as video and audio streams. So first, we need to convert the Web proxy logs into a form that an RNN model can process. We split the logs into individual client PC – destination server pairs and convert series of requests and responses into sequences[*6]. To improve noise tolerance, we eliminate duplicate Content-Types from within each sequence. We also limit sequences to a length of five[*7], deleting any lines beyond that. Finally, we convert each line in the sequence to an 84-dimension vector. The first 83 slots in the vector represent the Content-Type in one-hot encoding, and the final slot is a flag indicating whether the referer and request URL contain the same domain name.

Our training dataset comprises a benign sample of around 580,000 sequences constructed from some 3.9 million lines from Web proxy logs, and a malicious sample of around 300,000 sequences designed to emulate conceivable exploit kit patterns. Instead of using patterns observed in the wild for our malicious sample, we generate a comprehensive range of patterns representing content sequences that could conceivably be produced by exploit kits. Figure 15 shows examples of such pseudo-sequences. Our sample includes

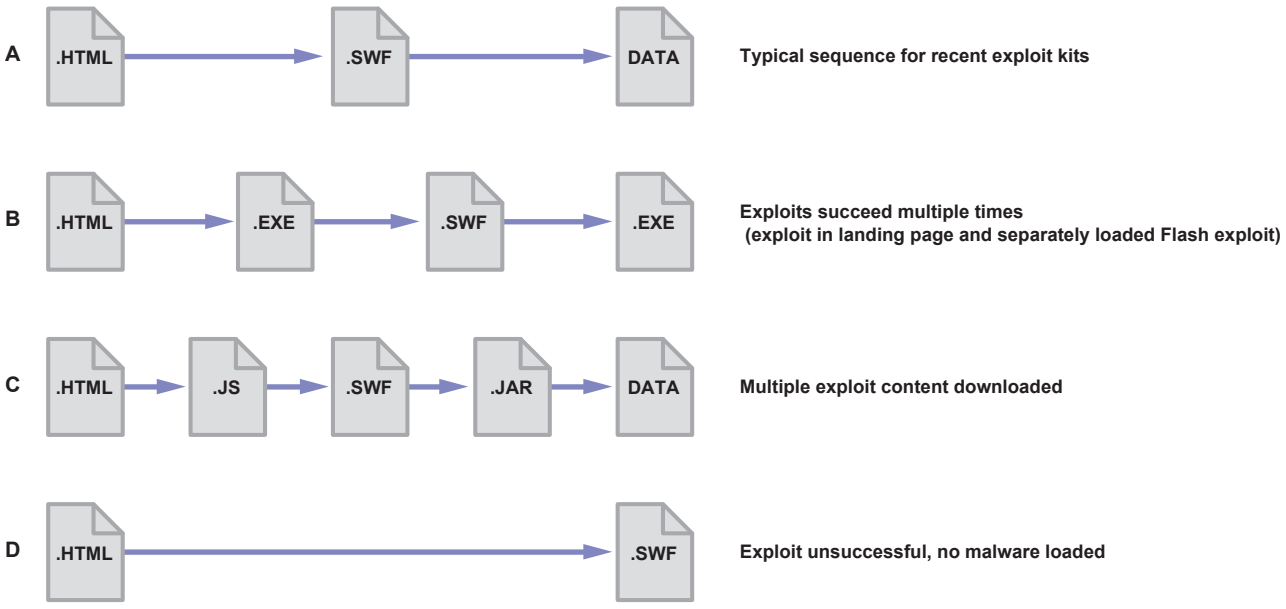

Figure 15: Examples of Generated Pseudo-sequences for Exploit Kits

*6 To be precise, we split log lines up according to client PC – destination server pairs, and then further split them into Web browser sessions (being the series of requests and responses caused by a Web browser in order to display the Web page at a given URL). The Web proxy logs we used have each Web browser session recorded separately. In more common environments, it is possible to split up Web proxy logs according to fields such as the timestamp.

