Alfons: A Mimetic Network Environment Construction System

Shingo Yasuda(✉), Ryosuke Miura, Satoshi Ohta, Yuuki Takano, and Toshiyuki Miyachi

Hokuriku StarBED Technology Center, National Institute of Information and Communications Technology, 2-12, Nomi, Ishikawa 923-1211, Japan
s-yasuda@nict.go.jp
http://starbed.nict.go.jp/en/

Abstract. 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. These building blocks provides a fine-grained way to manage disk images and files to reduce the construction cost. In this paper, we describe the design and implementation of the building block system called Alfons.

Keywords: Network testbed · Cyber range environment · Testbed construction system · Building block method

1 Introduction

The target of malware attacks is changing from an unspecified number of computers to a particular company or individual. Such attacks are called "targeted attacks." Malware that is utilized in targeted attacks has become increasingly sophisticated over time. Hence, the research and development of countermeasure technology are urgently needed.

The analysis of malware utilizes not only a static analysis but also a dynamic analysis because most malware that is captured at the first intrusion stage is a dropper that obtains the next malware from a command and control server (C&C server); therefore, it is not possible to ascertain its purpose without running the malware. Furthermore, some targeted-attack malware utilizes internal (private) information such as a proxy address or an account ID/PASSWORD. Such malware does not run as expected without these services or that configuration in an analysis environment. Accordingly, the analysis environment should be a mimetic environment with the same services and configuration that are configured for the targeted domain network.

In addition, there is a shortage of security personnel in Japan. In order to foster the knowledge of the security personnel, hands-on practice is effective.

S. Guo et al. (Eds.): TridentCom 2016, LNICST 177, pp. 59–69, 2017.
DOI: 10.1007/978-3-319-49580-4_6

Thus, we have cooperated with some event or security curricula that utilize tools including SecCap [1], the Hardening Project [2], and so on. Such a cyber range needs various network services and mimetic contents in their environments. For example, these services and content include generic and domestic network service applications within a targeted organization and their configurations and business content such as office documents, mail, and so on. We have researched and developed some tools for network testbed orchestration such as SpringOS [3]. These tools can construct a network environment for network software validation. However, an operator must manually install these data when constructing an event environment because SpringOS does not enough functionality for these requirements. This work is costly and tedious.

To address this, we proposed a mimetic environment construction system called "Alfons" for the dynamic analysis and cyber range [4]. Alfons allows for the configuration of applications and has an installation mechanism for content to reduce these costs. Alfons also can configure the applications without trace in the instance; moreover, it also can insert the content files into instance without trace. In this paper, we describe the design and implementation of Alfons. Moreover, we describe a basic performance evaluation based on a conceptual implementation.

2 Related Work

Dynamic Analysis System for Malware: Cuckoo [5] is a dynamic analysis tool for malware. Cuckoo builds a sandbox environment that can run an exe file and utilizes a Kernel-based Virtual Machine (KVM) as virtualization technology.

Christopher Kruegel [6] proposed a dynamic analysis approach that utilizes full virtualization. According to Kruegel, full virtualization such as KVM is more suitable than para-virtualization techniques, because of its affinity for the instruction level and the possibility of observation on Windows.

Miwa proposed a malware analysis platform called MIMI/MAT [7]. This is a full automatic dynamic malware analysis platform that utilizes the network domain, which has some virtual nodes including network services.

These tools usually utilize a clean node that is not used because they do not have the functionality for customizing the internal data of the node.

Virtual Environment Configuration Tools: A cloud controller such as OpenStack [8], VMWare vSphere [9] can create a network service node on demand. SpringOS [3] can manage a physical node on StarBED. Moreover, SpringOS can control the network topology such as the VLAN connectivity between physical nodes, even though other cloud orchestrators cannot. In addition, these orchestrator tools do not have functions that imitate content and configurations.

