A real-time correlation of host-level events in cyber range service for smart campus

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 14, 2018, accepted June 2, 2018, date of publication June 12, 2018, date of current version July 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2846590

A Real-Time Correlation of Host-Level Events in Cyber Range Service for Smart Campus

ZHIHONG TIAN[1], YU CUI[2], LUN AN [3], SHEN SU[1], XIAOXIA YIN[4], LIHUA YIN[1], AND XIANG CUI[1]

[1]Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China
[2]School of Cyberspace Security, Hunan Heetian Information Technology Co., Ltd., Changsha 410000, China
[3]Beijing University of Posts and Telecommunications, Beijing 100876, China
[4]College of Engineering and Science, Victoria University, Melbourne, VIC 3011, Australia

Corresponding authors: Lihua Yin (yinlh@gzhu.edu.cn) and Xiang Cui (cuixiang@gzhu.edu.cn)

This work is supported by the National Natural Science Foundation of China under Grant 61572153. and the National Key Research and Development Plan (Grant 2018YFB0803504).

ABSTRACT Smart campus is an exciting, new, and emerging research area that uses technology and infrastructure to support and improve its processes in campus services, teaching, learning, and research, especially, the explosive growth in knowledge makes the role of cybersecurity of smart campus become increasingly important. Cyber range is an adaptable virtualization platform consisting of computers, networks, and systems on which various real-world cyber threat scenarios and systems can be evaluated to provide a comprehensive, unbiased assessment of the security of information and automated control systems. As an important part of features, cyber range must provide the capability of data collection, aggregation, correlation, and replay for the scenario owner or any ''specialized users'' to review attacks–defense processes on known targets and future zero-day research. To this end, based on our previous work, the Heetian cyber range, we proposed a method named C2RS meaning ''a real-time correlation of host-level events in cyber range service.'' C2RS implements out-of-band data capturing for greater attack resistance with virtual machine introspection technique. This approach allows C2RS to isolate the data captured from monitored hosts. C2RS leverages these captured data by incorporating them into the volatility framework to aid in simplifying the analysis of operating system memory structures. Finally, we proposed an object-dependent method to analyze the evidence of illegal activity. We conduct extensive experiments to evaluate the functions and performance of C2RS in a dynamic service. Through the test, we confirm that the proposed method is effective for real-time correlation of host-level events in cyber range service.

INDEX TERMS Security education, cyber range, network security, correlation, smart campus.

I. INTRODUCTION

The Smart Campus will implement appropriate technology platforms to enhance campus services and facilitate teaching, learning, and research. Cyber Range can provide advanced cybersecurity training exercises for college students, revolutionize cybersecurity education. Just as every military and police force needs a firing range to hone weapons skills and battle tactics, many new technologies are being developed and introduced for Internet, home networks and sensor networks must be evaluated in a testbed before deployment. A cyber range is generally defined as a virtual environment that is used for cyberwarfare training and cyber technology development. It provides tools to help strengthen the stability, security and performance of cyberinfrastructures and IT systems to be used by government and military agencies [1]. Cyber ranges function like shooting or kinetic ranges, facilitating training in weapons, operations or tactics. Thus, cyber warriors and IT professionals employed by various agencies train, develop and test cyber range technologies to ensure consistent operations and readiness for real world deployment.

Generic cyber range utilizes platform virtualization to provide basic functionality for deploying virtual appliances, configuring flexible network topologies and the emulation of various network characteristics. Cyber range uses cloud controllers such as OpenStack and VMWare vSphere as network management configuration tools [2]. These tools are able to create and manage physical nodes and network experiments

with multiple virtual machines and network service among them. They have the functionality to control network topology among nodes. When cyber range is used in cybersecurity training, a collection of hosts are connected to each other via LAN network that are initialized. Based on the global information, the corresponding base images are started up to be ready. Next, it installs security content, creates a number of virtual machines and the network service. And then ends the cyber range creation process.

Regular cyber security awareness training is important, however, accurate network attack and defense assessment, as a big problem, must first be addressed [3]–[5]. With the training progresses, cyber range needs monitoring both virtual machines and virtual networks in real-time. Through the traffic capture, analysis and display, the trainees can clearly understand the details of training process and the cyber situational awareness, which can provide support for subsequent effect evaluations. The current typical approach utilizes host-based forensics techniques, which consists of some agents that embed a virtual machine. But this method has the risk of being attacked, and the ability to resist destruction is poor.

In this paper, based on our previous work of the Heetian cyber range [6], we present a real-time Correlation of host-level events in Cyber Range Service (C2RS) method. C2RS implements an out-of-band data capturing mechanism for greater attack resistance utilizing virtual machine introspection technology. This approach allows C2RS to isolate the data captured from monitored hosts. C2RS leverages these captured data by incorporating them into the Volatility framework to aid in simplifying the analysis of operating system memory structures. To aid the trainees aware the cybersecurity situational in cyber range, we proposed an object dependent method to achieve the evidence of illegal activity. The proposed C2RS method is scalable and robustness, which makes our method more applicable to other tasks.

