# IIR

Internet
Infrastructure
Review

July 2022
Vol. 55

Periodic Observation Report

## Email Security in the Modern Age —Password-protected ZIPs and DMARC Sender Authentication

Focused Research (1)

## Creating a mac_apt Plugin (Part 2)

Focused Research (2)

## Challenges and Solutions in Storage on Service Infrastructure

IIJ
Internet Initiative Japan

# Internet Infrastructure Review

July 2022 Vol.55

## Executive Summary

IIR Vol.54 (https://www.iij.ad.jp/en/dev/iir/054.html), published March 2022, offered a focused research report analyzing the changes in Internet connectivity that occurred when Russia annexed Crimea in 2014. That report appeared during Russia's contemporary invasion of Ukraine, which began on February 24, and a month or so after it was published, NetBlocks reported[*1] that on May 1, after a blackout of several hours, Internet connectivity in Russia-controlled areas had been restored but was being routed via Russian telecom providers.

It was surprising to see this happen immediately after our report on the Crimean case appeared in the IIR, and doubly surprising that, unlike what happened in 2014, the change of network happened at such an early juncture. Perhaps this is proof that control over the information flowing across the Internet is even more important than it was some eight years ago.

That this is happening now after the period of war and conflict that was the 20th century is behind us is quite shocking. Some observers have been highly vocal about the need to review the use of technology in the context of security, and the same goes for the information and communications technology that we are responsible for. I pray in earnest that the technologies available to humanity are used to enrich our lives, and I hope we can continue to move forward in that vein.

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.

Our periodic observation report in Chapter 1 looks at messaging. In our periodic observation report about messaging a year ago, we described how the Emotet virus was encrypting itself in ZIP files and thus running rampant. A widespread practice in Japan when attaching files to an email is to encrypt them in a ZIP archive and send the password to that archive in a separate email. This practice has made it difficult for companies to prevent Emotent infections. As of January 2022, IIJ adopted a policy of generally rejecting emails to which an encrypted ZIP file is attached. IIJ also tightened its DMARC policy for sender authentication in December 2021. The in this edition report discusses the background to IIJ adopting these security enhancements for its own email systems along with the issues it faced, and we hope this serves as a valuable reference for anyone managing an email system in their own organization.

The focused research report in Chapter 2 is the second part of our two-part series on mac_apt, a forensic analysis framework being developed for macOS. This installment explains how mac_apt plugins are created. It goes into the details of the data stored by mac_apt and discusses the actual design and implementation of a plugin. We encourage you to read through both installments.

Our second focused research report in Chapter 3 looks at challenges and solutions to do with storage on IIJ's service infrastructure. The amount of data generated and processed around the world is growing at an astounding pace, supported by advances in computer processing power and network data transfer speeds. All of that data is essentially amassed on what we call storage systems, and just like computers and networks, storage systems have also been evolving in a significant way. We at IIJ also operate a lot of storage systems designed to ensure that data is kept safe and sound. The report here provides a basic explanation of storage and discusses the FC-SAN and storage operation technologies that IIJ makes extensive use of.

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's MVNO Council, and in June 2021, he became a vice-chairman of the association.

*1 NetBlocks (https://netblocks.org/reports/internet-disruptions-registered-as-russia-moves-in-on-ukraine-W80p4k8K).

# Email Security in the Modern Age —Password-protected ZIPs and DMARC Sender Authentication

## 1.1 Introduction

We reported on trends in spam and virus numbers in the periodic observation report in IIR Vol. 51 last June (https://www.iij.ad.jp/en/dev/iir/051.html). Two points to note in that context are that, at the time, we had received up to 200 times the amount of spam received in the previous year, and that the virus Emotet was encrypting itself in ZIP files to avoid virus scans and thus running rampant.

In this issue, we look at two security enhancements IIJ has undertaken to protect itself from such threats. One is to eliminate the use of encrypted ZIP files, and the other is to tighten up DMARC. We would like to see all readers do the same and hope this article will be helpful in that regard.

## 1.2 IIJ Blocks Encrypted ZIP Files

### 1.2.1 Background to Blocking Encrypted ZIP File

IIJ changed its company-wide policies to, as a general rule, block encrypted ZIP files attached to emails as of January 26, 2022[*1].

A common practice in Japan when attaching files to emails is to encrypt them in a password-protected ZIP file and send the password to that file in a separate email as a way of preventing files from being missent[*2]. But not only is this largely ineffective in preventing information from being missent, it also has the fatal flaw of circumventing virus scans, and CISA (the US's Cybersecurity and Infrastructure Security Agency) recommends blocking the receipt of such files for this reason[*3].

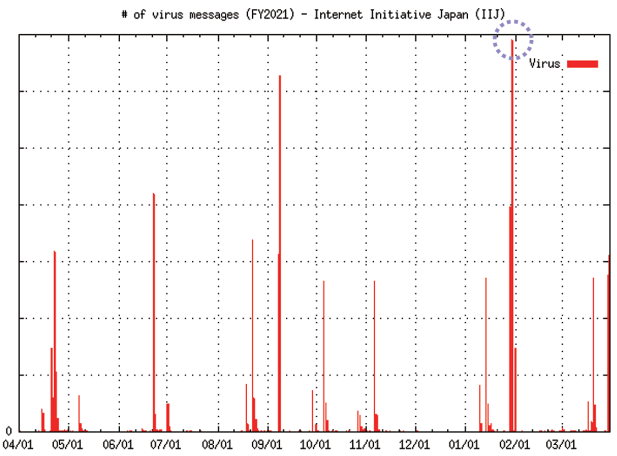


Figure 1: Viruses Arriving at IIJ Honeypots (April 2021 – March 2022)

And as mentioned, the quite rampant virus Emotet uses this method, and we can expect other viruses that do the same to appear in the future. To protect not only IIJ's internal data but also the important data our customers and business partners entrust us with, we made a management decision to not leave this risk unchecked, in accord with which we have been implementing a top-down response.

### 1.2.2 Preparing to eliminate encrypted ZIP files

This change of policy proceeded as follows.

- Information Systems Department explained the situation to the Risk Management Office and management
- Management explained the risks and outlined its plans internally
- Information Systems Department began working on steps to implement countermeasures
- Risk Management Office worked on a unified set of rules
- Explanation provided to internal departments and schedule mapped out
- Explanation provided to customers and business partners
- Policy changed

It took about a year from when management outlined its plans until the final change of policy took place. Our careful preparations have meant that we have experienced no major disruptions in the six months or so since.

### 1.2.3 Effect of Eliminating Encrypted ZIP Files

Coincidentally, the weekend following the day on which we began blocking encrypted ZIP files brought confirmation that Emotet, which was supposed to have been taken down, had made a comeback. It also appeared prominently at the end of January in IIJ's honeypots, as Figure 1 shows. IIJ's setup blocks the virus at the gateway so it is never received, and we thus had early confirmation of this being highly effective. Our internal network was thus kept safe.

*1 IIJ's press release: "Changes to our operations regarding password protected .zip files sent as an attachment followed by a separate email containing the password (PPAP)" (https://www.iij.ad.jp/en/ppap/).

*2 A practice also known as PPAP in Japan.

*3 CISA, "Emotet Malware" (https://www.cisa.gov/uscert/ncas/alerts/aa20-280a).

The lead up to implementing this series of policy shifts involved coordinating among a diverse range of internal stakeholders, including Risk Management, Information Systems, Sales, and Public Relations. Overhauling long-standing methods can be painful at times, but putting off responses to risks will solve nothing. We recommend taking swift action before serious incidents occur.

### 1.2.4 Are There Alternatives?

The topic of alternatives goes hand in hand with discussion about abandoning encrypted ZIP files. As we discussed in IIR Vol. 51, any alternative will have its pros and cons.

IIJ understands this well and accordingly offers three ways of sharing files with external parties.

The first is company-wide shared online storage, and this is probably the most orthodox method discussed as an alternative. It has its risks, however. An internal control-related drawback to archiving emails in online storage is that it can be difficult to trace the files later, and it may also confound efforts to detect insider crime because large amounts of files can be exfiltrated via a single URL.

The second is to send attachments as is. Encrypted ZIP files are not very effective in the case of missent emails, and they circumvent virus scans, so one may conclude that simply sending the files as is would be fine. In the case of some business partners, external Web access from within the recipient's company systems is not permitted, so sending the files directly is an option in those cases.

The third is to continue using the traditional encrypted ZIP method. This constitutes an exception. Email interactions involve both the sending and receiving of emails, so there are still cases in which some organizations and business partners have no choice but to use the old method. We enable exceptions in such cases, subject to a full understanding of the risks and approval by the relevant organizational heads.

Combining these three methods with an email audit system serves to ameliorate the drawbacks of each (Table 1).

To reiterate, blocking encrypted ZIP files across the board will mean you are no longer exposed to viruses such as Emotet. And even while this work is being done, attackers will be aiming at their next target. Changing company-wide policies can take time, so we recommend you begin working on a response right away.

## 1.3 IIJ Tightens DMARC Policy

### 1.3.1 Background to Tightening of DMARC Policy

IIJ introduced its DMARC policy in 2013, and for some time used p=none, which is a declaration to external domains that nothing should be done about emails that fail DMARC authentication. Even with p=none, it is possible to publish a DMARC record and receive DMARC reports. Aggregating the statistical data in those reports makes it possible to detect email spoofing, which is useful in terms of protecting your domain's brand. The IIJ Secure MX Service, a SaaS offering, also provides users with the ability to automatically aggregate DMARC reports and review the statistical data.

| | Advantages | Risks/Problems | | |
|---|---|---|---|---|
| 1. Use online storage | • Virus scanning can be done on the storage side of things<br>• Can be effective against information being missent in some cases<br>• Can send and receive large files | • File tracking/tracing is difficult, which is a drawback for internal control<br>• Difficult to detect internal crimes where files are exfiltrated | + | Email audit system |
| 2. Send files attached as is | • Virus scanning can be done on the gateway<br>• No additional equipment or investment required<br>• Applicable to any environment, regardless of who the recipient is | • No countermeasures against information being missent | | |
| 3. Conventional encrypted ZIP method (exception granted) | • No need to change conventions | • Avoids virus scans, so is unprotected and risky<br>• Emails are rejected by some recipient systems<br>• Cumbersome, requires effort on the receiving end<br>• Hinders operational streamlining and automation | | |

Table 1: Typical Advantages and Risks of Alternatives

Phishing emails and sender-spoofing emails that cleverly evade spam filters have become common in recent years, necessitating a multifaceted approach to the various email threats out there. As a provider of corporate email security SaaS, IIJ made the decision to change its policy to p＝quarantine to strengthen internal email security.

### 1.3.2 Sender Authentication

Before discussing IIJ's implementation of DMARC, we will first review sender authentication technology. SPF, DKIM, and DMARC are widely used for sender authentication across the globe, with each being defined by an RFC (Table 2).