*7 With the exploit kits we have observed recently, in almost no cases is content sent by the server more than five times. However, if such cases were to rise in future, we think the upper limit would need to be raised.

sequences that represent cases where several types of exploit content are loaded, cases where an exploit succeeds multiple times in a row, and cases where the exploit is unsuccessful and no malware download takes place.

To test our model, we used a malicious sample constructed from actual communication data for the following 14 exploit kits, and a benign sample of around 1.7 million sequences constructed from Web proxy logs for a time period that differs from that of the training set.

- Rig
- Nebula
- Terror
- Sundown
- KaiXin
- Neutrino
- Angler
- Nuclear
- Magnitude
- Fiesta
- Sweet Orange
- Goon
- Infinity
- Astrum

The model we built was able to detect all of the exploit kits listed above. And as shown below, false positive rates for our benign samples were also relatively low.

- Benign sample set 1
  Sequences: 562,390
  False positives: 642
  Accuracy: 0.9988
- Benign sample set 2
  Sequences: 574,452
  False positives: 681
  Accuracy: 0.9988
- Benign sample set 3
  Sequences: 576,294
  False positives: 639
  Accuracy: 0.9988

We have confirmed that using a whitelist of around 15 lines can halve the number of false positives listed above. When applying a system like ours to production environments, we would recommend combining it with other methods to narrow down the alerts, such as whitelists, host reputation, anomaly detection, and automated sandbox analysis.

Our slides[*1] available on the Black Hat Europe 2018 website also present a method of identifying Rig Exploit Kit from Web proxy logs using an MLP (multilayer perceptron) model. This method focuses on features of individual exploit kits' URLs, so while it is not suited to detecting unknown exploit kits, it is useful for identifying known exploit kits and tracking their variants. Using it with the RNN-based exploit kit detection method described here can improve the accuracy of detection of known exploit kits.

Hiroshi Suzuki

Malware & Forensic Analyst, Office of Emergency Response and Clearinghouse for Security Information, Advanced Security Division, IIJ
As a member of IIJ-SECT, Mr. Suzuki is a malware analyst and a forensic investigator. He has dedicated over 13 years to the areas. As a frequent speaker and trainer for international conferences, he has given presentations and trainings at Black Hat (USA, Europe and Asia) and FIRST TC multiple times.

Hisao Nashiwa

Threat Analyst, Office of Emergency Response and Clearinghouse for Security Information, Advanced Security Division, IIJ
Mr. Nashiwa is a member of IIJ-SECT, which is IIJ's private CSIRT.
His work includes incident response, malware analysis and network traffic analysis, and he has thus been investigating malicious activities over nine years. He has been researching cyber crimes such as those involving exploit kits and malware for many years and has expertise in malware analysis. He is a frequent conference speaker and has given talks and hands-on training sessions multiple times at international conferences such as Black Hat and FIRST TC.

Designing a Large-scale Email System

3.1 Introduction

In April 2017, we conducted a full system overhaul for IIJ Secure MX Service (SMX), which was launched in October 2006. The SMX service's primary focus is on providing email gateway functionality. Before customers' systems receive emails, the emails first pass through IIJ's email servers, where email-borne threats are filtered out using a variety of technologies, including virus filtering, spam filtering, sender authentication filtering, backscatter filtering, and sandbox filtering. The safe emails are then delivered to customers' email systems.

Over 10 years have passed since launch, and the landscape for emails services has changed substantially over that time, in terms of the volume and size of emails that pass through as well as other factors, from server specs through to email system requirements. SMX has seen system expansions on top of expansions to cope with those changes and thus now has a vastly different makeup from the system that it started out as. In many cases, however, fundamental aspects of system architecture can be traced all the way back to launch, and we have thus felt for quite a while now that the system has its limitations.

In our recent overhaul, therefore, we revised the entire system from the architecture up. This report describes the design of the SMX email system, particularly its delivery system, along with some background to the revisions.

3.2 Challenges and Goals of the System Overhaul

To begin with, we set a number of goals in view of issues with the old delivery system.