Vagrant [10], Ansible [11] and Chef [12] are management tools that configure a virtual environment. These tools enable users to configure the virtual node including the configuration of applications, the installation of content, and so on with recipes. A recipe is useful for customizing a virtual node on the basis of the template node image. However, many recipes are necessary when an operator creates many environments or a large environment with various nodes.

Cyber Range Environment Construction: We have cooperated with some events that partially apply these StarBED achievements [1,2]. ENCS,ICS-CERT, and Boeing have other types of cyber ranges, such as Red TEAM - Blue TEAM [13–15]. However, providing many various environments requires a high cost if we utilize these tools or achievements.

3 Definition of Requirements

In this section, we discuss the functionality that the orchestrator tools should have in each step of the network environment construction process. Current network orchestrators have the functions listed in Table 1. Although it appears that the functional coverage area is exhaustive, there is a lack of functionality caused as the premise mismatch of the individual functions. In addition, there is no tool that can transparently control the entire environment. We describe the individual functional requirements in the following.

Table 1. Support coverage of orchestrator tools

	Facility management	Network parameter configuration	Contents installation	Virtual/Physical node support
SpringOS	◯	◯(Agent)	◯(Agent)	Physical
Cloud controller	×	◯	×	Virtual
Vagrant	◯	◯	×	Virtual

Facility Management and Virtual/Physical Node Support: SpringOS has a function related to network topology management that utilizes a bare metal network switch configuration with a VLAN. A cloud controller such as VMWare vSphere and Vagrant can configure the network connection on a hypervisor utilizing a bridge or virtual switch devices instead of not supporting the management of these facilities. It is sufficient if one cyber range or dynamic analysis environment can run on a hypervisor. However, some malware samples have a function for sandbox detection, such as virtualization node detection [16]. There are some techniques for evading sandbox detection [17]; nevertheless, in some cases, we should run such malware on a bare metal machine. Hence, the orchestrator should support virtual and physical nodes; moreover, it should manage integrated with bare metal and virtual switch.

Network Parameter Configuration: All orchestrators can configure network parameters such as the IP address of a node. Moreover, there are some approaches that perform this configuration. SpringOS configures the network parameters by utilizing an agent program. The agent program configures the parameters on an experimental node. Owing to this, all experimental nodes have a management network connection to manage the agents. It is a simple

solution for generic network system software or application validation; however, this is bad for malware analysis and a cyber range. This is because the management connection has the risk of malware traffic leakage outside the environment; moreover, a specific daemon program will interfere with the cyber range and analysis. Vagrant needs a specific account in the construction node to configure the nodes and is also subject to interference. Hence, the orchestrator tool should configure or customize the experimental nodes without a trace.

Installation of Content: Some malware abuses the target internal proxy server to connect to the outside. Other malware explores contents such as office documents or mail. Malware such as this will not run as expected without those network services, configurations, and content in the environment. Hence, to analyze advanced malware, the software should have a function for configuring the network parameters and a function for inserting content. In addition, these functions should perform without a trace.

Environment Separation on a Network Testbed: In a generic network testbed such as StarBED, the bare metal nodes have a management network interface that can be found and used by an OS. In case of generic network software validation, this is useful for managing the environment. On the other hand, this has the risk of traffic leakage outside the environment for a malware analysis or cyber range; moreover, it will be a feature of the analysis environment. Thus, this feature will be the target of sandbox detection. Hence, the orchestrator tool should have environment separation mechanism.

4 Design and Implementation of Alfons

In this section, we describe the design and implementation of "Alfons," which is a mimetic environment construction system for dynamic analysis and cyber range construction. Alfons is assumed to operate on StarBED. Hence, Alfons utilizes the API provided by SpringOS to control the bare metal switches, power control, and OS installation of bare metal servers in StarBED. The other part of the SpringOS API on Alfons is written in Ruby.