The remainder of this paper is organized as follows: In section 2, we present previous work on event correlation in virtual machine. In Section 3, we present the proposed C2RS method and the correlation algorithms. The resultant experiments conducted to evaluate the method are discussed in Section 4. Section 5 concludes the paper, discusses and summarizes future research directions.

II. RELATED WORK

Many secure machine learning schemes have also been proposed for android malware detection and cloud data [7]–[11]. In our paper, we focus exclusively on the related works used forensics techniques for monitoring virtual machines in cyber range.

Most mature forensic investigation tools like EnCase [12] and Safeback [13] focus on capture and analysis of evidence from storage media on a single host. Mnemosyne is a dynamically configurable advanced packet capture application that supports multistream capturing, sliding-window based logging to conserve space, and query support on collected packets [14]. Evidence graph network forensic analysis mechanism [15] includes effective evidence presentation, manipulation and automated reasoning, although it is nice to present evidence correlation in graphic mode, this system is still a prototype and lacks the effective capability of inference. Tian *et al*. [16], [17] developed a network intrusion forensics system based on transductive scheme and Dempster-Shafer Theory that can detect and analyze efficiently computer crime in networked environments, and extract digital evidence automatically. ForNet [18] is a novel distributed logging mechanism that focuses on network forensic evidence collection rather than evidence analysis.

Above forensics systems are based on audit trails. Systems relying on audit trails try to detect known attack patterns, deviations from normal behavior, or security policy violations. One of the main problems with these systems is the unacceptably high overhead. To analyze logs, the system must keep information regarding all the actions performed, which invariably results in huge amounts of data, requiring disk space and CPU resources. Next, the logs must be processed to convert them into a manageable format, and then compared with the set of recognized misuse and attack patterns to identify possible security violations. Further, the stored patterns need to be continually updated, which would normally involve human expertise. An intelligent, adaptable and cost-effective tool that is capable of this is the goal of the researchers in network forensics.

Different from the above research, Walls *et al*. [19] developed DEC0DE, a system for recovering information from phones with unknown storage formats, a critical problem for forensic triage. Because phones have myriad custom hardware and software, they examine only the stored data. Via flexible descriptions of typical data structures, and using a classic dynamic programming algorithm, they are able to identify call logs and address book entries in phones across varied models and manufacturers. Li *et al.* [20] perform extensive study of existing fuzzy hashing algorithms with the goal of understanding their applicability in clustering similar malware. They developed a memory triage tool that uses fuzzy hashing to intuitively identify malware by detecting common pieces of malicious code found within a process. In [21], a host-based intrusion detection system offers a high degree of visibility as it is integrated into the host it is monitoring. A new approach, based on the k-Nearest Neighbor (kNN) classifier, is used to classify program behavior as normal or intrusive. Ko *et al*. [22] introduced an approach that integrates intrusion detection techniques with software wrapping technology to enhance a system's ability to defend against intrusions. In particular, they employ the NAI Labs Generic Software Wrapper Toolkit to implement all or part of an intrusion detection system as ID wrappers.

Because the above host-based methods operate at user level, unfortunately, these systems are quite susceptible to attack once an attacker has gained privileged access to a host. Besides, an operating system crash will generally cause the system to fail open. Since the host-based method runs in the same fault domain as the rest of the kernel, this will

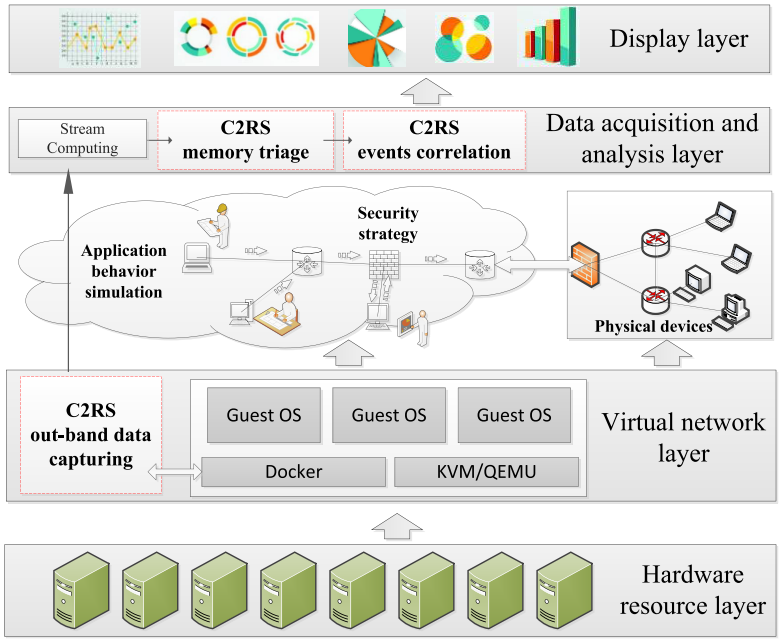


FIGURE 1. The architecture of HTCR.

often cause the entire system to crash or allow the attacker to compromise the kernel [23]. The C2RS proposed in this paper has the advantages that the above frameworks do not have: 1. an expandable, modular system architecture; 2. out-of-band data capturing for greater attack resistance with virtual machine introspection technique; 3. an object dependent method to analyze the evidence of illegal activity and reconstruct intrusion scenarios.