3.2.1 Review of the Architecture

The first goal was to revise the old architecture, which had been expanded excessively. Classic large-scale email systems commonly use architectures that line up simple mail servers (message transfer agent, MTA) in series as shown in Figure 1. Pre-overhaul, SMX's delivery system also used a similar architecture.

The biggest benefit of multistage MTA architectures like this is the high extensibility. Delivery system functionality can be expanded easily by linking in MTAs with the desired additional functionality. Also, because the MTAs are connected via SMTP, the standard protocol for email transmissions, linking in products from different vendors does not raise any concerns in terms of interface compatibility, and it is generally rare for product mixing and matching to cause any problems.

This method of expanding a system, however, comes with side effects and should thus not be overdone. The biggest side effect of concern is the increases in storage I/O and operating costs associated with adding more MTAs. In multistage MTA delivery systems, received data is written to storage every time an email passes through an MTA,

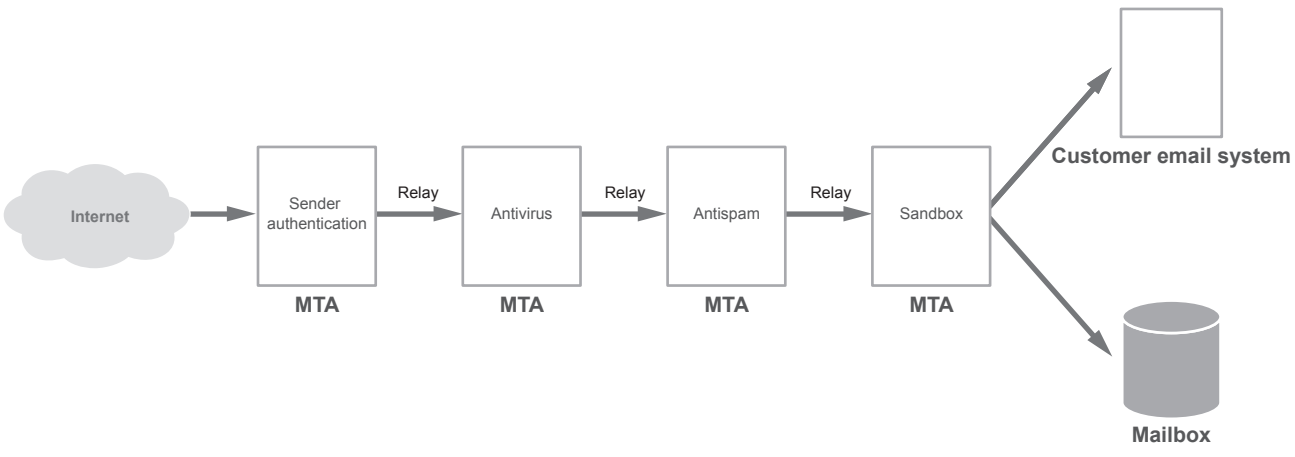

Figure 1: Architecture of a Classical Email System

resulting in storage I/O volumes several times that of the email size. Emails with virtually the same content are written to storage over and over, so taken as a whole, this sort of delivery system architecture results in a plethora of unnecessary storage I/O operations. When network storage solutions such as NAS and SAN are included, these I/O operations alone put a strain on the storage network, creating a bottleneck for the delivery system as a whole. Not only was storage a bottleneck for the old SMX delivery system, the system was also unable to take full advantage of advances in hardware speed and capacity because CPU and memory resources were being left idle.

Also, lining up different MTAs in series means filling the delivery system with a mix of various MTAs that operate in different ways. Not only does this increase the knowledge required—in terms of, e.g., how to view the logs, perform tasks, and deal with problems, and the build procedures to follow when expanding the delivery system—each MTA can also create forks in the delivery route, send out notification emails, and so on, which complicates the flow of emails within the delivery system and makes it tough to understand the delivery system as a whole.

With SMX, the delivery system was repeatedly expanded as scale increased, making for a complicated and expensive delivery system, further expansion of which had become difficult.

