4-4 The Design and Application of a Mimetic Network Environment Construction System ～ Security Verification Environment Building System and Contribution to Human Resource Development ～

Shingo YASUDA

Mimetic environments, which mimic actual networks including personal computers, network assets, etc., are required for cyber range or malware analysis. However, constructing various mimetic environments is costly and tedious because each environment has different network assets. Thus, we propose a building block system for constructing mimetic network environments for cyber security experiments. In this paper, we describe the design and applications of Alfons.

1 Introduction

There has been rapid change in the area surrounding information security with the Internet having made itself a part of the social infrastructure. In particular, malware—which through infection causes all sorts of actions to occur that the computer user did not intend—has become diverse and sophisticated, with, for example, the appearance of types such as "targeted attacks" that are used to attack a specific target [1]. Thus, the research and development of countermeasure technologies is a pressing task [2].

In the analysis of malware, in the early stages of infection activity, there are many cases in which the overall activity cannot be ascertained via passive analysis alone, for example in the case where malware introduces further malware via a Command and Control Server (C&C). Further, there are also cases in which previously obtained target's environment information is used in the creation of malware through, for example, social engineering. For this type of malware analysis, it is necessary to carry out active analysis by preparing a mimetic environment that emulates the target's environment in terms of the computer personally used by the target and their office peripherals, etc.

However, domestically in Japan when it comes to human resources engaged in security, the cultivation of security resources has become an issue, with reports that there is a deficit in such resources of 80,000 people, and, what is more, of those engaged in security 160,000 are lacking in skills [2][3].

In security human resource cultivation it is not just classroom learning but hands-on practice that is indispensable. Until now, the Hokuriku StarBED Technology Center has mainly been carrying out research and development on an automatically-constructing tool for network demonstration and verification environment in order to verify network systems [4][5]. The current authors make use of the results from this research to oversee the construction and operation of a practice and competition environment to support a variety of programs and events for security human resource education such as IT Keys [6], SecCap [7], and HardeningProject [8].

Because the configuration of the mimetic environment differs for each event and the individual events are not held with a high frequency, and because there are costs involved in constantly maintaining and operating the mimetic environment, a mimetic environment is constructed and then disassembled at the time of each event on the large scale network emulation platform StarBED. However, because StarBED research and development has mainly been carried out to support environment construction with an emphasis on network system demonstration and verification, with the exception of network setup construction, support has been based on demonstration and verification environments that use the same experiment instance grouping. Consequently, there is insufficient support for a variety of node creation, for example, in application setup and for differences in contents or user history, and the administrator of the environment construction must setup by hand via their own scripts, etc. and the cost in terms of human resources has been an issue in the construction of

environments for security practice and competition events.

There are tools such as Vagrant [9], Ansible [10], and Chef [11] that are designed to save labor in environment construction by automating node setup and OS installation. Through using these, it is also possible to have automatic node generation through duplication of the node image template, but there are also problematic factors, such as needing to write the procedural process sequence. Because mimetic environments for practice degenerate and simulate mainframe networks for actual businesses, etc., there is much redundancy when it comes to the type of OS used as the basis and the network service being provided within the mimetic environment. As a result of this, when it comes to the differences in each instance of actual mimetic environments, there are many cases where the only difference is the existence of files such as settings files, samples, and simulation documents. Consequently, it is more efficient to use a node generation method that more simply focusses on the replacement of files. Thus, the current authors have proposed "Alfons," a system that generates nodes in a building block style and constructs mimetic environments by inserting contents including the execution binary and settings files that form the difference between individual nodes and document files into the OS disk image that is the template. Through this system one can expect to see the simple construction and operation of mimetic environments that are configured with a diversity of nodes.

2 Environment construction and issues

The construction of a mimetic environment can be defined through the three phases shown in Fig. 1 based on node creation for use inside the mimetic environment. Discussed here are the procedures for constructing a regular environment in a test bed and the issues involved in such.

2.1 Template creation

Setup is carried out for shared items, for example, connecting the template OS to the physical server, installing it on the hypervisor as a virtual node, and then installing the necessary applications. When there needs to be multiple OS used in the mimetic environment, multiple templates are prepared. When the OS is the same but different applications are to be installed for each instance, either a different template is prepared for each or all are created with the installed template. However, if multiple applications are installed and the template universality is increased, management of the template image is simplified but the template image hypertrophies and there are cases in which negative effects occur such as when hypervisor disks, which take time to duplicate, are used up. Further, there are cases in which there is a loss of environment feasibility when, for example, multiple server-use applications are installed on client-use instances.