III. THE PROPOSED C2RS METHOD

In our previous work, we designed and developed a large-scale, realistic and real-time network testbed, named HTCR, whose architecture is shown in the figure 1. The hardware resource layer of HTCR contains hundreds of PCs, and switched networks. The virtual network layer uses platform virtualization technology to effectively emulate a realistic network for experimentation. The base images are under the raw format that is used for KVM virtualization. They contain a pre-installed operating system and several basic system configurations (e.g. host name, IP address, etc.). HTCR is dual support for physical and virtual nodes. HTCR can accurately simulate network traffic and attack scenarios, to provide an immersive experience that prepares trainees for a real-life attack and accelerates new hire onboarding. In addition, HTCR includes a rich catalog of training packages and scenarios, which can offer to the trainees.

To expand HTCR so as to be suitable for assessment of the security situation in the cyber range, we are now implementing C2RS. An out-band data capturing module is added into virtual network layer. The data acquisition and analysis layer are expended with a Memory triage module and an event correlation engine. In the following sections, the above modules will be described in detail.

A. VIRTUAL MACHINE INTROSPECTION BASED OUT-BAND DATA CAPTURING

The guest's runtime information that in-VM components consume to carry out their monitoring functionality either resides in the VM's memory or disk. For out-of-band data capturing, this runtime state needs to be obtained from outside the guest's context. This virtual machine introspection (VMI) [24] process can be broken down into two steps: (i) exposing VM runtime state, i.e., getting an out-of-band handle on the VM's memory and disk, and (ii) exploiting VM runtime state, i.e., interpreting the ex-posed memory and disk images to reconstruct the guest's runtime information.

LibVMI is a C library with Python bindings that makes it easy to monitor the low-level details of a running virtual machine by viewing its memory, trapping on hardware events, and accessing the vCPU registers.

In our proposal, LibVMI is used as bridge connecting Qemu-KVM and Volatility [25]. As volatility is related to specific operating system and all operating system's logic are running in Volatility. Volatility will read CR3 register and request memory blocks starting from specific physical address most of the time. These requests are converted from Python to C by PyVMI. Then, PyVMI will call LibVMI Kernel to connect to QEMU socket created by Qemu-KVM Patch and return data to Volatility.

To achieve this purpose, Qemu Patch and PyVMI are integrated into the source. Qemu Patch changes the method of physical memory accessing through UNIX socket which will

reduce almost 90% time in accessing Qemu-KVM and need not to pause Guest operating system. The latest code of the Patch is for Qemu Ver 2.8 and we modified part of the codes to adapt Qemu Ver 2.9. After patched, Qemu-KVM should be reconfigured, make and make install on the physical host.

PyVMI is adapted when programmers want to code in Python instead of C and is already packaged in 'libvmi/tools/pyvmi' directory. As upper-lever of the framework is volatility written in Python, PyVMI is definitely required. The older version of PyVMI cannot process well enough which has many errors that cannot compiled (the latest version has fixed these errors and supported Python 3). We modified related codes of PyVMI by merely picking up volatility-related functions, the most significate functions are shown in table 1.

TABLE 1. Most Significate Functions for connections to volatility.

Function Name	Descriptions
vmi_ins_init	Constructor Point operating system type and sysmap path
vmi_get_cr3	Get data of register CR3
vmi_get_memsize	Get Allocated Memory size for a VM
vmi_read_pa	Read dedicated length of data in memory through physical address
vmi_zread_pa	Read dedicated length of data in memory through physical address with zeros in memory holes instead of failing

B. MEMORY TRIAGE USING VOLATILITY

C2RS uses volatility which is designed to work on triage memory snapshot, to extract operating system memory structure. Through a triage analyzing on a physical memory image dumped from a VM, volatility could extract useful information from that image. This information contains process list, library list, process maps, open files, netstats, open ports, etc. At first, volatility only works on the memory image file to extract information and could not link to LibVMI to support real-time triage analyzing. Later, researchers developed an Address Space Plugin linking to PyVMI making volatility facilitating analysis on a running virtual machine.

But for a long running service that periodically facilitates analysis on multiple running virtual machines, this is not enough and it has several significate obstacles to be overstepped. First, volatility is a single time process and acts with parameters like 'python vol.py -f XXX.img –profile= OS-Type Commands'. This will invoke the whole process of initializing, logical processing and formatting outputs.

In the LibVMI mode, the time of initializing usually costs more than 10s and consumes most of the analyzing time compares to 100ms-level of commands like pslist. This is unacceptable when performing periodically facilitates analysis within second-level intervals. What we need is one time initializing and periodically processing configured commands. Second, a volatility instance can only perform one command, if multiple commands are needed one by one, the current version fails. Third, we want to monitoring multiple VMs at the same time, so that, we need a wrapper or a schedule service to receive configurations and dynamically create or delete a task which processing on a specific VM.