3.2.2 Better Filtering

The second goal was to improve the accuracy of virus filtering (antivirus) and spam filtering (antispam). Antivirus and antispam functions are crucial and constitute the dual centerpiece of email security services.

While many security vendors provide both antivirus and antispam services and products, the security industry is subject to rapid change; new attack methods constantly arise, and vendors are constantly developing new technologies to counter them. What this means is that Vendor A's product may offer high detection accuracy on one day, while Vendor B's service may offer high detection accuracy on another, and then the situation may suddenly change when Vendor C releases a new product offering superb accuracy. Put differently, this also means that sticking with any one vendor's engine comes with the risk of declining detection accuracy as the technology used by the engine becomes obsolete. In view of this, we felt we needed a mechanism for keeping virus and spam email detection accuracy high.

3.2.3 Avoiding Over-reliance on Any One Vendor

The third goal was to avoid over-reliance on any specific vendor. To provide a wide range of functionality, the SMX system incorporates numerous vendor products and services. Although many of the products and services offered are appealing, to avoid vendor lock-in, we felt we needed to retain control over how reliant we are on any single offering.

It is not uncommon, particularly among foreign vendors, for a company to be suddenly bought out by a competitor, resulting in its products and services being terminated. Suddenly eliminating the ability to use a product has no small impact on customers who incorporate it into their systems. Although we cannot reduce the impact of this to zero, we do need to take steps to minimize it.

3.3 Single-stage MTA for Effective Use of Hardware Resources

In overhauling the delivery system in line with the above goals, we first sought to harness the plentiful CPU and memory resources afforded by the increasingly high performance of commodity server hardware and to reduce storage I/O, which was a bottleneck for the system overall as well as a factor in costs.

The architecture we arrived at is the polar opposite of the previous one. In short, all processing will be completed within a single, multifunctional MTA, with this single-stage setup eliminating the unnecessary relaying of emails between MTAs (Figure 2). With this architecture, email is only written to storage once, so storage I/O operations are greatly reduced compared with the old delivery system, which wrote the same content to storage multiple times.

Also in the old delivery system, the storage I/O bottleneck meant that the CPU resources of most servers were left idle while CPU utilization rates were high on only some servers, where high-CPU-load tasks such as virus scans were being run. By using identically configured MTAs in parallel, previously idle CPU resources can be diverted for high-load processing tasks, making it possible to more efficiently use CPU resources across the entire delivery system (Figure 3).

Because SMX employs multiple antivirus and antispam engines, the single-stage MTA architecture requires several engines to be loaded into any one server's memory. In general, antivirus and antispam engines hold a lot of data in memory and thus tend to consume a lot of memory resources. Loading several such engines into memory necessitates the sort of total memory capacity that older servers could not accommodate, but the increasingly high performance of commodity server hardware , as mentioned above, has made this sort of setup possible.

3.4 Making Antivirus/Antispam Engines Interchangeable

Next, we designed the overall system to allow the antivirus and antispam engines to be replaceable at any time.

IIJ does not develop antivirus or antispam engines in-house but instead provides this functionality by incorporating such engines from security vendors into the delivery system. This means that IIJ cannot directly address any problems that arise with detection accuracy. Taking the opposite perspective, however, by taking advantage of the ability to cut obsolete technology loose and swiftly incorporate fresh technology, we aimed to design a system that is not tightly tied to any specific antivirus or antispam engine.