Table 3 shows the three available DMARC policies. It is important to note that a DMARC policy is merely a request from the sender asking email recipients to treat emails in a certain way. The receiving system will not necessarily handle emails as requested. A DMARC policy has no effect unless the email analysis system on the receiving end has functionality or filters for performing DMARC validation, but the point is that companies can demonstrate the validity of their domain by declaring a DMARC policy to the outside world.

Until fairly recently, the email filtering process had been left up to recipients, but with DMARC policies, we now have a revolutionary framework for email filtering that lets senders ask recipients to treat emails that fail DMARC authentication in a certain way.

### 1.3.3 Preparing to Implement Sender Authentication

To change your DMARC policy, you simply change your DMARC record, but your SPF record must be published and your DKIM signature implemented before doing so. A DMARC policy evaluates whether email is valid based on SPF and DKIM, so the point is to not create a situation in which your employees send out non-sender-authenticated email.

The following preparations are key here.

(1) Ensure employees are aware that company emails should only ever be sent from company email addresses (and the company email system)
(2) Consolidate email exit points to reduce the cost of implementing sender authentication
(3) (After implementing a DMARC policy) Regularly check the DMARC reports

Let's look at these three steps in detail.

#### ■ (1) Employee awareness

At IIJ, several types of emails are sent out from the iij.ad.jp domain.

- Business emails sent by IIJ employees
- Notification emails sent out by system devices
- Announcements sent to customers

| p= | When dmarc=fail, the recipient is asked to |
|---|---|
| none | do nothing |
| quarantine | quarantine the email |
| reject | reject the email |

Table 3: DMARC Policies

| Sender authentication | RFC | Overview |
|---|---|---|
| SPF | 7208*1 | By publishing an SPF record, administrators can declare to external parties that email sent from the IP addresses listed in the record is legitimate. |
| DKIM | 6376*2 | Electronically signing an email makes it possible to verify whether the content of the email has been tampered with. |
| DMARC | 7489*3 | Senders specify how recipients should handle emails that fail SPF or DKIM authentication. |

*1：https://datatracker.ietf.org/doc/rfc7208
*2：https://datatracker.ietf.org/doc/rfc6376
*3：https://datatracker.ietf.org/doc/rfc7489

Table 2: Characteristics of Sender Authentication

Previously, employees sent business emails out from various different internal servers. This was addressed by having the Information Systems Department manage the internal email system exit points and, ultimately, enacting a policy prohibiting employees from sending out emails using @iij.ad.jp other than from the internal email system, continuously monitoring transmissions sent from inside the company out onto the Internet, and sending notifications to users who violate the policy telling them to discontinue that behavior.

There were also problems with the emails being sent from the various internal systems. At IIJ, a number of services are operated by different departments, and email alerts and notifications were being sent out from all over the place. Currently, the department that controls each service sets up a unified exit point for all emails sent out by the service.

In addition, some group companies and regional offices had devices set up to send out email alerts using iij.ad.jp as the sender address, so we notified the people responsible for those devices and asked them to follow the IIJ policy.

### (2) Consolidation of email exit points

As noted above, the consolidation of email exit points is an important consideration in performing sender authentication.

In SPF records, email source IP addresses can be given in CIDR notation, so you can keep records from becoming verbose by masking off the email exit points using a /32 or /31 prefix, or possibly by going as wide as /28 or so, and this also allows you to reduce operating costs when adding or changing exit points. When there are a number of IP addresses from which email can be emitted, repeatedly using the "include" mechanism can hamper the SPF evaluation process or cause it to fail unintentionally because information has been left out (note that RFC 7208 limits the number of DNS lookups resulting from "include"terms in an SPF record to 10).

### (3) Regularly checking DMARC reports

IIJ receives DMARC reports (rua) from all over the place. That is, we regularly receive reports from a range of organizations informing us of the results of sender authentication on emails their systems have received from @iij.ad.jp. Since we control the email exit points, we can basically assume that any emails not actually coming from those exit points are spam, but we have observed some cases in which this was not true, and based on this information we have been gradually asking the relevant business units to update their policies.

- Alert emails
  There were cases in which SPF/DMARC was failing because the envelope from address in network devices, server monitoring systems, and the like had been arbitrarily set to iij.ad.jp.
- Promotional and recruiting emails
  These often use external SaaS offerings, and in some cases the envelope from address in the system had been set to iij.ad.jp and emails sent out with no consideration given to sender authentication.

## 1.3.4 Decision to Change DMARC Policy

The DMARC policy can also be progressively tightened to "quarantine" or "reject". IIJ decided to implement a p=quarantine DMARC policy because it still allows emails to be received even if the recipient filters them according to the DMARC policy.

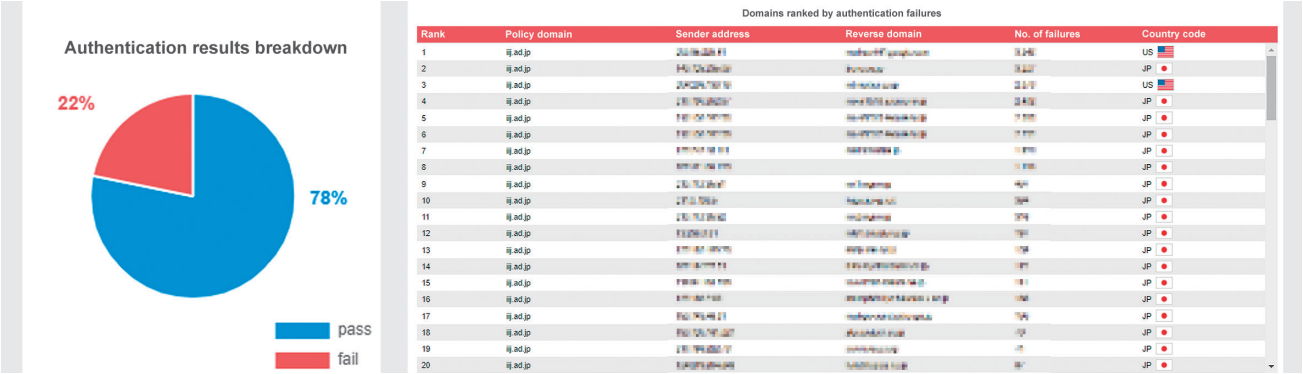


Figure 2: February 2022 DMARC Report Summary for iij.aj.jp
Authentication Results Breakdown (left), Top 20 Domains for Authentication Failures (right)

### Actual procedure

The procedure itself is very simple: you simply overwrite your DMARC record from p=none to p=quarantine. We omit the details here, but before doing the overwrite, you need to have an SPF record published and a DKIM signature implemented.

It takes a mere 10 minutes, even counting the time taken to confirm the record after it has been overwritten (Figure 3).

Within DMARC records, the adkim and aspf parameters, respectively for DKIM and SPF, allow you to specify the alignment mode for the authentication identifier (domain). They are set to either r (relaxed mode) or s (strict mode), the choice of which determines whether the organization's domain and the Header From field must match.

For example, if the organization's domain is example.com and the Header From field is system@alert.example.com, then SPF in alignment mode r will return a pass authentication result. In s mode, however, the domains must match exactly, so DMARC will fail in this case. The rua field specifies where DMARC reports are to be sent, and the aggregated results shown in the previous section represent a visualization of the reports received at dmarc-rua@dmarc.iij.ad.jp.

### Post-change impact

IIJ changed its DMARC record on December 15, 2021.

After the change, we kept a close eye on inquiries and other developments but noted no major disruptions. Some people may consider a change of DMARC record to be a difficult undertaking, but we can report that it actually does not have that much of an impact, and DMARC lets you declare to email recipients that "our systems only send out emails that are in accordance with our policies, so please go ahead and quarantine or reject any non-compliant emails", so we encourage you to consider implementing it on your own systems.

In the past, IIJ's honeypots have observed a high volume of spoofed emails being sent with domains for which DMARC policies had not been declared. For organizations that need to ensure the legitimacy of email content (financial institutions, government agencies, etc.), we recommend adopting DMARC because it allows a clear distinction to be made between legitimate and spoofed emails.

During the writing of this article, we adopted a p=quarantine DMARC policy for IIJ and monitored the situation for a few weeks, and once we had determined there to be no real impact on business emails, the DMARC record was changed to p=reject as of March 23, 2022. As with the change to p=quarantine, this also did not result in any inquiries or concerns being raised internally.

## 1.4 Sender Authentication Data

### 1.4.1 Sender Authentication Adoption Rates

Two years have now passed since the global rise of telework, and 2021 was a year that witnessed many cyberattacks based on emails using the Emotet virus.

Figures 4–6 show the aggregated sender authentication results as a percentage of total for email services provided by IIJ for the period April 2021 to March 2022.

```
$ dig _dmarc.iij.ad.jp txt

;; ANSWER SECTION:
_dmarc.iij.ad.jp. 3600 IN TXT "v=DMARC1; p=none; adkim=s; aspf=s; rua=mailto:dmarc-rua@dmarc.iij.ad.jp"
```

DMARC record for iij.ad.jp before the change

```
$ dig _dmarc.iij.ad.jp txt

;; ANSWER SECTION:
_dmarc.iij.ad.jp. 3600 IN TXT "v=DMARC1; p=quarantine; adkim=s; aspf=s; rua=mailto:dmarc-rua@dmarc.iij.ad.jp"
```

DMARC record for iij.ad.jp after the change

**Figure 3: DMARC Record for iij.ad.jp Before and After the Change**

Comparing the figures here with those reported in IIR Vol. 51 (https://www.iij.ad.jp/en/dev/iir/051.html), we note that the DKIM and DMARC pass ratios have increased. The DKIM pass ratio is up a few percentage points from last time, which may possibly indicate a moderate increase in the adoption of SaaS by companies amid the telework era. With the DMARC pass ratio also on the rise, the picture is one of interest in sender authentication increasing, albeit gradually.