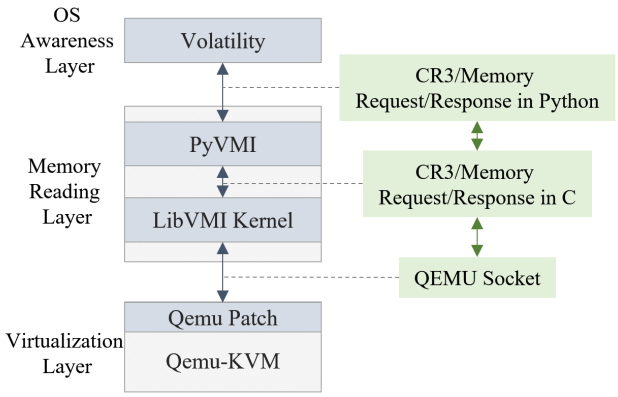


FIGURE 2. Workflow of the revised LibVMI.

To achieve these two requirements, we modified part of the source codes:

1. A schedule service is added to receive triage configurations, manage running tasks of VMs (create or delete), etc.

2. The entry logic of volatility (vol.py) is modified to a function entry with parameters to support calling from schedule service. For the purpose of running multiple commands sequent in one task, 'vol.py' is modified to have the ability of changing running parameters between commands.

3. A new address space plugin is added in to the framework which links to PyVMI to access physical memories.

4. The logic of address space selection for a type of operating system is modified, two global variables ('global_base_as' and 'libvmi_base_addr') are added. 'global_base_as' is used to record the top AS on the condition of AS re-selection and 'libvmi_base_addr' is used to record the bottom AS (PyVMI AS) for a specific VM. 'libvmi_base_addr' is constructed only once in the process of 'Initialize AS' and will not change any more, while 'global_base_as' may change between different commands in a running task. The modified volatility framework is shown in figure 3.

C. HOST-LEVEL EVENTS CORRELATION ENGINE BASED ON OBJECT DEPENDENCE

The output results of volatility are stored in the backend database in the form of evidence vector *EV*. Based on the evidence vector, the events correlation engine conducts evidence analysis and associate operations, reconstructs the invasion process, and submits the correlation results in the form of evidence graph. To this end, in the following, we first propose a few corresponding definitions.

Definition 1 (Object Dependency): We define object dependency as a binary relationship between 2 objects, i.e. a 3-tuple $OD = < Source, Sink, Operation >$, denoted as $Source \overset{Operation}{\Rightarrow} Sink$, which means *Source* depends on *Sink*.

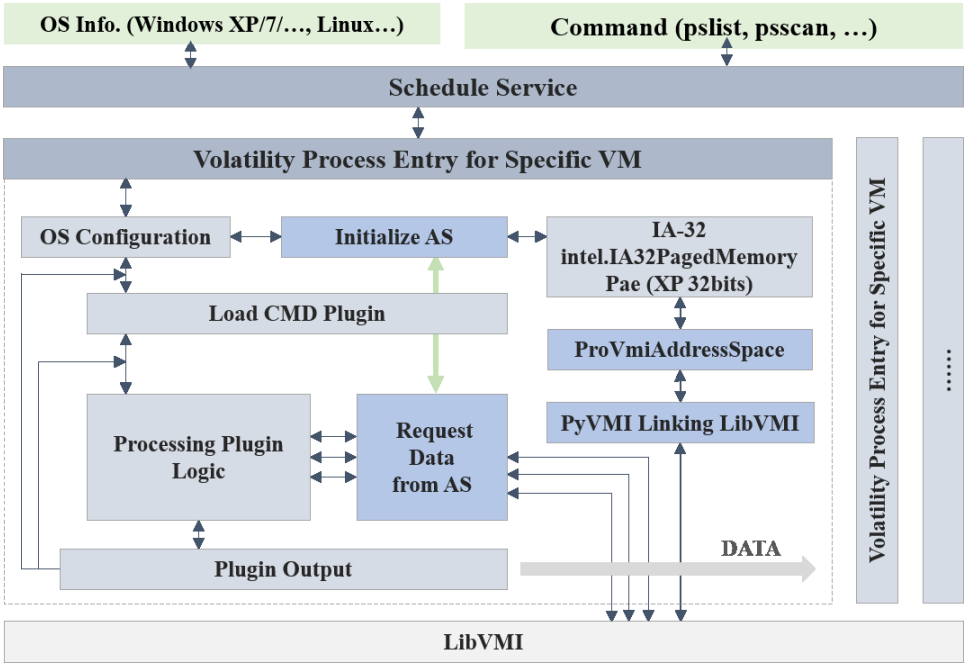


FIGURE 3. The modified volatility framework.

Here, *Source* and *Sink* are both objects; *Source* refers to the principle object; *Sink* refers to the target object; and Operation refers to the operation which the principle object acts on the target object.

Based on the definition of the object dependency, C2RS describes every evidence vector *EV* in forms of object dependency. For example, the evidence "a parent process creates a child process" could be denoted as $parent \overset{fork}{\Rightarrow} child$. Here the principle object (*Source*) is the *parent* process, and the target object (*Sink*) is the *child* process, and operation is the system call *fork*. If a following evidence "the child process writes a file" exists, the following evidence should be denoted as $child \overset{write}{\Rightarrow} file$, the principle object is the *child* process; the target object is the *file*; and the operation is the system call *write*.