First, we evaluated the antivirus and antispam engines of each security vendor. We ran scans for viruses and spam on emails received by honeypots run by IIJ, accumulating

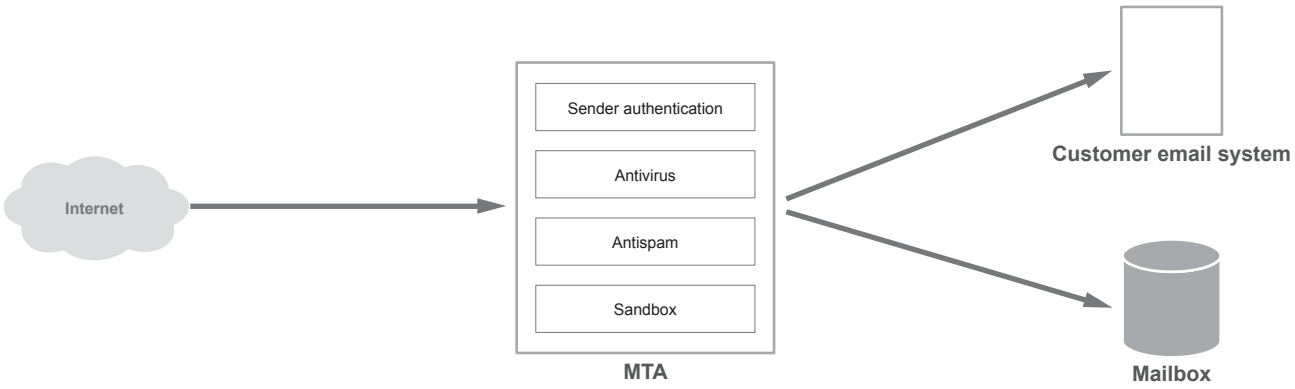

Figure 2: Architecture of the Overhauled Email System

statistics on detection performance over several months to facilitate comparisons. Virus and spam emails can exhibit epidemic-like qualities, so any evaluation of countermeasures over short periods can be heavily influenced by detection performance with respect to whatever spam campaign*1 happened to be active at the time, and thus may not provide a proper idea of long-term detection performance. This is why we used a fairly long validation period. During the validation period, we received product guidance from a range of security vendors. The approaches taken for antispam engines, in particular, are quite distinct across the different products available, and the results of our validation exercise are very interesting in that they reflect the differences in approach. It was an arduous process, but it gave us an understanding of the detection accuracy and tendencies of each engine.

To achieve enhanced detection accuracy, we combine, respectively, multiple antivirus engines and multiple antispam engines. We then combine the results given by each to arrive at the final detection results.

It is evident from the results of this validation process, and also simply from intuition gained through day-to-day operations, that no single engine is an out-and-out winner in terms of how long it takes to detect virus and spam emails after they begin to circulate. Engine A may be quicker to detect such emails during one campaign, while Engine B may be quicker during another, and so on. Combining multiple engines enables us to reduce detection misses during the early stages of a campaign. In the case of antispam engines, in particular, we combine engines that use different approaches so that the weaknesses of any one engine are compensated for by other engines.

Our main aim in combining multiple engines was to improve detection accuracy, but it also had secondary effects. One is the reduction of scan errors. It is not uncommon for virus and spam scans to fail to complete successfully because of corrupted email headers/attachments or deliberate content manipulation, for instance. Running scans with multiple engines makes it possible to significantly reduce the number of emails that cannot be scanned at all. Also, in rare cases, antivirus engines and antispam engines can crash when scanning specific emails or attachments. In such cases, you have the option of disconnecting the engine experiencing the problem as an emergency measure to allow emails to actually make it through the system. It also makes it possible to minimize the impact of a security vendor being bought out and its products becoming unavailable.