In order to automate the process of this template creation phase, there are OS install automation tools such as virt-install. Further, it is also possible to automate things such as post-OS install setup and application installation and setup with Chef and Ansible. However, the process at the time of template creation happens once for each template for each purpose. Consequently, automation by Chef or Ansible means that, while there is a saving in time through automation, there is a heavy burden of writing to be done by hand. As a result, many of these processes are done by hand. With the cloud services that have recently become popular, by providing users with the various minimum installed OS, simple usage has been realized.

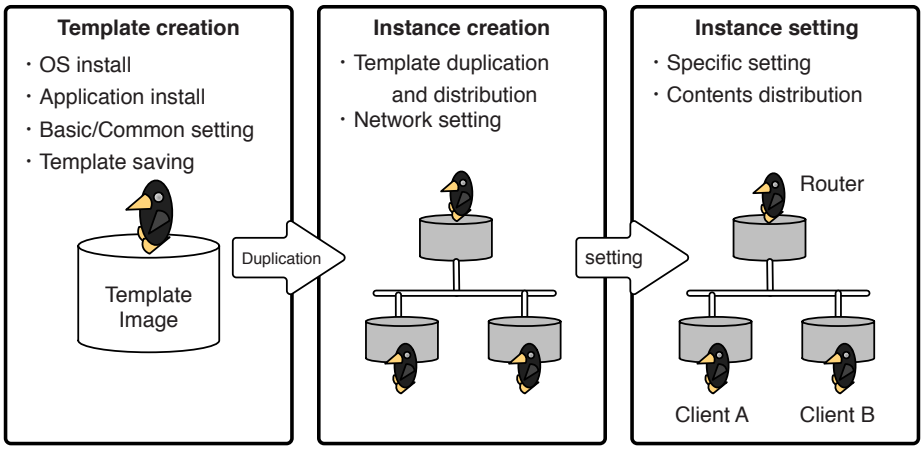

Fig. 1 Mimetic environment construction phase

2.2 Instance creation

In this phase, based on the disk image that is the template, instances are duplicated and distributed as actually used physical instances and virtual instances. At the same time, a mimetic environment network is constructed. With SpringOS and cloud controllers such as VMware vSphere [12] there are functions that support these duplication processes and network setup can also be carried out. However, accompanying duplication there is a change in instance-specific values such as MAC address, IP address, and host name, and because there are cases in which there is discordance with the OS or application settings, there are many cases where the duplicated nodes cannot be used just as they are.

With SpringOS there is no support for changes to network settings within the instance but based on developed as virtual nodes the administrator network interface is regulated for each physical node and, by distributing addresses at fixed intervals through DHCP based on the MAC address of the physical network interface, the OS can then be managed after duplication. However, there is no support for changes to application settings along with cloud controllers or SpringOS.

2.3 Instance setting

Contents such as the instance-specific settings, execute files, and documents that form the difference with the template are distributed on the created instance. Also modified in this phase are the OS or application inconsistencies that are generated from duplication from the template at the time of instance creation. The settings files used by applications in Xenebula and Anybed are distributed in the NFS mount region and, after each instance has loaded by mounting that region, the distribution of the setup files for each instance are aggregated. However, there can be negative effects such having the IO access to the corresponding region bottleneck. By preparing DAEMON and recipes, Chef and Ansible are able to automate the distribution of contents inside each instance. With SpingOS, too, automation is possible by writing setup tasks in in K language. However, with these methods it is necessary to do things such as setting-up a dedicated administration user for the instance and permanently stationing a DAEMON. Therefore, there can be negative effects such as the remaining of unnecessary artifacts depending on the mimetic environment purpose.

2.4 Dual support for physical and virtual nodes

Depending on the type of malware, there are those that will not act in the expected manner with virtual nodes [13] [14]. Here, if the malware creator is careful to not activate the malware in an analysis environment, this means such is done to hinder behavior analysis. In order to circumvent this, it is necessary to support both physical and virtual instances in construction of the environment and create them when necessary. With a regular cloud controller, environment construction can only be done with virtual nodes. Further, because SpringOS only supports physical nodes, if the user carries out hypervisor installing, virtual instances can also be handled, but the environment constructor/operator must administer and control hypervisor setup such as network bridge setup and attachment to instance bridge.