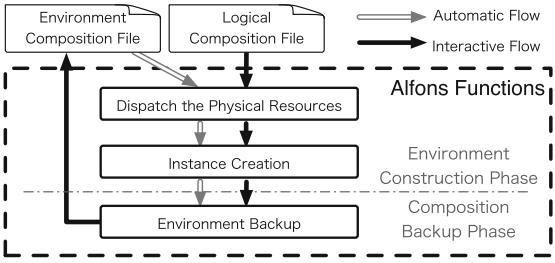

Fig. 1. Construction flow of Alfons

```
nbs:
  nb1:
  nb2:
vlans:
  vl1: {nb1: {eth0: ut}, nb2: {eth0: ut}}
  vl2: {nb2: {eth1: tg}}
```

Fig. 2. Sample logical composition file

4.1 Construction Flow

Alfons has two construction flows based on the CLI shown in Fig. 1. The first is an interactive and sequential construction flow with a logical composition file as seed information. We define the physical server resource as a "Nodebox" in Alfons. Although all networks are defined on a VLAN (802.1q), it is managed by the logical resource names in Alfons because the dynamic analysis and cyber range environments are usually created iteratively. On the other hand, the physical resources that can be used are not always the same in an experimental environment such as StarBED. Hence, in order to obtain environmental portability, it is necessary to separate the logical resource name and the physical resources. A sample of the logical composition is shown in Fig. 2. The logical composition file has adopted the YAML format. This logical composition file defines two Nodeboxes and two VLANs; moreover, two VLANs are connected to each Nodebox as an untagged or tagged VLAN.

First, a user writes and registers a logical composition file to Alfons in this flow. The user then defines the physical resource id, VLAN id to a Nodebox, and the VLANs name by a CLI command in the next step. Next, the user creates an experimental node that utilizes template disk images with content files. In addition, Alfons can save the environment as an environment composition file after the construction of an experimental environment.

The other construction flow is a fully automatic construction flow with an environmental composition file. A sample of the environmental composition file is shown in Fig. 3, which adopts the XML format. This file is created by Alfons as result of an interactive construction or the user who creates it. In addition, this file includes the physical resource and logical resource binding. Hence, the user can easily change a physical resource to construct a copy environment.

4.2 Instance Creation

In Alfons, a physical or virtual node in the environment is called an "instance"; moreover, the template instance dataset is called a "component." Basically, all modern OSs are collection of files on a file system. In other words, an original node is a clean OS disk image with uniquely polluted data files. Hence, we can create the original node image by inserting specified files into the template node image. Alfons adopts this method. A component is preliminarily registered and shared in Alfons by an operator. Alfons creates an instance with its component and the specified files that identify the instance. We call this creation method a "building block system." Figure 4 shows an image of this instance creation

```
<?xml version='1.0' encoding='UTF-8'?>
<NEED ID='Sample-2016-02-22' MiID='SAMPLE01' >
 <node ID='root' nodetype='root'/>
 <node ID='nb1' parentID='root' nodetype='actant'>
  <parameters> <network> <hostname> k001 </hostname>
    <interface name='eth0' bindvlan='731' vlanmode='untagged'/>
   </network> </parameters>
 </node>
 <node ID='ubuntu1404' componentID='ubuntu1404' parentID='nb1' nodetype='emulant'>
  <parameters>
   <hardware name='kvm'/>
   <network type='emulated'> <hostname> ubuntu01 </hostname>
    <interface name='eth0' bindvlan='eth0.731' vlanmode='untagged' ipaddr='192.168.1.1'\
    netmask='255.255.255.0' gateway='192.168.1.254' dns1='192.168.1.241' dns2=''\
    macaddr='AA:BB:CC:DD:EE:FF' media='e1000'/>
   </network>
  </parameters>
 </node>
</NEED>
```