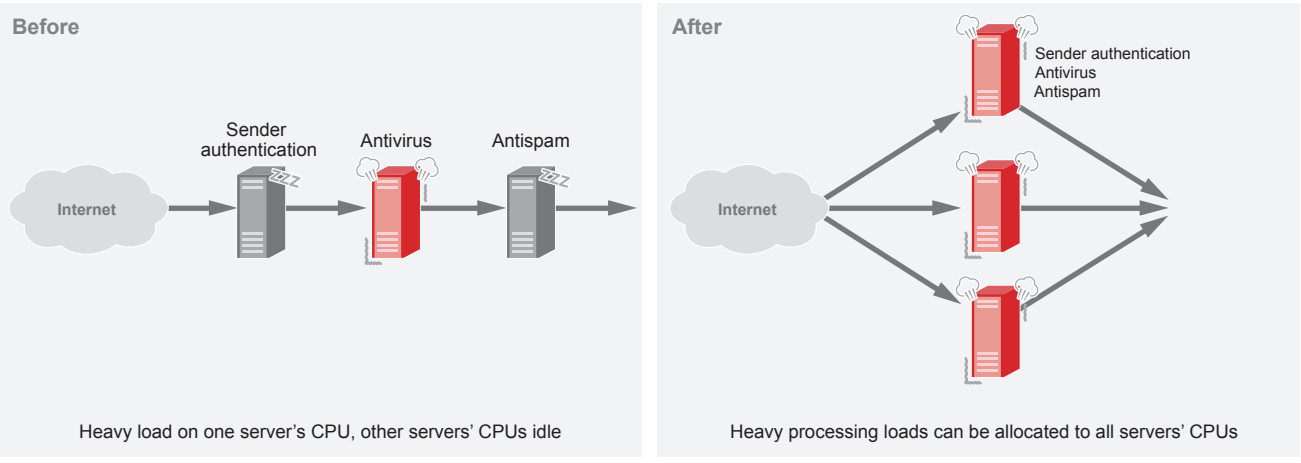

Figure 3: Effective Use of Hardware Resources

*1 Mass mailout of identical or similar spam emails.

3.5 Decision to Develop MTA In-house

The biggest problem we had when overhauling the delivery system was how to implement this design. This is what prompted our decision to develop the MTA in-house, but other methods were available to us, such as combining open source MTAs like Postfix and Sendmail, or the use of MTAs made by MTA vendors.

Postfix and Sendmail provide an interface called Milter, which provides advanced functionality and makes it possible to easily and safely implement email control and rewrite capabilities. On the other hand, given the architecture, adding Milter onto the system would result in a large I/O overhead in particular. And we also had to admit that it lacks the functionality for realizing a complex system like SMX. Although one idea could be to modify Postfix or Sendmail directly, the cost of keeping up with updates to the official package is far greater than you might imagine.

The option of adopting a vendor-produced MTA seemed like a very realistic one. Several MTA vendors develop MTA products for ISPs and large-scale senders. In these products as well, the idea of performing all processing in a single stage, as described above, is the mainstream approach. They also allow various security vendors' antivirus and antispam engines to be swapped in and out, and they allow flexible, extremely fine-grained customizations that meet the detailed requirements of large-scale delivery systems, and as such, MTAs from vendors are far more versatile and powerful than the open-source offerings. For us, this would have been the quickest and easiest way of creating the delivery system we had in mind. Indeed, many ISPs and large-scale senders use vendor MTAs, and parts of SMX's old delivery system also incorporated vendor MTAs.

The only, and biggest, concern we had with taking on a vendor MTA was that SMX would become one with that particular MTA. In a single-stage MTA architecture, the MTA itself is the delivery system. And the entire system beyond the delivery system, from the service specifications through to the operation procedures, would depend heavily on the MTA. In other words, what the MTA can do, SMX can do; and vice versa.

IIJ continues to actively expand SMX's functionality with a close eye not only on customer feedback but also on the latest trends in the email and security industries. So the SMX delivery system may at times require functionality that other operators do not require. In such situations, getting a vendor to add features that an MTA lacks is generally difficult. MTA vendors serve many customers, so they will inevitably put priority on developing functionality that many customers require, and on functionality that key customers want. Developing highly idiosyncratic and niche functionality will naturally be of low priority. Vendor MTAs would appear to be suitable when the required specifications are clear and when few specification changes are likely occur in future, but whether such offerings could support our proactive approach to expanding SMX is an unclear.

Vendor MTAs also come with the risk of the vendor being bought out. Because the system as a whole is heavily reliant on the MTA, were such a corporate acquisition to occur, the impact would be immense relative to any impact that might result in relation to the antivirus and antispam engines. Indeed, several acquisitions involving MTA vendors and products have taken place in the past few years. In absolute terms, not many vendors develop MTAs, so in percentage terms, this is a risk that cannot be ignored.