Based on the classification of the system calls involved in OD's operation, we divide C2RS's OD into 4 types, including "*process*/*process*" OD, "*process*/*file*" OD, "*process*/ *network*" OD, and "*process*/*signal*" OD. As a result, *Source* generally refers to a process, and *Sink* could be process, signal, file, or network. Operation refers to a system call. As shown in Figure 4 is the dependency relationships among different objects of C2RS.

Definition 2 (Evidence Graph): Evidence graph is a directed graph, and defined as a 2-tuple $EG = <N, E>$. Here N refers to the node set, a node could be any kind of objects in *OD*, and E refers to the edge set. If $\exists n_1, n_2 \in N$, and $n_1 \overset{Operation}{\Rightarrow} n_2$, then there exists a directed edge e from n_1 to n_2.

The evidence graph provides the most visual representation of the object dependencies, thus the correlation could be

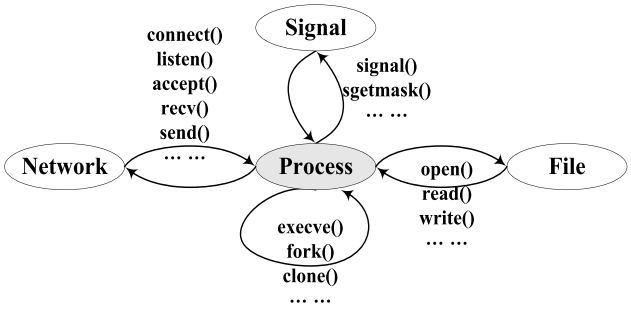


FIGURE 4. Object Dependency relationships transformation.

recognized as the generation of an evidence graph as comprehensively as possible. In order to accelerate tche procedure of key object nodes and key steps searching in the intrusion process for the forensics analyst, we define the following 3 graph operations.

Definition 3 (Precedent Graph Operation): Given anevidence graph $EG = <N, E>$, and an object node $n \in N$, the precedent graph operation *precedent*(n, EG) returns EG's maximum subgraph $PG=< N', E' >$, which satisfies (1) $n \in N'$, (2) for $\forall n' \in N'$ and $n' \neq n$, there exists a directed path from n' to n, (3) for $\forall e \in E'$, e exists on a path directed to n. We refer PG as node n's precedent graph in EG.

Definition 4 (Subsequent Graph Operation): Given an evidence graph $EG = <N, E>$ and an object node $n \in N$, subsequent graph operation *subsequent*(n, EG) returns EG's maximum subgraph $SG = <N', E'>$, which satisfies (1) $n \in N'$, (2) for $\forall n' \in N'$ and $n' \neq n$, there exists a directed

path from n to n', (3) for $\forall e \in E'$, e exists on a path directed from n. We refer SG as node n's subsequent graph in EG.

Definition 5 (Key Graph Operation): Given an evidence graph $EG = <N, E>$ and an object node $n \in N$, key graph operation $key(n, EG)$ returns EG's maximum subgraph $KG = <N', E'>$, which satisfies (1) $n \in N'$, (2) for $\forall n' \in N'$ and $n' \neq n$, there exists a directed path from n to n' or form n' to n, (3) for $\forall e \in E'$, e exists on a path directed from n or directed to n. We refer KG as node n's key graph in EG.

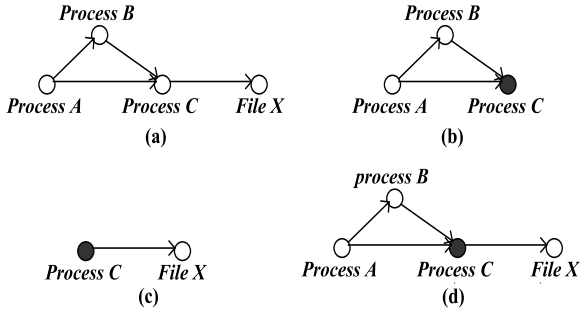


FIGURE 5. Schematic of the 3 graph operations. (a) Evidence graph. (b) PG = precedent(process C, EG). (c) SG = subsequent(process C, EG). (d) KG = key(process C, EG).

As shown in Figure 5 is the schematic of the above 3 graph operations. The key object node is *process* C (marked with black circle). Obviously, the key graph of an evidence graph is equivalent to the union of its precedent graph and subsequent graph, i.e. $key(n, EG) = precedent(n, EG) \cup subsequent(n, EG)$.

Based on the object dependencies, the events correlation engine is able to reconstructs the intrusion process by 2-stage operations (Focusing and Analyzing), and presents the correlation results in the form of evidence graph.

1) FOCUSING STAGE

Using data queries to retrieve all the evidence vectors EV satisfying the specified process PID and the specified time range in the database. We restrict the time range of the PID because the values of the PID, INODE and some other data structures in the operating system take loop count. If the time range is not limited, the process of the evidence vectors with the same PID values may refer to different processes, and impact the accuracy of the correlation results.