3 Alfons instance creation method

In order to solve the issues discussed in Section **2**, through Alfons, template administration, instance creation, and setup are realized on one system. This section discusses its design.

3.1 File type and introduction phase

Broadly speaking, there are two types of contents that are worked into instances depending on the strength of the interdependent relationship between the contents. The first type consists mainly of files that have a dependent relationship where the OS controls the saving environment and applications, and libraries, for example, rewrite each other's files. In the creation of these types of files, there needs to be standardized OS and application script processing. And, because it is common for those contents to be generated when the usage purpose is decided for things such as servers, clients, routers, and software switches, they are not subject to Alfons automation.

The other type of contents consists of special application settings and application data, etc. These files have a low level of relationship dependency with other OS and applications, and in many cases changes can be completed through file positioning/replacement. The setup and distribution of these contents are carried out during the instance setup phase, but in the construction of mimetic environments up until now, instances were implemented after development and activation on each physical node and hypervisor. However, because processes after development and activation are remotely controlled, there needs to be a

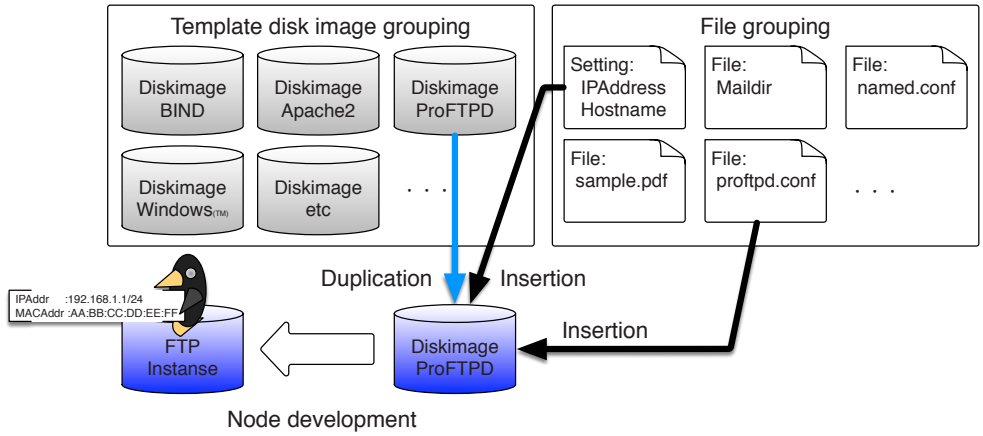

Fig. 2 Creation of building block type instances

script process using administer DAEMON and remote login. In mimetic environments directed at isolated environments for cyber practice and the like, because there can be negative effects such as the remaining of unnecessary artifacts from processes involving permanent stationing of an administration DAEMON and remote login, after creation there is a need for processes that will delete these.

In the case of environment construction tools such as the predecessors of Alfons, which are used in IT Keys and SecCap, in order to stop these negative effects, the environment construction system mounts the instance disk image duplicated from the template and directly edits it so it can insert malware without activating an instance. With this method, because no instance is activated, unnecessary artifacts are unlikely to remain in the instance disk image at the time of malware distribution. This method is employed in Alfons and, as shown in Fig. 2, an instance is created in which setup files and application data are inserted with the same method.

3.2 Disk Image sharing/reuse

In environment construction, because it takes time to create a clean disk image installed by the OS which forms the template, this should preferably be shared/reused as much as possible. Actually, when a cloud controller is used it is common for the immediate post-install instance to be registered as a template. Further, with a regular cloud service, too, it is common for the user to be provided with an OS image that has been minimally installed. With StarBED, too, in iSCSI connection node activation, a standard OS template is provided and nodes can be activated by duplicating this.

In the case of Alfons, it is not only a standard OS image that is provided but also the shared functionality between users for user-created images. With these templates, various types of SNAP can be envisaged for items such as those with only OS installed or for those with specialist applications installed. Wantonly increasing the number of types of template both uses up storage space and lessens the frequency of reuse, but in order to realize a diversity of OS that can be provided as templates at low cost, one can envisage that shared use between users would make it effective.

Further, when an instance is created from a template image on Alfons, there is functionality for creating from the same template both [i] physical instances that were directly installed via a physical server and [ii] virtual instances that are installed as virtual nodes on a hypervisor.