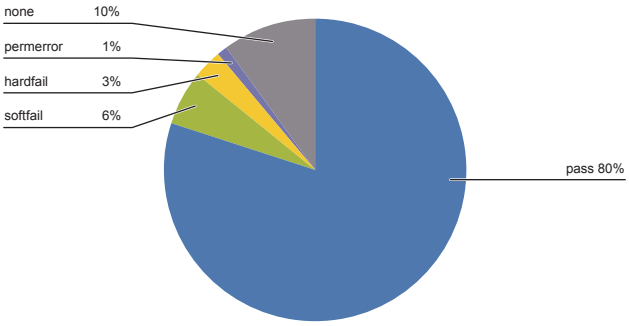


Figure 4: Breakdown of SPF Authentication Results

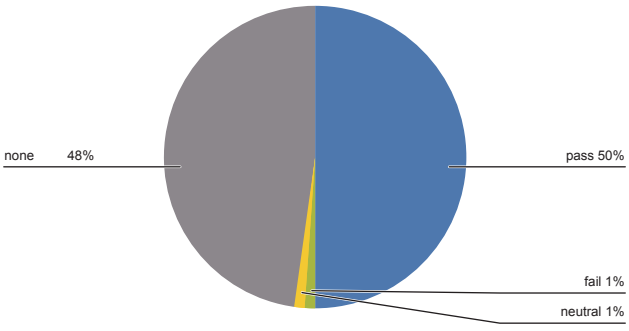


Figure 5: Breakdown of DKIM Authentication Results

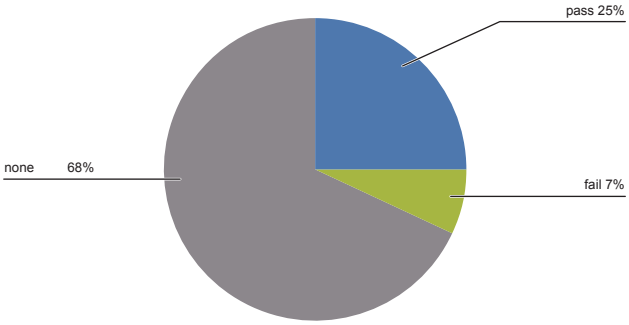


Figure 6: Breakdown of DMARC Authentication Results

**Isamu Koga**

Manager, Operation & Engineering Section, Application Service Department, Network Division, IIJ
Mr. Koga joined IIJ in 2007. He is engaged in the operation of email services and investigates email-related trends in the wild. To keep customers' email boxes safe, he communicates information about the latest attack methods, trends in spam, and countermeasures.

**Yusuke Imamura**

Engineer, Operation & Engineering Section, Application Service Department, Network Division, IIJ
Mr. Imamura joined IIJ in 2015. He is engaged in the operation of email services. His past experience working at IIJ Europe benefits him in fulfilling his global role.

# Creating a mac_apt Plugin (Part 2)

In IIR Vol.54, we took a look at the demo plugin provided by the mac_apt forensic analysis framework for macOS to understand the basic structure of mac_apt plugins[*1]. In this installment, I discuss the data stored in "~/Library/Caches/<Application Bundle ID>/Cache.db" and go over the implementation of a mac_apt plugin for analyzing this artifact. If you haven't read the article in IIR Vol.54 yet, you may find it easier to follow along if you go back and read that first.

## 2.1 What Information Does Cache.db Hold?

First, we look at the information recorded in Cache.db. This is a cache of HTTP/HTTPS data transfers made via APIs like NSURLRequest. The Cache.db file is an SQLite-format database and holds the data in the database tables shown in Tables 1–5. The cache data is basically stored in this database, but data above a certain size is stored as a file in the fsCacheData directory (Figure 1). This is apparently called the "CFURL Cache", because the database table name starts with "cfurl_cache".

| Column | Data type |
|---|---|
| entry_ID | The entry ID |
| response_object | BLOB in plist format. Holds the URL accessed, HTTP status, response header, etc. |
| request_object | BLOB in plist format. Holds the URL accessed, access method, request header, etc. |
| proto_props | Unknown (used for cache control?) |
| user_info | Unknown (all NULL to the extent this author has checked) |

Table 1: cfurl_cache_blob_data

| Column | Data type |
|---|---|
| entry_ID | The entry ID |
| isDataOnFS | Flag indicating the format of the response data from the server<br>0: Response body from the server is stored in receiver_data.<br>1: Response body from the server is stored in a file. |
| receiver_data | Holds the response data from the server (isDataOnFS = 0) or the name of the file in which it has been stored (isDataOnFS = 1).<br>The file is saved in "~/Library/Caches/<App Bundle ID>/fsCacheData/". File names are in UUID format. |

Table 2: cfurl_cache_receiver_data

| Column | Data type |
|---|---|
| entry_ID | The entry ID |
| version | Unknown (all 0 to the extent this author has checked) |
| hash_value | Unknown |
| storage_policy | Unknown (all 0 to the extent this author has checked) |
| request_key | URL of the destination accessed |
| time_stamp | Timestamp of when the URL was accessed |
| partition | Unknown (all NULL to the extent this author has checked) |

Table 3: cfurl_cache_response

| Column | Data type |
|---|---|
| schema_version | The schema version (all 202 to the extent this author has checked, except in cases where the table itself does not exist) |

Table 4: cfurl_cache_schema_version

| Column | Data type |
|---|---|
| cfurl_cache_response | Maximum value for entry_ID of cfurl_cache_response? |

Table 5: sqlite_sequence

*1 Focused Research (1) in Internet Infrastructure Review (IIR) Vol.54, "Creating a mac_apt Plugin (Part 1)" (https://www.iij.ad.jp/en/dev/iir/054.html).

Looking at this information, you can check not only the date and time of program data transfers and the destination URLs but also the responses received from the servers. User and program activity histories are crucial to forensic analysis, so this artifact is a very useful source of information.

## 2.2 Designing a Plugin

### 2.2.1 Data to Acquire

Before we get into creating the plugin, let's look a little closer at what information we can get from the artifact.

Tables 1–3 indicate we can obtain the data transfer timestamp, the destination URL, the client's request method and header, the server's HTTP status and response body, and the response body from the server. The data in these tables are linked by a key called entry_ID. And the "<Application Bundle ID>" part of the file path to where Cache.db is stored identifies the program that made the transfer. We need the plugin to collect this information and save it as an analysis result.

Note that the response_object and request_object in the cfurl_cache_blob_data table (Table 1) are stored as plist-format BLOBs (Figure 2)[*2]. This data should not be left in plist format when stored in the analysis results. Instead, it should be parsed to make it easy for the analyst to determine the contents.

```
% ls -alR ~/Library/Caches/com.apple.osascript
total 128
drwxr-xr-x    4 macforensics  staff    128  2 10 17:36 .
drwx------+ 150 macforensics  staff   4800  4 27 15:12 ..
-rw-r--r--@   1 macforensics  staff  65536  5 19  2021 Cache.db
drwxr-xr-x@   4 macforensics  staff    128  2  2  2021 fsCachedData

/Users/macforensics/Library/Caches/com.apple.osascript/fsCachedData:
total 344
drwxr-xr-x@ 4 macforensics  staff     128  2  2  2021 .
drwxr-xr-x  4 macforensics  staff     128  2 10 17:36 ..
-rw-r--r--@ 1 macforensics  staff  116503 11  9  2020 2B1680C0-DAE0-4EA0-9EC0-C4FC7F86A8C0
-rw-r--r--@ 1 macforensics  staff   53755  2  2  2021 A391D5EC-9FCF-4993-A0AF-EEF2C871EF6A
```

Figure 1: File Structure

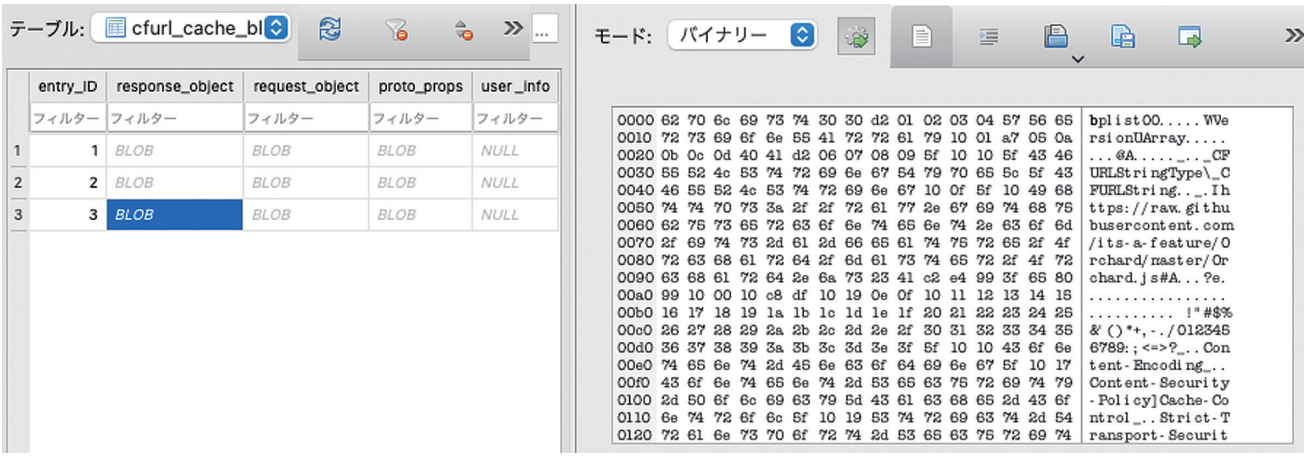


Figure 2: Sample response_object

*2 The "bplist00" sequence at the start is the plist binary format's magic number.

Figure 3 shows the result of exporting the request_object data using DB Browser for SQLite[*3] or other software, parsing it with plutil, the standard macOS command. It looks like the data we need for forensic analysis is contained in elements 18 and 19 of the Array. Element 18 is the HTTP request method, and 19 is the HTTP request header.

Element 19 also holds Base64-encoded data in its "__hhaa__" field. This is a plist in binary format; Figure 4 shows the decoded content. It is the same as the HTTP request header and thus we can conclude that it does not need to be included in the analysis results. The response_object data can also be examined in the same manner (Figure 5). In this case, element 3 of the Array holds the HTTP status and element 4 holds the HTTP response header.

```
% plutil -p cfurl_cache_blob_data__request_object_2.bin
{
  "Array" => [
    0 => 0
    1 => {
      "_CFURLString" => "https://www.example.com/"
      "_CFURLStringType" => 15
    }
    2 => 60
    3 => 1
    4 => "__CFURLRequestNullTokenString__"
    5 => 1
    6 => 134
    7 => "__CFURLRequestNullTokenString__"
    8 => "__CFURLRequestNullTokenString__"
    9 => 1
    10 => 0
    11 => 0
    12 => 0
    13 => 0
    14 => 0
    15 => -1
    16 => "__CFURLRequestNullTokenString__"
    17 => 2
    18 => "GET"
    19 => {
      "__hhaa__" => "

YnBsaXN0MDDTAQIDBAYIXxAPQWNjZXB0LUVuY29kaW5nVkFjY2VwdF8QD0FjY2VwdC1
MYW5ndWFnZaEFXxARZ3ppcCwgZGVmbGF0ZSwgYnKhB1MqLyqhCVVqYS1qcAgPISg6PF
BSVlgAAAAAAAABAQAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAXg=="
      "Accept" => "*/*"
      "Accept-Encoding" => "gzip, deflate, br"
      "Accept-Language" => "ja-jp"
    }
    20 => "__CFURLRequestNullTokenString__"
    21 => "__CFURLRequestNullTokenString__"
  ]
  "Version" => 9
}
```

**Figure 3: Result of Parsing the request_object Data**

```
% echo
YnBsaXN0MDDTAQIDBAYIXxAPQWNjZXB0LUVuY29kaW5nVkFjY2VwdF8QD0FjY2VwdC1
MYW5ndWFnZaEFXxARZ3ppcCwgZGVmbGF0ZSwgYnKhB1MqLyqhCVVqYS1qcAgPISg6PF
BSVlgAAAAAAAABAQAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAXg== | base64 -d |
plutil -p -
{
  "Accept" => [
    0 => "*/*"
  ]
  "Accept-Encoding" => [
    0 => "gzip, deflate, br"
  ]
  "Accept-Language" => [
    0 => "ja-jp"
  ]
}
```