2) ANALYZING STAGE

According to the definition of object dependency, we transfer the evidence vectors retrieved in the focusing stage into the forms of object dependencies, and reasoning the evidence graph following algorithm 1. Finally, we remove the incorrect object dependency relationships in the evidence graph with algorithm 2 to improve the credibility and reliability of the correlation results.

IV. EXPERIMENTS EVALUATIONS

To test the performance of the new C2RS framework, a time consumption test on a CentOS 7.3 virtual machine was processed on a physical machine with Intel E5-2620 2.10GHz CPU and 64GB memories. We choose 6 common used commands which are shown in table 2.

Algorithm 1 Evidence Graph Reasoning Algorithm

```
INPUT : The generated evidence vector set (EV) in the
focusing stage.
OUTPUT : Evidence graph EG
1:  foreach element v in set(EV) {
2:          convert v to object dependency Source    Sink;
3:          flag = 0;
4:     foreach object O in graph EG{
5:             IF Source == O {flag ++; BREAK; }
6:  }
7:  IF (flag != 0) add object Source to graph EG;
8:  flag = 0;
9:  foreach object O in graph EG{
10:            IF Sink == O {flag ++; BREAK; }
11:  }
12:  IF (flag!=0) add object Sink to graph EG;
13:  add edge from Source object to Sink object;
14:  }
15:  RETURN EG;
```

Algorithm 2 Evidence Graph Pruning Algorithm

```
INPUT: The generated set EV and EG to be pruned.
OUTPUT: the pruned evidence graph
1:  foreach element v in set(EV) {
2:     IF v.revl == 0 or v.mapped == 0;{
3:       convert v to object dependency Source    Sink;
4:       delete the edge from object Source to object Sink in
         graph EG;
5:  }
6: }
7: RETURN EG;
```

TABLE 2. Commands used in performance test.

Command	Type	Descriptions
linux_bash	Command	Recover bash history from bash process memory
linux_pslist	Process	Gather active tasks by walking the task_struct
linux_psenv	Process	Gathers processes along with their static environment variables
linux_netstat	Network	Lists open sockets
linux_netfilter	Network	Lists Netfilter hooks
linux_dmesg	Kernel	Gather dmesg buffer

Each command is tested for 20 times in which 10 times is running on modified codes and the rest are running on the original codes. Time consumed for each test are recorded. Attention that in the test of modified codes, time cost in initialization process is omitted as for a long-term running process, the initialization is processed only once. The results of the tests are shown in figure 6.

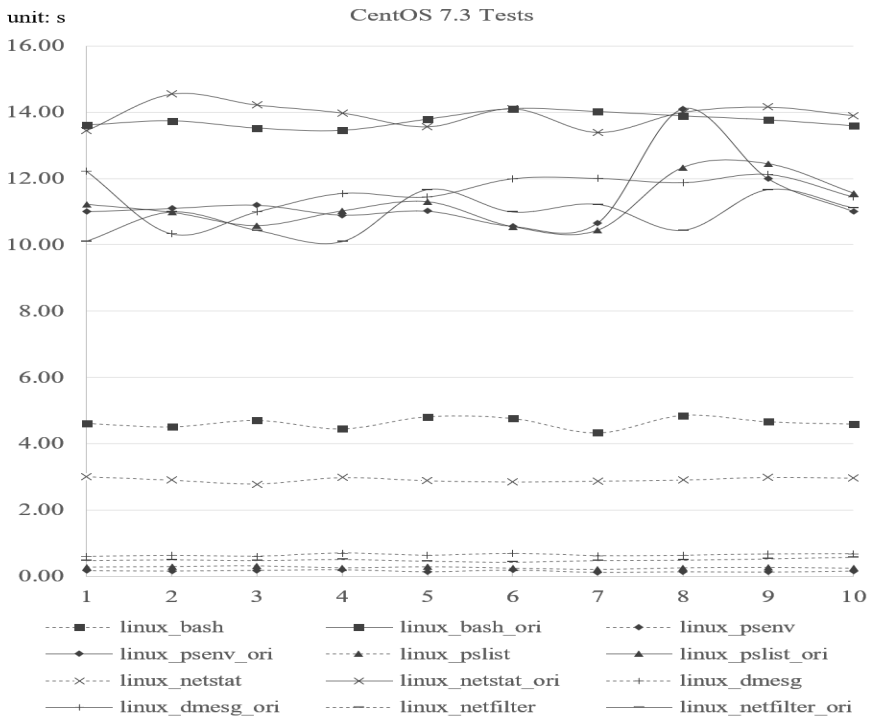


FIGURE 6. Time consumption for C2RS and the original volatility.