Fig. 3. Sample environmental composition file

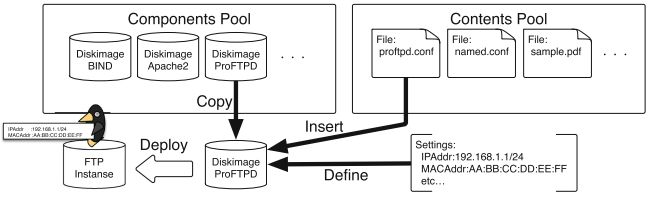

Fig. 4. Instance creation by a building block system

method. A user creates an instance with identifying information such as the hostname, IP address, MAC address, bound VLAN name, and inserted content after determining the component as a base node image. For example, a user can create an FTP service instance with the specified proftpd.conf file if Alfons has the CentOS component that includes the ProFTPD package. To do this, Alfons customizes the instance image file on the Alfons server through a file system mount before transporting the instance image file to a Nodebox; therefore, Alfons can customize without leaving unwanted traces.

4.3 Configuration File Generation

Alfons has the functionality for simple file insertion as well as file string replacement of inserted files. With this function, a user can replace the strings in the configuration files of a component. There are two types of replacement mechanisms. The first mechanism is simple replacement utilizing a format such as __STRING__. For example, this mechanism is utilized for hostname replacement. The user must define the hostname when creating an instance. If a component has configuration files including the replace string "__hostname__," Alfons replaces this when customizing the component image, etc. /etc/hosts,...

The other mechanism is replacement with a specific script. This mechanism utilizes "__(SCRIPT,ARG)__" as a replacement string. "SCRIPT" is a script name that is registered with a component by an operator. The operator can register any language script if it can run on the Alfons server, e.g., a shell script,

Perl, Python, and Ruby. The next column of the script name is the argument, which can be specified as zero or with more arguments delimited by commas. Nesting descriptions are also possible with these formats. For these reasons, it is also possible to define the processing results through multiple stages.

They are very simple instance customization solutions for generating distinctive instance from a component. Ansible or Vagrant and SpringOS can customize the instance images by a specific user account or specific agent program with a specific language such as a recipe. In contrast, these solutions are combination of general regular expressions and scripting language; additionally, they do not need a specific user account or an agent program. This solution should be easily understood by the user.

4.4 Physical and Virtual Node Support

The user can choose either a virtual node instance or a physical node instance based on a component. Thus, the user can construct an optimum environment for analysis or cyber range depending on the type of malware. Alfons automatically installs a hypervisor into a Nodebox if the instance type is a virtual node. In addition, Alfons directly installs an instance into the Nodebox if the operator chooses a physical node as the type of instance. Altogether, the operator does not need to manage the installation of the hypervisor. The current version of Alfons supports the Linux KVM and ESXi as a hypervisor. In addition, Alfons controls the bridge devices in a hypervisor to attach physical network devices to instance tap devices since SpringOS can control only bare metal switches.

5 Evaluation of Conceptual Implementation

In this section, we describe the evaluation of a conceptual implementation utilizing a small cyber-range environment.

5.1 Evaluation Environment

We defined a cyber-range environment that assumes the CTF shown in Fig. 5 for evaluation. This environment has five Nodeboxes in total, including a global router Nodebox and four team Nodeboxes. The team environment has three internal segments: the client, internal-server, and DMZ segments. There are five instances in total for each segment: the firewall, Windows 7 client, file server, database, and DNS/mail instances. Basically, most of the configuration is the same in each team environment, but the IP tables (NAT policy) and global IP address of the firewall instance and the BIND/Postfix/Dovecot configuration of the DNS/mail instance are different. We utilize the six servers listed in Table 2 for this evaluation. These servers connect to a bare metal switch. Only the Alfons server has flash disk storage because the Alfons server copies the component disk image to customize it. The servers for the Nodebox have only HDD drives.