**Figure 4: Result of Parsing the request_object's __hhaa__ Field**

```
% plutil -p cfurl_cache_blob_data__response_object_2.bin
{
  "Array" => [
    0 => {
      "_CFURLString" => "https://www.example.com/"
      "_CFURLStringType" => 15
    }
    1 => 628074307.809312
    2 => 0
    3 => 200
    4 => {
      "__hhaa__" => "

YnBsaXN0MDDdAQIDBAUGBwgJCgsMDQ4QEhQWGBocHiAiJCZcQ29udGVudC1UeXBlVEV
0YWdXWC1DYWNoZVNBZ2VfEBBDb250ZW50LUVuY29kaW5nVlNlcnZlcldFeHBpcmVzXU
NhY2hlLUNvbnRyb2xURGF0ZV5Db250ZW50LUxlbmd0aF1BY2NlcHQtUmFuZ2VzVFZhc
nldTGFzdC1Nb2RpZmllZKEPXxAYdGV4dC9odG1sOyBjaGFyc2V0PVVURi04oRFcIjMx
NDc1MjY5NDcioRNTSElUoRVWNTY4Nzk3oRdUZ3ppcKEZXkVDUyAobnliLzFEMkYpoRt
fEB1UaHUsIDAzIERlYyAyMDIwIDA5OjA1OjA3IEdNVKEdXm1heC1hZ2U9NjA0ODAwoR
9fEB1UaHUsIDI2IE5vdiAyMDIwIDA5OjA1OjA3IEdNVKEhUzY0OKEjVWJ5dGVzoSVfE
A9BY2NlcHQtRW5jb2RpbmehJ18QHVRodSwgMTcgT2N0IDIwMTkgMDc6MTg6MjYgR01U
AAgAIwAwADUAPQBBAFQAWwBjAHEAdgCFAJMAmACmAKgAwwDFANIA1ADYANoA4QDjAOg
A6gD5APsBGwEdASwBLgFOAVABVAFWAVwBXgFwAXIAAAAAAAACAQAAAAAAAAoAAAAAA
AAAAAAAAAAAABkg=="
      "Accept-Ranges" => "bytes"
      "Age" => "568797"
      "Cache-Control" => "max-age=604800"
      "Content-Encoding" => "gzip"
      "Content-Length" => "648"
      "Content-Type" => "text/html; charset=UTF-8"
      "Date" => "Thu, 26 Nov 2020 09:05:07 GMT"
      "Etag" => ""3147526947""
      "Expires" => "Thu, 03 Dec 2020 09:05:07 GMT"
      "Last-Modified" => "Thu, 17 Oct 2019 07:18:26 GMT"
      "Server" => "ECS (nyb/1D2F)"
      "Vary" => "Accept-Encoding"
      "X-Cache" => "HIT"
    }
    5 => 1256
    6 => "text/html"
  ]
  "Version" => 1
}
```

**Figure 5: Result of Parsing the response_object Data**

*3 DB Browser for SQLite (https://sqlitebrowser.org/).

The receiver_data field in the cfurl_cache_receiver_data table (Table 2) holds the response body received from the server. This information is also useful for forensic analysis, so we save it in the analysis results. But rather than including the data in the analysis results in the form of a file saved in the fsCacheData directory, we should instead export the file. This avoids unnecessarily increasing the analysis results file size.

### 2.2.2 Data Acquisition Method

Next, we consider how to go about acquiring the data. Since Cache.db is a SQLite database, information can be retrieved using SQL queries. Given the results above, we can use the SQL query in Figure 6 to obtain the required information.

The plist data stored in the response_object and request_object can be parsed using the Python plistlib module[*4]. __hhaa__ is excluded from the parse result.

### 2.2.3 Plugin Implementation Approach

At this point, we have determined what data to obtain and how to acquire it, and based on this, we can lay out an implementation approach for the plugin as shown below.

1. Process a macOS disk image or exported artifacts.
   - 1.1 If processing a disk image, process all artifacts in all users' "~/Library/Caches/" directories.
   - 1.2 If processing exported artifacts, process all artifacts in the specified directory.
2. Save the following data in the analysis results.
   - 2.1 Data obtained via the SQL query in Figure 6
   - 2.2 Results of parsing response_object and request_object
   - 2.3 Application Bundle ID
   - 2.4 In the case of disk images, export the files in the fsCacheData directory

```
SELECT entry_ID, time_stamp, request_key, request_object, response_object, isDataOnFS, receiver_data
       FROM cfurl_cache_response JOIN cfurl_cache_blob_data USING (entry_ID)
       JOIN cfurl_cache_receiver_data USING (entry_ID)
```

**Figure 6: SQL Query for Obtaining the Required Data from Cache.db**

*4 You can also use the CommonFunctions.ReadPlist() method provided by mac_apt.

## 2.3 Creating the Plugin

Now let's start creating the plugin. The plugin discussed here was merged via a Pull Request in July 2021, so you can find it on the mac_apt GitHub repository[*5].

Here, I give an outline of what the plugin does. Please refer to the source code for further details if necessary[*6].

### 2.3.1 Properties

I set the properties as shown in Figure 7. The plugin is called CFURLCACHE. As it processes disk images and exported artifacts, it uses "__Plugin_Modes = "MACOS,ARTIFACTONLY"".

### 2.3.2 Entry Points

#### ■ Plugin_Start()

Plugin_Start () is the plugin entry point when you run mac_apt.py (Figure 8).

mac_info.users holds a list object containing the user information on the disk image. You can use this to loop through and process the artifacts for all users (line 179). But since the same home directory can be configured for use on multiple accounts, the plugin skips home directories that have already been processed (lines 180–182).

```
__Plugin_Name = "CFURLCACHE" # Cannot have spaces, and must be all caps!
__Plugin_Friendly_Name = "CFURL cache"
__Plugin_Version = "1.0"
__Plugin_Description = "Parses CFURL cache and extract date, URL, request, response, and received data."
__Plugin_Author = "Minoru Kobayashi"
__Plugin_Author_Email = "unknownbit@gmail.com"

__Plugin_Modes = "MACOS,ARTIFACTONLY" # Valid values are 'MACOS', 'IOS, 'ARTIFACTONLY'
__Plugin_ArtifactOnly_Usage = 'Provide the path to "/Library/Cache/" folder under user home'
```

Figure 7: Properties

```
173 def Plugin_Start(mac_info):
174     '''Main Entry point function for plugin'''
175     cfurl_cache_artifacts = []
176     cfurl_cache_base_path = '{}/Library/Caches/'
177     processed_paths = set()
178
179     for user in mac_info.users:
180         if user.home_dir in processed_paths:
181             continue # Avoid processing same folder twice (some users have same folder! (Eg: root & daemon))
182         processed_paths.add(user.home_dir)
183         base_path = cfurl_cache_base_path.format(user.home_dir)
184         if not mac_info.IsValidFolderPath(base_path):
185             continue
186         cache_folder_list = mac_info.ListItemsInFolder(base_path, EntryType.FOLDERS, include_dates=False)
187         app_bundle_ids = [folder_item['name'] for folder_item in cache_folder_list]
188         for app_bundle_id in app_bundle_ids:
189             cache_folder_path = os.path.join(cfurl_cache_base_path.format(user.home_dir), app_bundle_id)
190             cache_db_path = os.path.join(cache_folder_path, 'Cache.db')
191             if mac_info.IsValidFilePath(cache_db_path) and mac_info.GetFileSize(cache_db_path) > 0:
192                 ExtractAndReadCFURLCache(mac_info, cfurl_cache_artifacts, user.user_name, app_bundle_id, cache_folder_path)
193
194     if len(cfurl_cache_artifacts) > 0:
195         PrintAll(cfurl_cache_artifacts, mac_info.output_params, '')
196     else:
197         log.info('No CFURL cache artifacts were found!')
```

Figure 8: The Plugin_Start() Function

*5 cfurl_cache.py (https://github.com/ydkhatri/mac_apt/blob/3e823ee36bdf133c4de3503848435033ee20943d/plugins/cfurl_cache.py).

*6 Note that, if the plugin has been updated since this writing, line numbers in this article may not correspond to those in the current source code.

For unprocessed home directories, the mac_info object's IsValidFolderPath() method is used to check for the existence of the home directory in the disk image (line 184), and the ListItemsInFolder() method is used to create a list of directories in "~/Library/Caches/" (line 186).

The IsValidFilePath() method is then used to check for the existence of Cache.db files in each of the directories (line 191). If the file exists, the ExtractAndReadCFURLCache() function is called to obtain analysis results for Cache.db and export the artifact file (line 192). The analysis results are stored in cfurl_cache_artifacts as a list object.

Once all of the Cache.db files have been analyzed, the results are stored using the PrintAll() function (line 195).

**■ Plugin_Start_Standalone()**

Plugin_Start_Standalone() is the plugin entry point when you run mac_apt_artifact_only.py (Figure 9).

input_files_list contains the name of the directory to be processed as specified on the command line (line 202). Beyond that, the function basically runs through the same process as Plugin_Start(), but with this entry point, the OpenAndReadCFURLCache() function is called to perform the analysis instead of the ExtractAndReadCFURLCache() function (line 212).

### 2.3.3 Data Analysis

**■ ExtractAndReadCFURLCache()**

This function opens a Cache.db file in the disk image, analyzes the data, saves the analysis results, and exports the artifact file (Figure 10).

Cache.db is opened inside the OpenDbFromImage() function (line 147). This function uses the connect() method of the SqliteWrapper class provided by mac_apt to get a connection to the SQLite database. As this class is a wrapper around the standard Python sqlite3

```
199 def Plugin_Start_Standalone(input_files_list, output_params):
200     '''Main entry point function when used on single artifacts (mac_apt_singleplugin), not on a full disk image'''
201     log.info("Module Started as standalone")
202     for input_path in input_files_list:
203         log.debug("Input file passed was: " + input_path)
204         cfurl_cache_artifacts = []
205         if os.path.isdir(input_path):
206             cache_folder_list = os.listdir(input_path)
207             app_bundle_ids = [f for f in cache_folder_list if os.path.isdir(os.path.join(input_path, f))]
208             for app_bundle_id in app_bundle_ids:
209                 cache_folder_path = os.path.join(input_path, app_bundle_id)
210                 cache_db_path = os.path.join(cache_folder_path, 'Cache.db')
211                 if os.path.isfile(cache_db_path) and os.path.getsize(cache_db_path) > 0:
212                     OpenAndReadCFURLCache(cfurl_cache_artifacts, '', app_bundle_id, cache_folder_path)
213
214         if len(cfurl_cache_artifacts) > 0:
215             PrintAll(cfurl_cache_artifacts, output_params, input_path)
216         else:
217             log.info('No CFURL cache artifacts were found!')
```

Figure 9: The Plugin_Start_Standalone() Function

```
145 def ExtractAndReadCFURLCache(mac_info, cfurl_cache_artifacts, username, app_bundle_id, folder_path):
146     cfurl_cache_db_path = os.path.join(folder_path, 'Cache.db')
147     db, wrapper = OpenDbFromImage(mac_info, cfurl_cache_db_path, username)
148     if db:
149         ParseCFURLEntry(db, cfurl_cache_artifacts, username, app_bundle_id, cfurl_cache_db_path)
150         mac_info.ExportFolder(folder_path, os.path.join(__Plugin_Name, username), True)
151         db.close()
```

Figure 10: The ExtractAndReadCFURLCache() Function

module, you can use sqlite3 methods to execute SQL queries and so forth.