3.3 Resource control

In order to construct a mimetic environment on a server cluster, there is a need for different kinds of setup and control of physical resources such as the control of physical node power supply, the introduction of OS images, and network setup. In SpringOS, an API module grouping is provided in order to control power supply and call up from programs the DAEMON groupings that carry out network setup through VLAN. Because Alfons presupposes the use of StarBED, this API module is used in the control of these physical resources. Even with regular cloud controllers, because the same sort of API is provided it is possible to transplant this into other environments.

When an instance using a virtual node is created, it is not only physical source control that is needed but also the setup/control of hypervisor. In SpringOS, because it is only the API for controlling physical resources that are provided as experimental devices, setup and control of the hypervisor is left to the user. Alfons independently controls

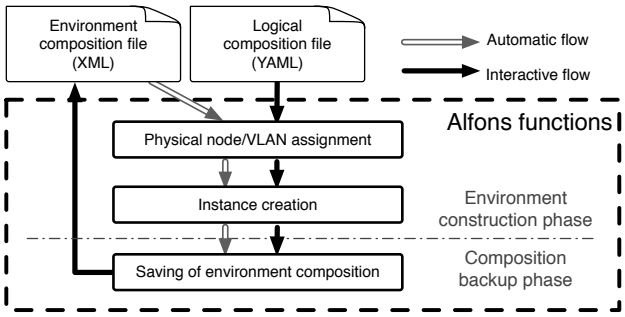

Fig. 3 Environment construction flow

virtual resources, and by coordinating with the control of physical resources provided by SpringOS, control of physical and virtual resources can be carried out in an integrated manner. In the current system, Linux KVM and VMware vSphere Hypervisor act as the hypervisor.

4 Alfons environment construction flow

In Alfons, the mimetic environment is constructed in the two different ways shown in Fig. 3 based on instance creation procedures discussed in Section **3**. Outlined in this section are the procedures for this and the configuration-writing files used in environment construction.

4.1 Environment construction flow

With Alfons, there is support for simultaneous environment construction, based on resource logic configuration files, for both [i] sequential dialogical construction through CLI from allotment of physical resources to the distribution of experiment nodes and [ii] environment configurations written for the environment as a whole. In the dialogical environment construction flow, logical configuration files are created, the physical node ID and VLAN ID to actually be assigned to these are designated, and sequential instances are introduced. Alfons inserts the contents related to the instance ID into the designated template and creates and introduces the designated types of physical and virtual instances into the physical server. At this time, because Alfons automatically introduces a hypervisor into the physical server when necessary, there is no need for the user to oversee the hypervisor. Additionally, in Alfons there is the functionality for inputting environment configurations that were dialogically created/altered as environment configuration files. Based on these environment configuration files, it is possible for Alfons to completely automate environment construction. These environment configuration files are in a file format conforming to XML and also include information that associates physical resources with logical resources. By changing, for example, the physical resource ID associated with the logical resource ID it is easily possible to define environments that are identical or partially different environments.

In malware analysis and practice there are many cases in which, after disassembly, the same environment or an improved version of an earlier one is reused. With Alfons, through the simultaneous construction function based on environment configuration files, experimental environment construction is automated for environment reconstruction/duplication, etc., and it is possible to easily reuse environment configurations.

4.2 Configuration writing format

In order to support construction of mimetic environments, there needs to be a writing format that allows for the system to interpret the network information of users/organizations that are targeted for attack. As discussed in Subsection **4.1**, environment construction in Alfons is carried out using configuration writing files based on two types of format depending on their purpose. The first are logic configuration files in which logical configurations are written in YAML format on the physical resources of the mimetic environment. The second are environment configuration files in which, written in XML format, are the mimetic environment configurations that were actually constructed on the logical resources that were written with the logical configuration files. Along with associating the logical configuration with the assigned physical resources, by way of the setup file, the configuration can be saved/reused by writing the results of the sequential environment construction on said setup file.

5 Actual examples

By using Alfons and the tools that were its predecessor, the Hokuriku StarBED Technology Center has constructed various practice/competition environments on the large scale emulation environment that is "StarBED," and has been supporting the cultivation of human resources. Outlined in this report are three representative examples.