The final option is in-house development, but this is also not an easy option. MTAs that support the huge levels of communication flows seen at the ISP level require extremely high levels of stability, robustness, and performance. Such a system would also need to provide the diverse functionality and flexibility of a system like SMX. And the technical capabilities to support future expansions in functionality would also be needed.

The decision to develop such an MTA from scratch in-house is perhaps not a very easy one. IIJ, however, does have experience and knowhow in developing many email system components itself. It also has a battery of reliable development teams, and this is why we were able to pull the trigger on developing our system in-house—the risk was high, but we also saw that we had much to gain.

3.6 Overhaul Outcomes

3.6.1 Achieving our Development Goals

The overhaul project encompassed the entire system, including the MTA component. More than a full year passed before the first release was out. It was the biggest development project I have ever been part of.

A lengthy development period and repeated testing were needed to complete the system, but ultimately it met all of the goals we set. Our flexible, versatile MTA made it possible to create a delivery system with a single-stage MTA architecture. In accord with our design, we reduced storage I/O operations, which had been an issue with the old system, thereby enhancing the performance of the delivery system overall. Major reviews of the virus filters and spam filters also resulted in improved detection accuracy. And constantly keeping an eye on the detection rates of each engine we use has also enabled us to take swift action whenever detection rates change.

Developing an MTA in-house has freed us from the risk of MTA vendors being acquired, and it has also enabled us to minimize the effect of security vendors being acquired because we are now able to swap engines in and out. Although perhaps quite obvious, I think the most important outcome of this system overhaul is that we have laid foundations that will enable IIJ to run its own services as it sees fit, essentially immune to the vicissitudes of vendor acquisitions.

3.6.2 Secondary Benefits

We also realized some secondary benefits beyond the goals that we originally set.

First, troubleshooting speed improved. When problems occur with a vendor MTA, details of the condition are reported to the vendor and a fix requested. But if it is unclear how to reproduce the problem, or if the details cannot be passed to the vendor because they contain customer data, then it may take considerable time for the vendor to confirm the problem or, as is often the case, the vendor may be unable to ascertain what the problem is at all. With in-house development, however, the operations and development teams can work closely with one another. This means that the root cause of any problems, in particular, can be identified extremely quickly, and proper provisional and permanent responses implemented.

Also, and while this is not something that affords direct comparisons, I feel that developing the system in-house resulted in strong motivation levels for the team. When developing a system that incorporates MTAs or other products from vendors, the development team develops ways of interfacing with those vendor products, but this frequently involves a seemingly futile struggle against unclear product specifications and is often not much fun on a personal level. In-house development, on the other hand, entails a far, far greater amount of work, and we were concerned about the burden this would place on the development team. Yet, while the team was certainly busy, I never felt an atmosphere of fatigue or exhaustion setting in. In the end, I think there is something exciting about building a large-scale system on your own and seeing it gradually come together.

We released the new system in April 2017, and we spent a full year migrating over to it from the old one. We have received various opinions and requests since release, prompting us to add functions and fix issues, and I think we have been able to so swiftly make these improvements precisely because we developed the system in-house.

I would note that my intention is not to criticize systems that use vendor products across the board. Both approaches have their advantages and disadvantages, so you need to select the right balance depending on the situation. Finally, I would also note that SMX incorporates many vendor products, including some from foreign vendors, into its system, and we work closely with vendors on a daily basis as we provide our services to customers.

Here, I have described the design of the SMX delivery system. With its newly acquired architecture, we will continue to evolve the SMX service going forward.

Takahiko Suzuki
Senior Engineer, Service Development Section, Application Service Department, Network Division, IIJ
Since joining IIJ in 2004, Mr. Suzuki has continued to work on email service development.
He is the developer of yenma, an open-source sender authentication filtering program.

May 2019 Vol.42

IIJ
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-MKTG020-0040

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/