The data analysis happens not in the ExtractAndReadCFURLCache() function but in the ParseCFURLEntry() function. It is set up this way so that both ExtractAndReadCFURLCache() and OpenAndReadCFURLCache(), described below, can use the same processing routine (line 149).

Finally, the ExportFolder() method is used to export the artifact file (line 150). The first argument is the folder path to export from, the second is the name of the export destination folder, and the third is the overwrite flag.

The export destination folder specified by the second argument is created in the "Export" folder that is created within the output destination folder specified on the mac_apt command line. A look at the source code of other plugins shows that they basically use "__Plugin_Name" for this. But with CFURL Cache, because there is an artifact file for each user, the user name is also included in the export destination folder name.

### ■ OpenAndReadCFURLCache()

This function opens the exported Cache.db file, analyzes the data, and saves the analysis results (Figure 11). It does not export artifact files.

Cache.db is opened inside the OpenDb() function. This function gets a connection using the mac_apt CommonFunctions class's open_sqlite_db_readonly() method. Data analysis is done by the ParseCFURLEntry() function, as noted above.

### ■ ParseCFURLEntry()

This function issues a SQL query and retrieves the required data from Cache.db (Figure 12). It also parses the acquired data and saves the analysis results.

First, it gets a list of table names, and if a cfurl_cache_schema_version table exists, it gets the schema version (lines 118–121). To the extent I have checked, only version 202 is ever used[*7].

Next, it uses the SQL query in Figure 6 to get the required data (lines 125–128). As mentioned above, the request_object and response_object data are in a binary-format

```
153  def OpenAndReadCFURLCache(cfurl_cache_artifacts, username, app_bundle_id, folder_path):
154      cfurl_cache_db_path = os.path.join(folder_path, 'Cache.db')
155      db = OpenDb(cfurl_cache_db_path)
156      if db:
157          ParseCFURLEntry(db, cfurl_cache_artifacts, 'N/A', app_bundle_id, cfurl_cache_db_path)
158          db.close()
```

Figure 11: The OpenAndReadCFURLCache() Function

*7 In some cases, depending on the macOS version, there is no cfurl_cache_schema_version table present.

plist. The data are analyzed by the ParseRequestObject() and ParseResponseObject() functions described below (lines 130–131).

The receiver_data object type depends on what it holds. If it holds the response body, it will be “bytes”. If it holds the name of the file (UUID) in the fsCacheData directory, it will be “str” (lines 132–135).

Finally, it saves the acquired data as an entry in the analysis results (lines 140–143). The plugin defines the CfurlCacheItem class to hold analysis results (Figure 13), and the entry is an instance of that class. The class has no methods; it is simply there to group the data together.

```
114  def ParseCFURLEntry(db, cfurl_cache_artifacts, username, app_bundle_id, cfurl_cache_db_path):
115      db.row_factory = sqlite3.Row
116      tables = CommonFunctions.GetTableNames(db)
117      schema_version = 0
118      if 'cfurl_cache_schema_version' in tables:
119          schema_version = CheckSchemaVersion(db)
120      else:
121          log.debug('There is no cfurl_cache_schema_version table.')
122
123      if 'cfurl_cache_response' in tables:
124          if schema_version in (0, 202):
125              query = """SELECT entry_ID, time_stamp, request_key, request_object, response_object, isDataOnFS, receiver_data
126                         FROM cfurl_cache_response JOIN cfurl_cache_blob_data USING (entry_ID)
127                         JOIN cfurl_cache_receiver_data USING (entry_ID)"""
128              cursor = db.execute(query)
129              for row in cursor:
130                  http_req_method, req_headers = ParseRequestObject(row['request_object'])
131                  http_status, resp_headers = ParseResponseObject(row['response_object'])
132                  if type(row['receiver_data']) == bytes:
133                      received_data = row['receiver_data']
134                  elif type(row['receiver_data']) == str:
135                      received_data = row['receiver_data'].encode()
136                  else:
137                      log.error('Unknown type of "receiver_data": {}'.format(type(row['receiver_data'])))
138                      continue
139
140                  item = CfurlCacheItem(row['time_stamp'], row['request_key'], http_req_method, req_headers,
141                                        http_status, resp_headers, row['isDataOnFS'], received_data,
142                                        username, app_bundle_id, cfurl_cache_db_path)
143                  cfurl_cache_artifacts.append(item)
```

Figure 12: The ParseCFURLEntry() Function

```
41  class CfurlCacheItem:
42      def __init__(self, date, url, method, req_header, http_status, resp_header, isDataOnFS, received_data, username, app_bundle_id, source):
43          self.date = date
44          self.url = url
45          self.method = method
46          self.req_header = req_header
47          self.http_status = http_status
48          self.resp_header = resp_header
49          self.isDataOnFS = isDataOnFS
50          self.received_data = received_data
51          self.username = username
52          self.app_bundle_id = app_bundle_id
53          self.source = source
```

Figure 13: Class that Holds Analysis Results

### ■ ParseRequestObject() and ParseResponseObject()

This function gets data from the request_object and response_object (Figure 14, Figure 15).

As noted above, the request_object array's 18th element holds the HTTP method, and the 19th holds the HTTP request header (lines 96–97 in Figure 14). We also exclude the __hhaa__ field (line 100 in Figure 14). Similarly, the response_object array's 3rd element is the HTTP status, and the 4th is the HTTP response header (lines 106–107 in Figure 15).

### 2.3.4 Saving the Analysis Results

### ■ PrintAll()

This function saves the analysis results in the format specified on the command line (Figure 16). cfurl_cache_info defines the column names and types used when storing the analysis results (lines 162–164). The analysis data items are collected into a list object in the same order as the items in the cfurl_cache_info definition, and in the final step, the WriteList() function writes the data to a file (lines 168–171).

## 2.4 Example of the Plugin in Action

FIgure 17 shows the analysis results from the plugin discussed here (columns to the right of Received_Data have been trimmed from the screenshot). I hope you'll agree that it is easy to examine the data once the information is organized like this.

```
94  def ParseRequestObject(object_data):
95      object_array = plistlib.loads(object_data)['Array']
96      http_req_method = object_array[18]
97      header_list = object_array[19]
98      req_headers = []
99      for header, value in header_list.items():
100         if header != '__hhaa__':
101             req_headers.append("{}: {}".format(header, value))
102     return http_req_method, "\r\n".join(req_headers)
```

Figure 14: The ParseRequestObject() Function

```
104 def ParseResponseObject(object_data):
105     object_array = plistlib.loads(object_data)['Array']
106     http_status = object_array[3]
107     header_list = object_array[4]
108     resp_headers = []
109     for header, value in header_list.items():
110         if header != '__hhaa__':
111             resp_headers.append("{}: {}".format(header, value))
112     return http_status, "\r\n".join(resp_headers)
```

Figure 15: The ParseResponseObject() Function

```
161 def PrintAll(cfurl_cache_artifacts, output_params, source_path):
162     cfurl_cache_info = [('Date', DataType.TEXT), ('URL', DataType.TEXT), ('Method', DataType.TEXT), ('Request_Header', DataType.TEXT),
163                         ('HTTP_Status', DataType.TEXT), ('Response_Header', DataType.TEXT), ('isDataOnFS', DataType.INTEGER), ('Received_Data', DataType.BLOB),
164                         ('User', DataType.TEXT), ('App_Bundle_ID', DataType.TEXT), ('Source', DataType.TEXT)]
165
166     data_list = []
167     log.info(f"{len(cfurl_cache_artifacts)} CFURL cache artifact(s) found")
168     for item in cfurl_cache_artifacts:
169         data_list.append([item.date, item.url, item.method, item.req_header, item.http_status, item.resp_header, item.isDataOnFS, item.received_data, item.username, item.
            app_bundle_id, item.source])
170
171     WriteList("CFURL cache", "CFURL_Cache", data_list, cfurl_cache_info, output_params, source_path)
```

Figure 16: The PrintAll() Function

| | Date | URL | Method | Request_Header | HTTP_Status | Response_Header | isDataOnFS | Received_Data |
|---|---|---|---|---|---|---|---|---|
| | フィルター | フィルター | フィル... | フィルター | フィルター | フィルター | フィルター | フィルター |
| 1 | 2020-11-09 01:58:45 | https://stackoverflow.com/ | GET | Accept: */*... | 200 | Content-Encoding: gzip... | 1 | 2B1680C0-DAE0-4EA0-9EC0-C4FC7F86A8C0 |
| 2 | 2020-11-09 02:10:43 | https://www.example.com/ | GET | Accept: */*... | 200 | Content-Type: text/html; charset=UTF-8... | 0 | <!doctype html>... |
| 3 | 2021-02-02 06:58:03 | https://raw.githubusercontent.com/its-a-feature/Orchard/master/Orchard.js | GET | Accept: */*... | 200 | Content-Encoding: gzip... | 1 | A391D5EC-9FCF-4993-A0AF-EEF2C871EF6A |

Figure 17: Analysis Results from the Plugin

## 2.5 Conclusion

This two-part series has walked through the creation of a mac_apt plugin. While I have provided a broad understanding of plugin structure and process flow, I certainly have not covered all of the APIs mac_apt provides. Looking at other plugins also would be a great way to further your understanding.

mac_apt is a powerful forensic analysis tool, but the best way to get support for more artifacts is to write your own plugins. As explained in the previous installment, many artifacts are in plist or SQLite format, so it's very easy to look through the data, and the most important advantage is that you can analyze the data you require for your purposes in the format of your choice.

Finally, some readers may be more interested in how to go about reading and making sense of the mac_apt analysis results than in creating plugins. I would recommend the presentation slides[*8] and analysis data[*9] from the macOS hands-on forensics workshop given at the Japan Security Analyst Conference 2022 (JSAC2022) as a useful reference in this case. The workshop used mac_apt analysis results to create a forensic timeline of malware incursion. A video of the workshop is also available[*10].

**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.

*8 Workshop slides (https://jsac.jpcert.or.jp/archive/2022/pdf/JSAC2022_workshop_macOS-forensic_en.pdf).
*9 Analysis data (https://jsac.jpcert.or.jp/archive/2022/data/JSAC2022_macos_forensic_workshop_without_malware.7z).
*10 [JSAC2022] Workshop: An Introduction to macOS Forensics with Open Source Software (https://www.youtube.com/watch?v=Mor9EpInrXM).