5.1 IT Keys, enPiT-Security (SecCap)

IT Keys is an educational hub-type project for industry-academia-government collaboration that came about as part of MEXT's "FY2007 Progressive Education Program

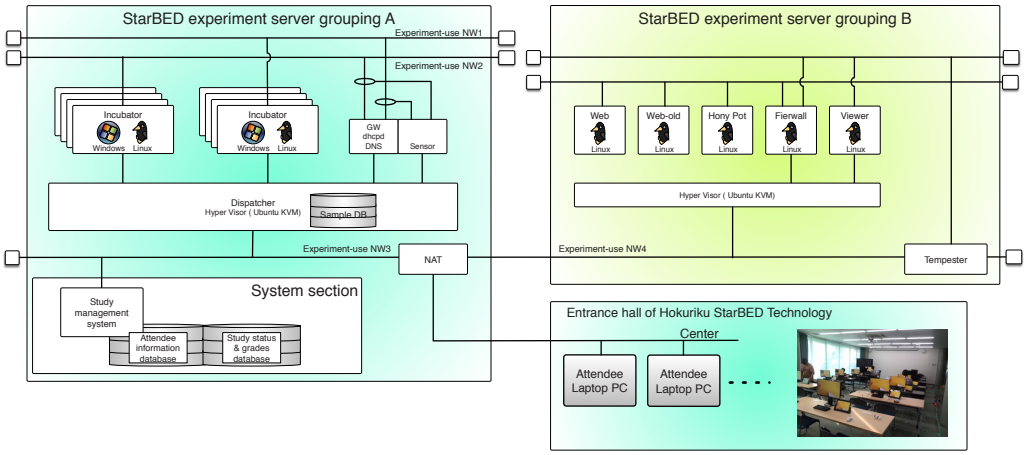

Fig. 4 IT Keys/SecCap practice environment example

for IT Specialist Training," with the goal of cultivating technicians and workers that can provide leadership on the ground in administrating and operating information networks. Continuing from FY2007 to FY2010 and centered in the Kansai area of Japan, the project involved four graduate schools specializing in information (Nara Institute of Science and Technology, Graduate School of Osaka University, Graduate School of Kyoto University, and Japan Advanced Institute of Science and Technology) and four enterprises/groups (NICT, The Research Institute of Information Security, Japan Computer Emergency Response Team Coordination Center, and NTT Communications). Students were gathered together, mainly from graduate schools specializing in information, lectures and workshops were given at the four graduate schools as a course designed to cultivate security specialists, and responsibility for giving the lectures and practice workshops was allotted to the separate organizations involved in the project. Included among these was an intensive three day malware analysis and practice workshop held at the Hokuriku StarBED Technology Center, and every year around 20 attendees came together at the Center. The attendees learned about things like malware types, infection sources, infection routes, and countermeasures while using malware samples actually collected from the Internet in the mimetic environment constructed on StarBED.

enPiT-Security (aka SecCap) is a human resource cultivation program that takes off from IT Keys and which has been running since FY2011. It involves five universities (Institute of Information Security, Nara Institute of Science and Technology, Japan Advanced Institute of Science and Technology, Tohoku University, and Keio University) collaborating under MEXT's "Education Network for Practical Information Technology: Security Division" and runs the same sort of practice workshops as IT Keys in the same intensive way at the Hokuriku StarBED Technology Center.

IT Keys and SeeCap use an environment construction tool that was a predecessor to Alfons to construct a practice environment with a structure like that in Fig. 4 while changing the practice environment according to the malware type and practice type. The attendees split off into teams of two and each uses the laptop computer they are loaned to control Windows or Linux terminals with practice environments created in StarBED. Each team's environment is configured with multiple network instances and the practice is structured such that they work to identify the type of malware infecting the terminal they are using, to find the terminal that is the source of the infection, to come up with countermeasures, and to present their findings in a report.

This practice workshop includes practical activities that are not conducted elsewhere, and students get to use detection engines to collect information, such as hash values, behaviors, and program names, on malware samples that were really collected from the Internet. Because this sort of practice requires a large scale isolated environment it cannot be implemented with regular cloud services or with the resources of individual universities. This is unique example of practice that only the Hokuriku StarBED Technology Center can offer in which a flexible isolated environment can be constructed.

5.2 Hardening project

The "Hardening Project" is an event that has been planned and executed since 2012 by the WASForum Hardening Project committee and NICT gives special support in constructing a competition environment on Hokuriku StarBED. The greatest unique feature of the

Hardening Project is that, whereas regular security competition events focus mostly on testing attack-side skills, this event focuses solely on testing techniques for protection in order to discover and honor the top engineers with the most advanced skills in protection techniques. The competition is shaped to have participants form teams to operate a virtual EC site and, while receiving and resisting a real-time eight hour operator-side attack, compete to have the highest level of sales for their sites. From the 2015 Hardening 10 MarketPlace onward the "market place" concept was introduced, and the competition was made more practical with the adoption of the rule that during the competition each team could purchase professional services such as antivirus software and security diagnosis services out of their earnings. At first, the Hardening Project was held in Tokyo but at present the venue has been changed with an eye to making it an international competition and is held in Okinawa twice a year.