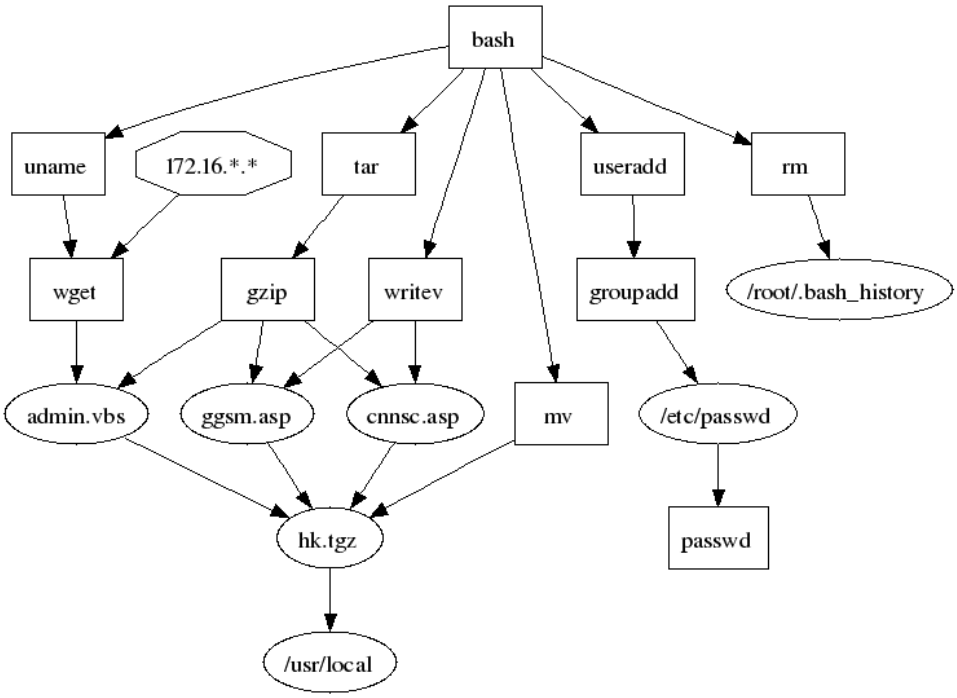


FIGURE 7. Example of intrusion scenario reconstruction.

As shown in figure 6, the modified codes consume little time compared to the original codes. Take pslist for example, the modified codes only cost about 270ms in average, while on contrast, the original codes need 11240ms. As shown in the figure, it can be seen that the overhead of time and space are not expensive.

```
Parts of output for modified volatility:
Initializing:
Volatility Foundation Volatility Framework 2.6
['vol.py', '--profile', 'Linuxcentos7_3_x64x64', '-l', 'vmi://instance-00000570', 'linux_bash', '--tz=PRC']
'{sysmap="/home/sysmap/System.map-3.10.0-514.6.2.el7.x86_64";ostype="Linux";linux_name=0x678;linux_tasks=0x430;li
nux_mm=0x468;linux_pid=0x4a4;linux_pgd=0x58;}'
Found CLS:<class 'volatility.plugins.addrspaces.ProListAddrSpace.ProVmiAddressSpace'>
Found CLS:<class 'volatility.plugins.addrspaces.amd64.LinuxAMD64PagedMemory'>
Final Base_as : <volatility.plugins.addrspaces.amd64.LinuxAMD64PagedMemory object at 0xb234dd0>
Return global_base_as: <volatility.plugins.addrspaces.amd64.LinuxAMD64PagedMemory object at 0xb234dd0>
Return global_base_as: <volatility.plugins.addrspaces.amd64.LinuxAMD64PagedMemory object at 0xb234dd0>
Processing and Output:
Pid    Name    Command Time          Command
------ ------- --------------------- --------------------
...    ...     ...                   ...
4111   bash    2018-03-22 12:01:09   ll
4111   bash    2018-03-22 12:01:11   top
4111   bash    2018-03-22 12:17:23   groupadd test
4111   bash    2018-03-22 12:17:35   simple-command
4111   bash    2018-03-22 12:17:35   dB□
4111   bash    2018-03-22 12:17:35   useradd -g test test
4111   bash    2018-03-22 12:17:54   wget ftp:// 172.16.100.18/SOURCE/test.tar
```

FIGURE 8. Part of the output of 'linux_bash' command.

Figure 7 shows an example of the invasion process of intrusion scenario reconstruction with evidence and correlation operations analysis. The visualization part was performed using the Graphviz tool. We can observe the attacker's invasion process in the figure. It first uses the useradd and groupadd commands to add a system administrator and set the corresponding authority, then reset the user password through passwd. In order to further install the attack script, the attacker uses the command uname to view the kernel version information, and then download the attack script admin.vbs via the FTP on a remote host 172.16.100.18 and write two webshells: cnnsc.asp and ggsm.asp, respectively. The two webshells are used to manipulate files on the server. Next, the attacker packages the vbs script and winshell as hk.tgz by executing the tar and gzip commands, and moves them to the /usr/local directory for backup so that the back door can be used later. Finally, the attacker deletes the /root/.bash_history file and destroys the history command record.

This process is monitored by command of 'linux_bash' with 10s intervals and part of the output is shown in figure 8. From which, we can see that the modified codes load a 'Linuxcentos7_3_×64×64' OS type and connected to VM of instance-00000570 with a sysmap 'System.map-3.10.0-514.6.2.el7.×86_64' locating at '/home/sysmap'. Moreover, the address spaces for this VM and 'linux_bash' show that the layer of 'ProListAddrSpace' is wrapped by 'LinuxAMD64PagedMemory'.