# Challenges and Solutions in Storage on Service Infrastructure

## 3.1 Introduction

IIJ has used storage array systems in its service infrastructure since the 2000 launch of its resource-on-demand service IBPS, the predecessor to the IIJ GIO cloud service. Both IIJ GIO and NHN (Next Host Network), a cloud system for IIJ's own services, currently use storage array systems in their infrastructure, and the capacity of those storage systems is constantly being increased.

Here, we start by describing storage in general to give the reader a deeper understanding of what it actually is. We then discuss what sort of storage and storage networks IIJ employs as it strives to provide services that customers can rely on.

## 3.2 A General Overview of Storage

### 3.2.1 What is Storage?

Storage is a general term for anything that stores data and programs. Familiar consumer storage devices include the hard disk drives (HDDs) and solid state drives (SSDs) used in personal computers, USB thumb drives used to carry data around, SD cards inserted into smartphones and digital cameras, media used to hold music and video such as CDs and Blu-ray discs, and network-attached storage (NAS) devices sometimes referred to as network-compatible HDDs. If you've been using PCs for long enough, you may at some point have experienced an HDD or SSD failure that stopped your computer from working.

The enterprise storage used in IIJ's service infrastructure is designed for high availability. It contains multiple HDDs, SSDs, and other components to ensure that, even if one component fails, that failure will not result in data being lost or access to storage being unavailable. These sorts of storage systems are known as storage array systems, disk array systems, or enterprise storage, but we usually refer to them simply as "storage".

This section focuses on HDDs and SSDs used as storage components, and explains how storage array systems work and what the network that connects a storage array system and servers looks like.

### 3.2.2 Difference between HDDs and SSDs

Both HDDs and SSDs are classified as block storage, and when present in a PC or server, they are generally formatted with a file system supported by the installed OS before being put into use.

Inside HDDs, a disk coated with a magnetic material, called the platter, is rotated at high speed, and a magnetic head reads and writes data from the platter. Because the gap between the platter and the magnetic head is narrower than a human hair, HDDs are vulnerable to impact and can break down if moved during operation. They also contain mechanically driven parts such as motors, and the malfunction of these drive parts can also cause drive failures.

HDD speed is determined by the rotational speed of the platter. As you know, PCs sold up until only a few years ago used HDDs for storage, so it was normal for users to have to wait a long time for the OS, such as Windows, to boot up after turning the machine on. As such, HDDs can pose a bottleneck in modern systems. More recently, SSDs are often used to speed things up.

SSDs use semiconductor elements called flash memory. As they have no mechanical parts, they are quieter than HDDs, more resistant to impact, generate relatively little heat, and, above all, offer faster read/write speeds. SSDs tend to fail less often than HDDs, but SSDs do have a lifespan dictated by how many write cycles the flash memory can endure.

The difference between consumer and enterprise HDDs and SSDs is that enterprise products offer superior component

precision, durability, and lifespan. Some enterprise products apparently also undergo a pre-aging process before being shipped to weed out initial defects.

### 3.2.3 RAID

While not limited to HDDs and SSDs, potential causes of failure exist in both cases, and there is the possibility of data loss due to failure. To prevent such data losses, storage systems generally use technology called RAID (Redundant Array of Independent Disks) to increase data availability. There are several types of RAID, with RAID 1, RAID 5, and RAID 6 currently being the most commonly used.

RAID 1 mirrors data across at least two drives and offers data protection in the event one of the drives fails. RAID 5 requires at least three drives, with parity information (error correction code) distributed across the drives. Data is protected if one of the drives fails, and data can be corrupted if two drives fail. RAID 6 uses at least four drives, with two parity blocks distributed across all drives. Data is protected in the case of up to two concurrent drive failures, and data can be corrupted if three drives fail.

As an aside, RAID 0 also exists and is designed not for data protection but for increasing performance. RAID 0 is usually not used by itself but in combination with RAID 1, RAID 5, or RAID 6.

Typically, RAID creates a logical unit (LU) or volume out of all or part of the RAID area created in the disk device group and provides that unit or volume to the server as a disk device.

RAID also has a spare drive feature. A spare drive is one that normally remains unused. If a drive failure occurs, data is automatically migrated to the spare, or data is recomputed from the remaining drives and the parity information and restored to the spare drive. RAID spare drive functionality has been in widespread use for a long time, but a drawback of this configuration is that I/O operations are concentrated on the spare drive when a drive fails, resulting in a performance hit. Some recent storage array systems on the market therefore provide a spare for all drives in the system as a means of reducing the performance degradation.

With RAID now explained, let's look at a simple example of each RAID level based on a setup with six 4TB HDDs (no spare drives). Table 1 summarizes capacity and availability for each level.

You can see that RAID 1 is the least capacity efficient, next comes RAID 6, and then RAID 5 is the most capacity efficient. Given the efficiency and availability characteristics, IIJ's storage systems use RAID 5 or RAID 6.

### 3.2.4 Storage Array Systems

Storage array systems consist of a controller that controls the storage functionality along with various types of drives and power supply units. They are designed to have redundancy such that the system can continue to execute read and write requests even if one of the devices fails. Normally, a storage array system must be connected to servers and the like via a network to be usable, and the protocols used differ depending on the way in which data needs to be exchanged. Specifically, there are protocols for file access, protocols for block access, and protocols for object access such as Amazon AWS S3. Here, we focus on file access and block access.

| RAID level | Capacity | Availability |
|---|---|---|
| RAID0 | 24TB | × |
| RAID1 | 12TB | ✓ |
| RAID5 | 20TB | ✓ |
| RAID6 | 16TB | ✓ |

Table 1: Capacity and Availability at Different RAID Levels

### 3.2.5 File Access Protocols

First, let's discuss file servers (NAS) and other types of commonly known storage. Protocols used on NAS include the Network File System (NFS) used on UNIX/Linux operating systems, and the Server Message Block (SMB) and Common Internet File System (CIFS) protocols used on Windows. When configuring file storage on an ordinary server, a volume carved out of a RAID group is formatted, and the area is mounted and used on other servers using NFS or SMB sharing. Applications for file storage include storing web server content, sharing data between applications, and sharing files in an office setting.

### 3.2.6 Block Access Protocols

Next, we look at methods for using HDD- and SSD-like block storage over a network. Protocol types include iSCSI and Fiber Channel (FC) as well as the recently released NVMe over Fabrics (NVMe-oF). A network that mainly connects block storage and servers is called a Storage Area Network (SAN), with storage networks that use FC generally being referred to as FC-SANs and Ethernet storage networks using iSCSI being referred to as IP-SANs.

As a quick introduction, products using FC began appearing on the market in 1993, and iSCSI was published as an RFC by the IETF in 2003, with products using it starting to ship thereafter.

iSCSI is a standard that uses the SCSI storage protocol over TCP/IP, an Ethernet protocol. Dedicated hardware is not required, and it can be configured using the types of NICs installed as standard in Ethernet switches and servers. If you know a thing or two about storage technology, perhaps you are using iSCSI on your home network.

FC is a standard that uses the SCSI storage protocol over FC networks designed exclusively for storage. Dedicated hardware is required, the switches need to be FC switches, and the servers need to have FC-HBAs. Specialized knowledge of these products is also required.

## 3.3 Very Large FC-SANs

IIJ GIO Infrastructure P2 and IIJ GIO Infrastructure P2 Gen.2 use FC-SANs to connect block storage and servers.

As explained in Section 2, FC requires dedicated hardware and specialized knowledge, so the bar to adoption is generally thought of as being high, but having been building and operating FC-SANs in-house since 2003, IIJ has developed an understanding of the characteristics of FC-SANs and IP-SANs and the ability to choose appropriately between both depending on the system environment.

A reason for implementing an FC-SAN, for example, is that FC-SAN switches are equipped as standard with functions that are difficult to recreate on an IP-SAN.

### 3.3.1 Multi-tenancy

When accommodating multiple systems in one network, a common approach to ensuring security between systems with IP-SAN is to use Ethernet VLANs to logically divide the network. This is not particularly difficult to configure when connecting dedicated storage to a single system as shown in Figure 1. When connecting a single storage system to multiple systems as shown in Figure 2, however, the interface on the storage side needs to support multiple VLANs (VLAN Tagging, IEEE 802.1Q).

You might ask whether VLAN Tagging is well supported by storage array systems currently on the market. Many NAS products offer support, but only a handful of products for IP-SANs that use iSCSI and the like provide support, and even then there are restrictions on the number of VLANs, and it is often difficult to accommodate a large number of systems.

Let's turn to FC. Only protocols designed specifically for storage, not Ethernet protocols, are used on FC-SANs, and security between servers is maintained at the FC-SAN level. To enhance FC security, we block access to storage used by other systems using a feature called zoning, which links storage (target) and server (initiator) as shown in Figure 3.

And unlike with Ethernet, sharing a single storage device with multiple systems only requires you to configure zoning for each system, as shown in Figure 4, so there is no need to agonize over array system selection. Another point would be that using an FC-SAN eliminates the need for IP address design between the servers and the storage.

Also, in these examples, there is one switch for both FC-SAN and IP-SAN, but when there are multiple switches between the storage and the servers, VLAN needs to be configured for all of the switches in the case of IP-SAN, whereas FC-SAN provides a mechanism for the zoning information to be shared by all of the switches, so only one needs to be configured. So the configuration workload is lighter in the case of FC-SAN.

### 3.3.2 Lossless Networks

FC provides high performance, low latency, and lossless transmission as standard. If packet losses occur on an Ethernet network, data integrity is maintained using TCP's retransmission control scheme. In environments where storage packet losses occur frequently, storage performance can degrade and applications can experience fatal errors. So when setting up an IP-SAN with Ethernet, network design and other such factors require careful consideration.

Setting up a lossless Ethernet similar to FC can be done by selecting switches that make it possible to use Data Center Bridging (DCB) and other such functionality, but you also need to do additional configuration of the server NICs and storage DCB.