Figure 5 shows the general topology of the Hardening 10 MarketPlace competition environment, and Fig. 6 shows the environment topology for one team. Each team's environment is configured with 22 instances and this is controlled from the Okinawa event venue by connection via a VPN by remote desktop to a competition environment

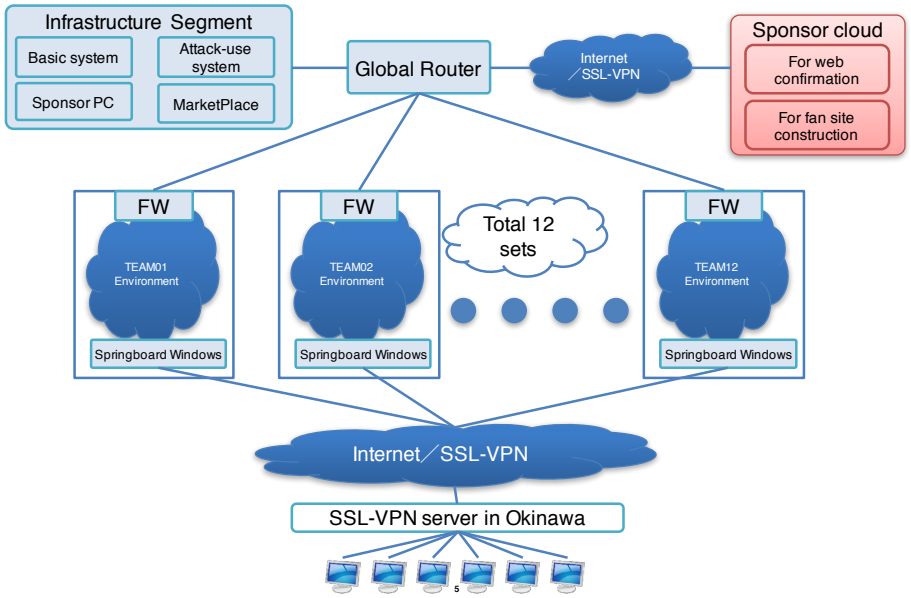

Fig. 5 Overview of Hardening 10 MarketPlace competition environment

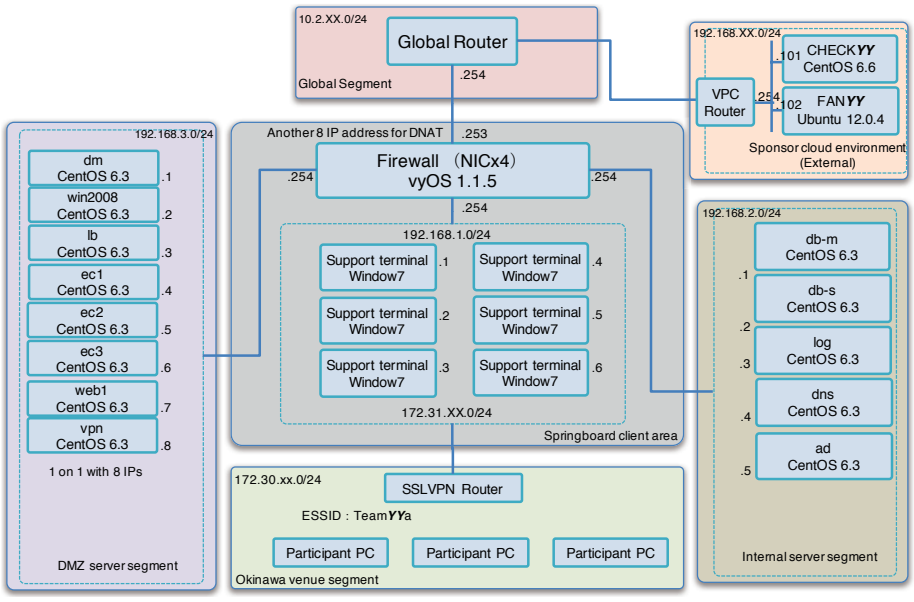

Fig. 6 Hardening10 MarketPlace EC site environment (1 Team)