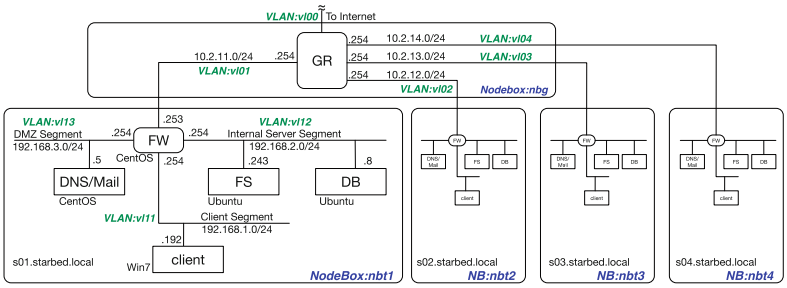

Fig. 5. Network topology for evaluation

Table 2. Hardware specifications of the evaluation environment

Server	CPU	Memory	DISK	Network IF
Alfons server	Intel Xeon®E5-2620 v3 x 2	64 GB	FusionIO SX300 x 2	10 Gbps
Nodebox server	Intel Xeon®X5670 x 2	48 GB	SATA HDD x 2	1 Gbps
Experimental switch	D-Link®DGS-3427	-	-	-

5.2 Performance Evaluation

We constructed the environment ten times for evaluation. In this evaluation, all instances are virtual nodes utilizing the Ubuntu 14.04 Linux KVM as a hypervisor. Table 3 lists the processing times and disk image sizes. Alfons installs the hypervisor only once when creating a new Nodebox. In addition, the current version of Alfons runs all processes sequentially. Alfons took about 1624 s to construct a TEAM environment and about 6754 s to finish the entire construction. Alfons generated the configuration files (BIND files, Postfix files, and so on) on the basis of the template configuration files.

We measured the relationship between the inserted data size and the processing time. In this experiment, we created the firewall instance utilized in the previous experiment with five dummy datas which is different data sizes. Figure 6 shows the results. The results show that the time required for the process depends on the template disk image size and the size of the inserted content.

5.3 System Evaluation

Table 4 summarizes a system evaluation based on a comparison of the conceptual implementation of Alfons and recent tools. Alfons can control the physical facilities and can create a virtual instance on a hypervisor. All of the instances in the environment could communicate with each other; moreover, all of the defined network services run as expected. In addition, all instances, other than

Table 3. Average processing times of the evaluation environment

Section	Disk image size [MB]	Min [s]	Quartile point [s]	Median [s]	Quartile point [s]	Max [s]
HV	10240	546.5	569.0	577.6	587.7	618.2
gr	1334	92.9	94.1	95.4	97.4	103.0
gw	1387	71.4	75.9	80.6	84.5	99.4
dm	3303	122.6	132.9	145.5	154.8	182.6
fs	4989	147.7	162.5	173.1	195.8	222.8
db	4699	133.8	152.9	161.2	181.3	213.8
win7	9987	273.0	306.1	332.5	373.8	479.4
TEAM	-	1572.6	1603.8	1624.2	1639.1	1675.1
TOTAL	-	6507.6	6565.1	6754.3	6789.4	7020.1

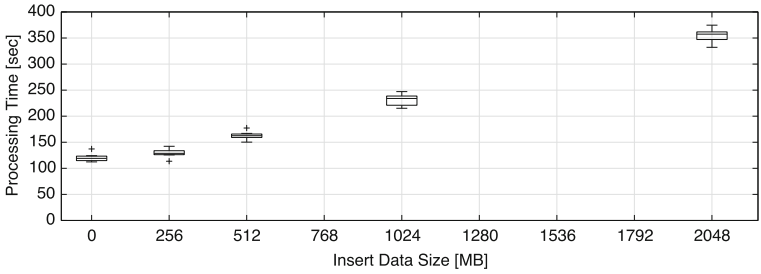

Fig. 6. Processing time for each size of inserted data content

Table 4. Support coverage of Alfons and orchestrator tools

	Facility management	Network parameter configuration	Content installation	Virtual/Physical node support
Alfons	◯	◯(Local)	◯(Local)	Physical/Virtual
SpringOS	◯	◯(Agent)	◯(Agent)	Physical
Cloud controller	×	◯	×	Virtual
Vagrant	◯	◯	×	Virtual

Global Router, do not have any management network interfaces. The user can control the Windows client instance with a VNC on the hypervisor. Thus, this evaluation environment is isolated from other testbed facilities. Alfons can configure the ownership and permission of inserted files without a special account in the instance; therefore, there is no trace of the work in the instances.