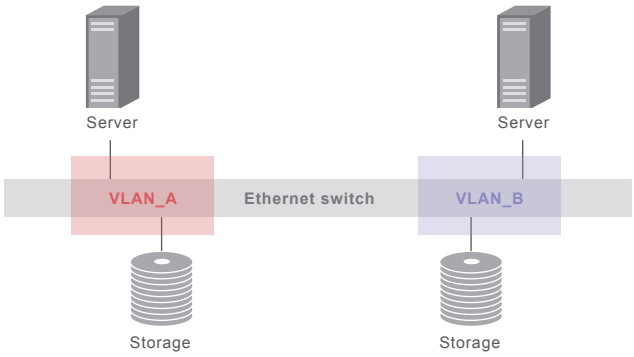


Figure 1: iSCSI Storage Dedicated Configuration

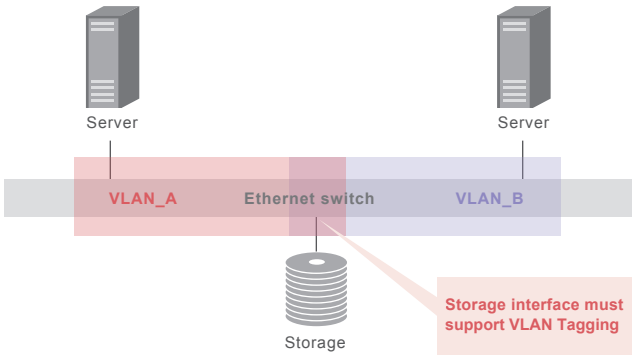


Figure 2: iSCSI Storage Sharing Configuration

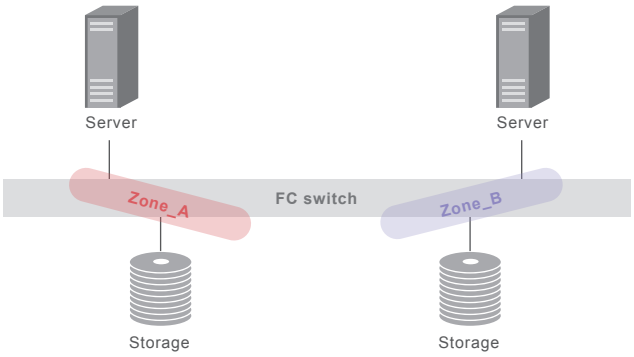


Figure 3: FC Storage Dedicated Configuration

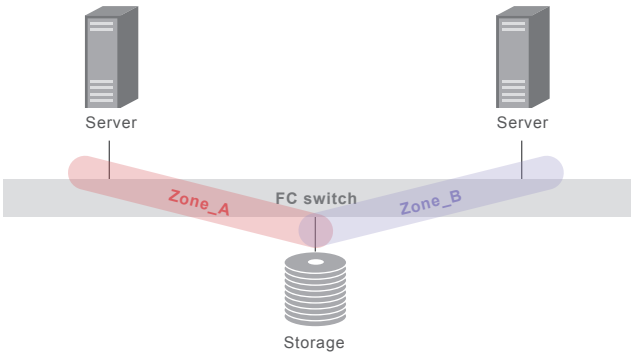


Figure 4: FC Storage Sharing Configuration

### 3.3.3 FC Fabric

With FC Fabric, the following types of fabric topologies can be created. In Figure 5, the servers and storage are connected by a single switch, but in reality you would use two sets of switches to ensure there are two routes between the servers and storage, meaning that a single fault will not cause the system to stop working.

First, let's consider the single-switch topology. IIJ initially used this setup because it allows you to start small. The drawback of this topology has to do with scalability; as more servers are added and storage is increased, you are forced into doing extensive maintenance if you try to scale up.

Figure 6 shows the Core-Edge, Edge-Core-Edge, and Full Mesh topologies.

The Core-Edge topology is effective when storage and servers are densely packed on a single floor of a datacenter. We use it for GIO P2 Gen.2 and some parts of GIO P2. Figure 6 makes it look like the FC switches are connected by a single cable, but they are actually connected by two or more to mitigate the impact if any one of them is disconnected. This uses functionality provided by FC switches called Inter-Switch Link (ISL). This is similar to Ethernet switch port channels, but unlike Ethernet switches, if an FC switch has an ISL Trunk license, the connections will

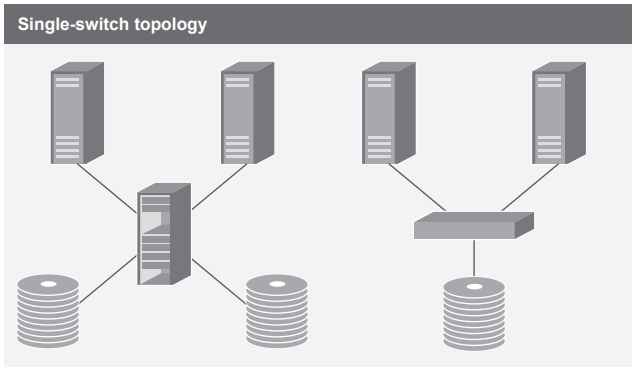


Figure 5: FC Fabric Topology with a Single Switch

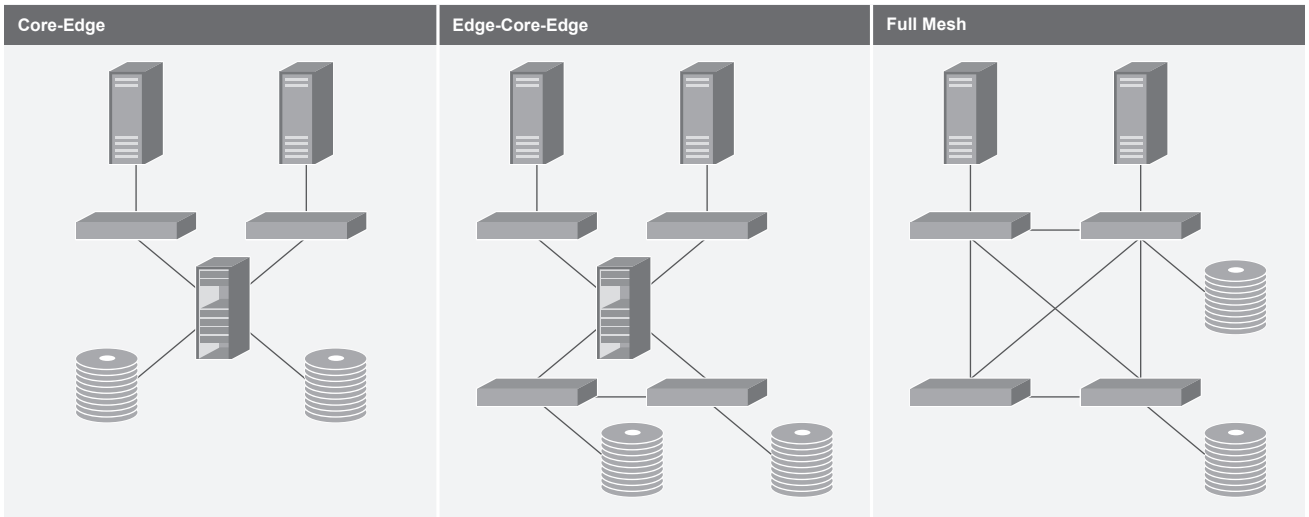


Figure 6: FC Fabric Topology with Multiple Switches

automatically be configured as a single trunk link. For example, if four 32Gbps FC switches are connected with ISL Trunk, the bandwidth between the switches is 32 x 4 = 128Gbps.

Edge-Core-Edge is used when the storage and the servers are installed separately on multiple floors of a datacenter. This can be, for example, when storage cannot be installed near the core, so an edge switch for storage needs to be set up to make the connection with the core. At large scales, an Edge-Core-Core-Edge topology can also be used for cores installed on different floors, and because a lot of traffic can be expected to travel between the cores in this case, we make an effort to design in a large number of ISL Trunks.

We use Full Mesh when datacenter racks contain a mix of storage devices and servers in a small-scale setup. It is worth noting that you often need to configure advanced settings when using Full Mesh with an ordinary Ethernet switch. With FC switches, the ability to automatically configure storage–server routes in an appropriate manner means that you do not need to do any special configuration, so the setup is very easy in this case.

### 3.3.4 FC Fabric Services

FC switches provide a fabric service to manage all devices connected to an FC-SAN and the information needed to connect servers and storage. To give just one example of a fabric service, if zoning between the storage and the servers is set up properly, the servers will automatically be able to connect to the storage without any special settings having to be configured on the server side. In the IP-SAN space, a similar feature called the Internet Storage Name Service (iSNS) exists, but implementations are limited and there are not many system environments in which it is used. The approach with IP-SAN, therefore, is for the servers to keep a record of the storage devices they connect to. Figure 7 summarizes the differences.

If you only have a few servers, for instance, there is not much difference between FC-SAN and IP-SAN in terms of the amount of work required to connect, but with dozens of servers or as many as 100 or so servers, the engineers managing the servers have a lot of work on their hands in the case of an IP-SAN because all of the servers need to have a record of the storage devices they connect to.

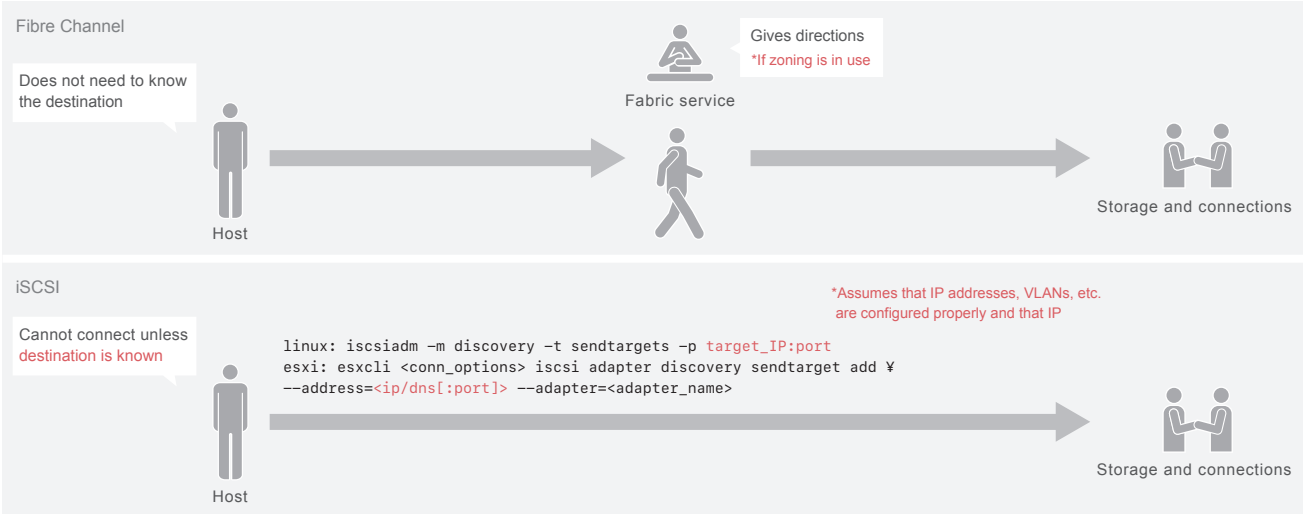


Figure 7: Difference between FC-SAN and IP-SAN Connections

### 3.3.5 Criteria for Selecting FC-SAN

On the operations side, FC switches provide the ability to upgrade firmware versions without stopping FC traffic as well as features to forcibly take ports offline if the error count on a port exceeds a certain level due to SPF+ module faults or the deterioration of cables, thereby preventing any further adverse impact on the FC Fabric.

In view of this, IIJ's decision criteria is as follows. Either FC-SAN or IP-SAN are fine if there is no requirement that a storage array system be used by multiple systems, and FC-SAN is preferable if there is a need for the array to be shared by multiple systems.