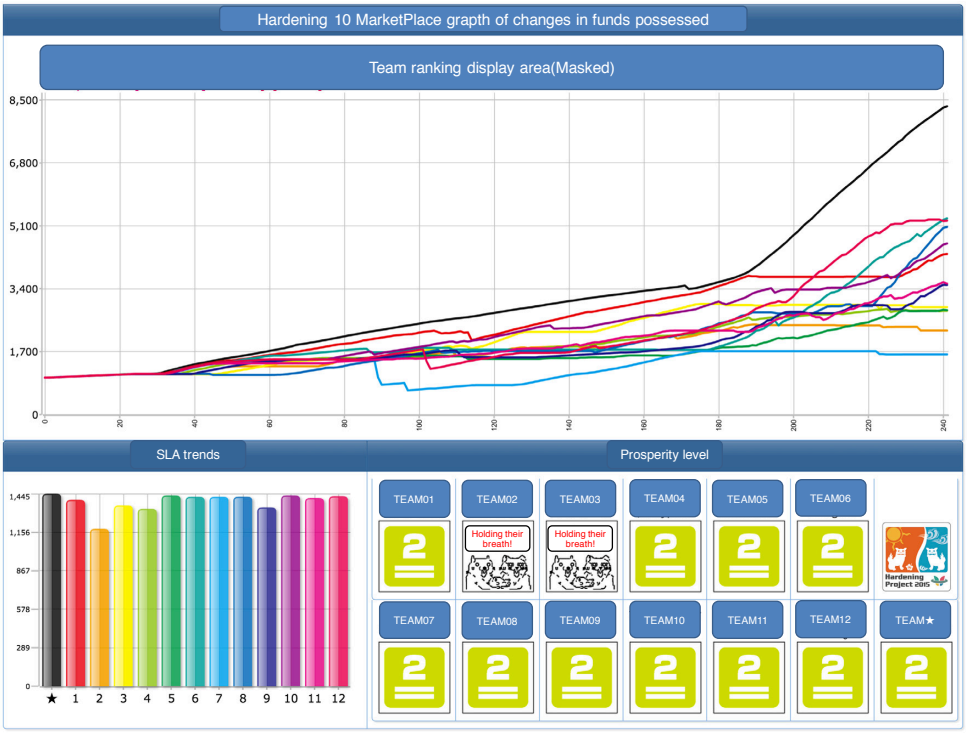

Fig. 7 Hardening10 MarketPlace graph of changes in funds possessed

Table 1 Environment configuration scale (physical node number and instance number)

Event Name	Schedule	Location	Instance type	Number of instances per team	Total number of instances in competition environment	Number of physical nodes used
Zero	2012/04	Tokyo	Physical	5	43	43
One	2012/10	Tokyo	Virtual	10	114	26
One Remix	2013/07	Tokyo	Virtual	17	154	30
10 APAC	2014/06	Okinawa	Virtual	22	154	30
10 Evolutions	2014/11	Okinawa	Virtual	6	88	16
10 MarketPlace	2015/06	Okinawa	Virtual	22	335	57
10 ValueChain	2015/11	Okinawa	Virtual	21	275	49

support terminal (Windows) created on StarBED. Further, at present, one part of an environment that the attack does not directly reach is also created on the cloud environment of a sponsor business, and by connecting to StarBED it becomes a multicloud large scale competition environment with a total of 335 instances.

Figure 7 shows changes in the funds possessed by each team. Each team has 10,000,000 yen worth of virtual funds when they start the competition. Because their funds drop when purchases are made of inventory or security products, the graph falls. If the site goes down due to attack, etc. (purchase crawler program) and customers are unable to purchase merchandise, the graph will flatten. Further, from Hardening 10 MarketPlace onward, the Service Level Agreement (SLA) indicator was introduced. This indicator expresses the prosperity level, and if the EC site is operated in a healthy manner by, for example, responding properly to incidents, the SLA level rises, resulting in an increase in the rapidity of customer purchases and an earlier rise in profits. The operator side plays not only the role of attacker but also acts as the in-house managing president of the participant team's "company" by dealing with emails such as those enquiring regarding site malfunction and those from clients regarding site tampering. These email interactions are also reflected in SLA. For this reason, Hardening Project competitions are not merely a test of technical skill, but also test human factors such as strategy, team management, and decision-making. Further, participants come from a wide variety of working backgrounds, from students to engineers and office staff, and the competitions test comprehensive skills for dealing with security as an organization.