6 Conclusion

In this paper, we described the requirements for the dynamic analysis of malware and a cyber range environment construction system; moreover, we proposed the building block system "Alfons" for seamless environment construction. Alfons supports both virtual and physical nodes, and it can manage integrated with bare metal and virtual bridge management. Alfons also can configure the applications without trace in the instance; moreover, it also can insert the content files into instance without trace.

In addition, we evaluated a conceptual implementation of Alfons that utilized a sample environment assumed to be a cyber range. We constructed this environment consisting of 21 instances on five Nodebox servers in about 6754 s. The instances in the environment have no trace of processing such as a specific account and log; moreover, the instances have no management network connection, i.e., its cyber range environment was isolated. These results indicate that our proposed system is effective.

However, since this case is still insufficient, it is necessary to obtain feedback from the findings with future utilization. In addition, methods for increasing the speed of Alfons will be considered, such as simultaneous deployment to multiple Nodeboxes, because the current version of Alfons runs all processes sequentially. We are going to continue research and development for further optimization.

Acknowledgment. The authors thank S. Miwa, Ph.D. from the National Institute of Information and Communications Technology and T. Inoue, Ph.D. from the Japan Advanced Institute of Science and Technology for their insightful comments and suggestions. The authors thank H. Nakai and K. Akashi for their generous support. The authors thank the Hardening Project for giving us the opportunity to practice with the system.

References

1. SecCap: Education network for practical information technologies-security-(only available in japanese) (2015). https://www.seccap.jp
2. Hardening 10 APAC: A security competition like no other (2014). http://wasforum.jp/hardening-project/hardening-10-apac-en/
3. Miyachi, T., Nakagawa, T., Chinen, K.i., Miwa, S., Shinoda, Y.: StarBED and SpringOS architectures and their performance. In: TRIDENTCOM, vol. 90, pp. 43–58 (2011)
4. Yasuda, S., Miura, R., Ota, S., Takano, Y., Miyachi, T.: Building block type construction system for mimetic environment (only available in japanese). In: Proceedings of Internet Conference 2015 JSSST, vol. 77, pp. 69–78, October 2015
5. Cuckoo Sandbox (2015). http://www.cuckoosandbox.org/
6. Kruegel, C., Emulation, F.S.: Achieving successful automated dynamic analysis of evasive malware. In: Black Hat (2014)

7. Miwa, S., Miyachi, T., Eto, M., Yoshizumi, M., Shinoda, Y.: Design and implementation of an isolated sandbox with mimetic internet used to analyze malwares. In: Benzel, T.V., Kesidis, G. (eds.) DETER Community Workshop on Cyber Security Experimentation and Test 2007, Boston, Ma, USA, 6–7 August 2007. USENIX Association (2007)
8. OpenStack (2015). https://www.openstack.org/
9. VMWare vSphere (2015). http://www.vmware.com/products/vi/
10. Vagrant (2015). https://www.vagrantup.com
11. Ansible (2015). http://www.ansible.com/home
12. chef (2015). https://www.chef.io
13. ENCS: European network for cyber security (2015). https://www.encs.eu
14. ICS-CERT: The industrial control systems cyber emergency response team (2015). https://ics-cert.us-cert.gov
15. CRIAB. http://www.boeing.com/defense/cybersecurity-information-management/
16. Lindorfer, M., Kolbitsch, C., Milani Comparetti, P.: Detecting Environment-Sensitive Malware. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 338–357. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23644-0_18
17. Detecting Malware and Sandbox Evasion Techniques (2015). https://www.sans.org/reading-room/whitepapers/forensics/detecting-malware-sandbox-evasion-techniques-36667