That said, this decision criteria could change in future. If a greater range of IP-SAN storage array system products that offer more flexible support for VLAN and the like were to become available, then one would expect the better choice to be IP-SAN over Ethernet, which offers transmission speeds of 100Gbps or more.

## 3.4 Operations Technology

IIJ takes a number of steps to ensure the stable operation of storage systems.

### 3.4.1 Automating Service Delivery

Up until around 2010, customer requests to use storage on IIJ GIO were accommodated by having an engineer whose main role is operating storage systems go in and manually change settings on the storage system and FC switches. Back then, this only had to be done once a week or so at most, which is fairly infrequent, so the engineers were quite capable of getting everything done in time manually.

After 2010, however, once a week turned into several times a week, and even several times a day. Anticipating that it would no longer be feasible to do everything manually, we designed and built systems that fully automate storage system configuration changes.

Automating the configuration of storage is easy with commercially available applications, but commercially available applications commonly offer more than just storage automation functionality (and they are expensive), so we decided to build the storage control part in-house.

Storage control is done in languages such as Python and Ruby. Normally when changing a storage configuration using such a language, it is preferable to use the Storage Management Initiative Specification (SMI-S) or a special API. But the storage products and FC switches we were using at the time did not by themselves support SMI-S or a suitable API. We would have needed separate (expensive) applications to do this, so our standard was to use the command line to manage storage.

The command line is great for when a human wants to provide input and see the output, but the output generally comes in a format that makes it very difficult to get at the appropriate values in the results using a programming language, so the programs we created had to employ fairly cumbersome text processing routines. Moreover, some of the equipment we were using would return a 0 exit status (command executed successfully) even if there were errors in the command or parameters, so we also had to add in our own error handling.

Firmware upgrades on the storage array systems and FC switches present the most trouble. Some products give slightly different command line output before and after a version upgrade, so we needed to run tests in a development environment before performing a version upgrade in the production system.

More recent storage and FC switch offerings are a lot easier to program than the products available back then. With increasing support for configuration management software such as Ansible, device API implementations are moving forward, and storage vendors are now providing modules that can be used to control storage using Python and the like.

### 3.4.2 Storage Infrastructure Monitoring

Storage system monitoring methods include using ping, syslog, SNMP, and the like to monitor the systems, as well as remote monitoring by the manufacturer using the manufacturer's own proprietary monitoring tools. Initially, we used the monitoring service provided by IIJ. Because the number of servers connected to any one storage system was small back then, it did not take long for the storage operations team to be made aware of any faults after they occurred.

As the scale of our operations increased, however, the number of servers connected to any one storage system also increased. As a consequence, IIJ's monitoring service would detect hundreds of alerts for any single fault occurring on those storage systems, and this meant that it was taking longer for the storage operations team to recognize faults.

We therefore decided to build our own monitoring system for the storage operations team that would provide storage system alerts directly to the team. This shortened failure detection times, which were in the tens of minutes, down to the level of a few minutes.

### 3.4.3 Data Migration

HDDs and SSDs installed in storage array systems start to develop faults more frequently after operating continuously for about five years. IIJ looks to replace each piece of storage array system equipment once it has been running for five years, and data must of course be migrated at such times.

In a vSphere environment, you can migrate data with little impact using Storage vMotion and the like, but in other environments, you need to migrate data using some other method. One way is to mount both the source and destination storage on the server or other device and migrate each file individually. The process is shown in Figure 8. In Windows, you can use drag-and-drop or robocopy, for instance, while the options on Linux include cp, tar, and rsync.

This is a very simple way to do it and something we did a lot at IIJ in the past, but in system environments where write operations are executed once only, a single migration run can result in inconsistencies at the level of the entire file repository. So when copying files, our approach is to perform the migration multiple times across several days.

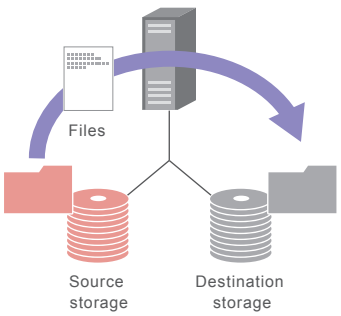


Figure 8: File-by-File Migration

In specific terms, we first do a copy of everything. Next, every business day, we do a copy of anything that has changed. We then single out directories that are frequently written to in parallel and, finally, we stop applications to prevent write operations and then copy only the files that have been written to.

Next, if we are using a logical volume manager on a physical server, we use the logical volume manager's functionality to perform the migration. The process is shown in Figure 9.

You need to have knowledge of the logical volume manager in this case, but data can be migrated with less of an impact than with the file-based migration method described above. And while there is a slight performance degradation for applications running on the storage system, the impact on performance is not huge. Specific examples here include the

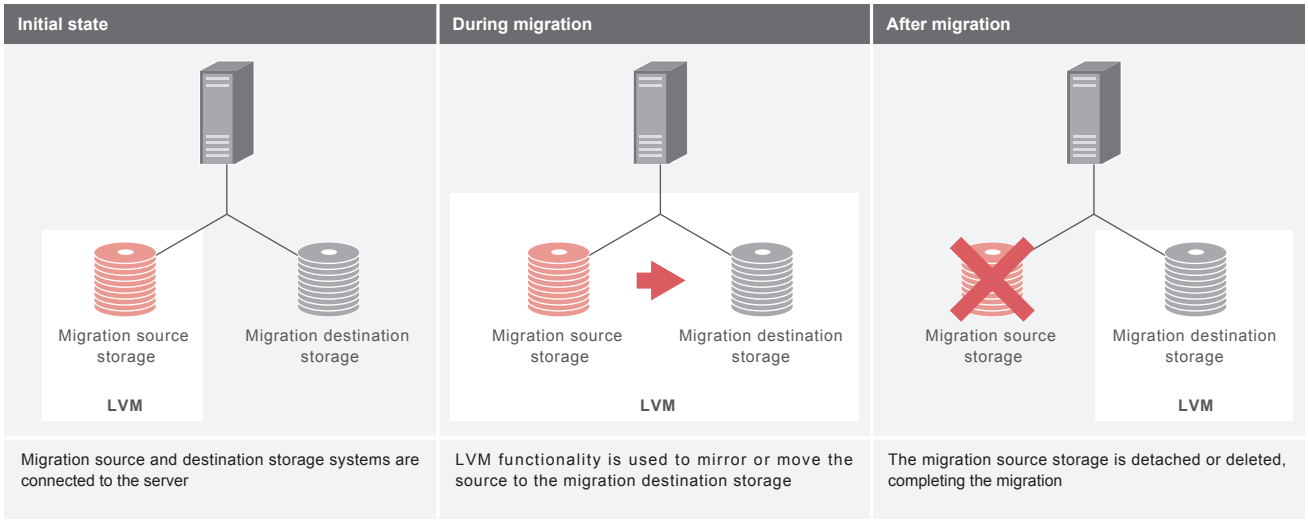


Figure 9: Migration Using Logical Volume Functionality

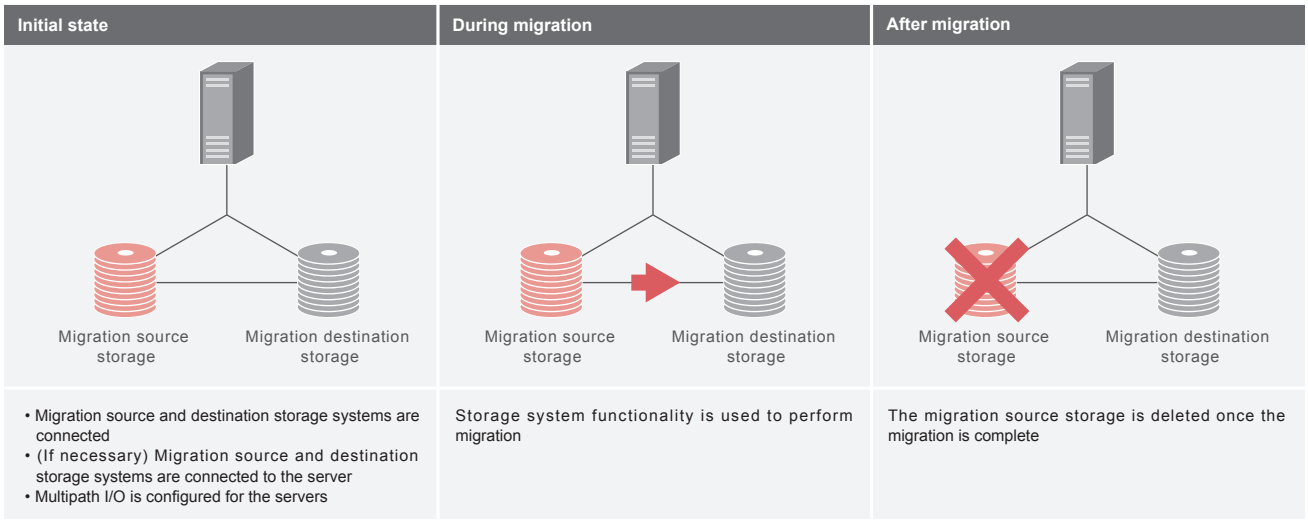


Figure 10: Migration Using Storage System Functionality

use of Logical Volume Manager (LVM) on Linux or HP-UX to temporarily mirror the migration source and destination storage device and, subsequently, disconnect the source once everything has been synchronized, and to limit the discussion to Linux, the use the LVM pvmove command to perform migration.

In environments that do not use a logical volume manager, however, these methods cannot be used, so the only choices are to migrate file by file, as mentioned above, or to use storage migration functionality as described below.

Finally, let's look at the use of storage migration functionality. Figure 10 shows how this works.

Some of the midrange-class storage systems and above these days have functionality allowing data to be migrated between storage systems of the same model or series of products, and some products allow one-way data migration to other vendors' products. Using this method makes it possible to accomplish all of your data migration regardless of the system environment in which the storage is being used. A point to note here is that this is premised on you using the same protocol on both the migration source and destination. That is, you cannot change the protocol used between the server and storage if, for example, you want to use FC on the migration source and iSCSI on the migration destination. Also, depending on the environment in which the storage is being used, you may need to stop the server around the time the migration is performed.

## 3.5 Conclusion

This report has introduced the reader to the way IIJ handles storage by looking at common storage functionality, going into a technical discussion of storage array and storage network adoption, and discussing storage operation technologies. IIJ will continue striving to ensure customers can always feel confident in relying on the storage they use.

**Takahiro Kikuchi**
Senior Engineer, Technology Development, Cloud Division, IIJ
After joining IIJ, Mr. Kikuchi's role involved designing, building, and operating storage infrastructure for IIJ cloud services. He is currently involved in surveys and research activities in the area of storage, the development of engineers both within and external to IIJ, and SINA Japan Forum working groups.

July 2022 Vol.55

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-MKTG020-0053

**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/