At first, the environment scale of the competition environment construction of the Hardening Project was comparatively small and was constructed using SpringOS, but with the exception of the first event they have all been constructed with virtual nodes and, after hypervisor duplication and introduction on SpringOS environment, construction was carried out by hand. Having expanded the various types of functions in Alfons, which had been in the process of being researched and developed based on this knowledge, it was introduced in the environment construction of the Hardening 10 MarketPlace held in June of 2015. Table 1 lists the resources used at the Harding Project competition event. Every time the competition is held, the environment becomes gradually larger. And, since around the time of Hardening 10 MarketPlace when Alfons was introduced, the competition environment has rapidly grown in size. However, as a result of introducing Alfons, because the construction burden has eased, there has been almost no change for staff working on creating the competition environment, and, particularly for the instance development work, network setting was carried out by a small number of people.

6 Summary

At the Cybersecurity Laboratory, research and development is carried out on the building block-type mimetic environment construction system Alfons. This system not only builds mimetic environments for active analysis of malware but can also be used in the construction of practice environments for the training of security human resources. And, including the tools that were its predecessors, in addition to the three practice environments discussed in this report, work is also being carried out on the construction of various other large and small practice environments.

For governments, businesses, and individuals, security will become even more important in the future. Further, with the popularization of ICT devices in Japan it will become an issue closer to home for people. For that reason, one can see that cyber practice, too, should not stop at just the training of current specialists, and will need to be extended to hands-on practice workshops that everyday people are also able to take—in the same way that in the past people used to normally receive physical security training for crime-prevention and evacuation measures.

Moving into the future, the Cybersecurity Laboratory will continue to work to respond to needs in construction for a variety of practice environments and to support and contribute to the cultivation of human resources and the research and development of environment construction technologies that can realize a variety of practice workshops at low cost.

References

1 Mandiant APT1: Exposing One of China's Cy- ber Espionage Units. http://intelreport. mandiant.com/Mandiant_APT1_Report.pdf, 2013.

2 National center of Incident readiness and Strategy for Cybersecurity (NISC). Cyber Security Strategy. http: //www.nisc.go.jp/active/kihon/pdf/ cyber-security-senryaku-set.pdf, 2013.

3 Information-technology Promotion Agency, Japan (IPA). Basic survey on development of information security personnel. http://www.ipa. go.jp/files/000014184.pdf, 2012.

4 Toshiyuki Miyachi, Junya Nakata, Ken-ichi, Beuran Razvan, Shinsuke Miwa, Takashi Okada, Makoto Misumi, Satoshi Uda, Masashi Yoshizumi, Yasuo Tan, Shin-ichi Nakagawa, Yoichi Shinoda. StarBED Large scale network experiment environment. Journal of Information Processing Society of Japan vol.49, no.1, pp.1–14, 2008.

5 Toshiyuki Miyachi, Takeshi Nakagawa, Ken ichi Chinen, Shinsuke Miwa, and Yoichi Shinoda. StarBED and SpringOS architectures and their per- formance. In *TRIDENTCOM*, vol.90, pp.43–58, 2011.

6 ITKeys. IT specialist program to promote Key Engineers as securitY Specialists. http://it-keys.naist.jp, 2015.

7 SecCap. enPiT-Security. https://www.seccap. jp, 2015.

8 Hardening Project. A comprehensive security competition for website hardening. http://wasforum.jp, 2015.

9 Vagrant. https://www.vagrantup.com, 2015.

10 Ansible. http://www.ansible.com/home, 2015.

11 chef. https://www.chef.io, 2015.

12 VMWare vSphere. http://www.vmware.com/products/vi/, 2015.

13 Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. Detecting environment-sensitive malware. In Proceedings of the 14th International Confer- ence on Recent Advances in Intrusion Detection, RAID'11, pp.338–357, Berlin, Heidelberg, 2011. Springer-Verlag.

14 Detecting Malware and Sandbox Evasion Techniques. https://www.sans.org/reading- room/whitepapers/forensics/ detecting-malware-sandbox-evasion-techniques-36667, 2015.

Shingo YASUDA, Ph.D.

Researcher, Cybersecurity Laboratory,
Cybersecurity Research Institute
Cyber Range Environment Construction,
Network Testbed