V. CONCLUSIONS

In this paper, based on our previous work of the Heetian Cyber Range, we present a real-time Correlation of host-level events in Cyber Range Service (C2RS) method. C2RS implements an out-of-band data capturing mechanism for greater attack resistance utilizing virtual machine introspection technology. Using this approach allows C2RS to isolate the data capturing from the monitored host. C2RS leverages from these captured data incorporated into the Volatility framework to aid in simplifying the analysis of operating system memory structures. To aid the trainees aware the cybersecurity situational in cyber range, we proposed an object dependence method to achieve the evidence of illegal activity. The proposed C2RS method is scalable and robustness, which makes our method more applicable to other tasks. Correlation of host-level events is just the beginning of a long, complicated investigative process. There are many directions to explore in the future. First, we plan to improving performance and reducing memory usage. Second, additional areas of future work include building a distributed computing implementation with terabytes of log data.

ACKNOWLEDGMENT

The first four authors (Zhihong Tian, Yu Cui, Lun An, and Shen Su) contributed equally to this paper. The authors express their sincere appreciation to the editors and the anonymous reviewers for their helpful comments.

REFERENCES

[1] S. Chapman, R. Smith, L. Maglaras, and H. Janicke, "Can a network attack be simulated in an emulated environment for network security training?" *J. Sens. Actuator Netw.*, vol. 6, no. 3, p. 16, 2017, doi: 10.3390/jsan6030016.

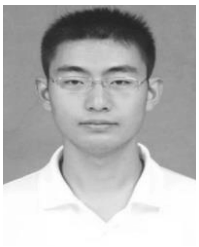
[2] A. Cook, R. Smith, L. Maglaras, and H. Janicke, "Measuring the risk of cyber attack in industrial control systems," in *Proc. 4th Int. Symp. ICS SCADA Cyber Secur. Res.*, Belfast, U.K., Aug. 2016, pp. 23–25.

[3] J. Cheng, R. Xu, X. Tang, V. S. Sheng, and C. Cai, "An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment," *Comput., Mater. Continua*, vol. 55, no. 1, pp. 95–119, 2018, doi: 10.3970/cmc.2018.055.095.

[4] Y. Liu, H. Peng, and J. Wang, "Verifiable diversity ranking search over encrypted outsourced data," *Comput., Mater. Continua*, vol. 55, no. 1, pp. 37–57, 2018, doi: 10.3970/cmc.2018.055.037.

[5] J. Cui, Y. Zhang, Z. Cai, A. Liu, and Y. Li, "Securing display path for security-sensitive applications on mobile devices," *Comput., Mater. Continua*, vol. 55, no. 1, pp. 17–35, 2018, doi: 10.3970/cmc.2018.055.017.
[6] F. Binxing, J. Yan, L. Aiping, and Z. Weizhe, "Cyber ranges: State-of-the-art and research challenges," *J. Cyber Secur.*, vol. 1, no. 3, pp. 1–9, 2016.
[7] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine learning based android malware detection," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2017.2789219.
[8] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Comput. Secur.*, vol. 72, pp. 1–12, Jan. 2018, doi: 10.1016/j.cose.2017.08.007.
[9] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," in *Cluster Computing*. Springer, 2017, pp. 1–10, doi: 10.1007/s10586-017-0849-9.
[10] P. Li *et al.*, "Multi-key privacy-preserving deep learning in cloud computing," *Future Gener. Comput. Syst.*, vol. 74, pp. 76–85, Sep. 2017.
[11] C.-Z. Gao, Q. Cheng, X. Li, and S.-B. Xia, "Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network," in *Cluster Computing*. Springer, 2018, pp. 1–9, doi: 10.1007/s10586-017-1649-y.
[12] *EnCase Forensic Tool*. Accessed: Jun. 27, 2018. [Online]. Available: http://www.guidancesoftware.com
[13] *SafeBack Bit Stream Backup Software*. Accessed: Jun. 27, 2018. [Online]. Available: http://www.forensics-intl.com/safeback.html
[14] G. Vigna and A. Mitchell, "MNEMOSYNE: Designing and implementing network short-term memory," in *Proc. IEEE Int. Conf. Eng. Complex Comput. Syst.*, Dec. 2002, pp. 91–100.
[15] W. Wang and T. Daniels, "Network forensics analysis with evidence graph," in *Proc. Digit. Forensic Res. Workshop (DFRWS)*, New Orleans, LA, USA, 2005, pp. 1–7.
[16] Z. Tian, W. Jiang, and Y. Li, "A transductive scheme based inference techniques for network forensic analysis," *China Commun.*, vol. 12, no. 2, pp. 167–176, Feb. 2015.
[17] Z. Tian, W. Jiang, Y. Li, and L. Dong, "A digital evidence fusion method in network forensics systems with dempster-shafer theory," *China Commun.*, vol. 11, no. 5, pp. 91–97, May 2014.
[18] K. Shanmugasundaram, N. Memon, A. Savant, and H. Bronnimann, "ForNet: A distributed forensics network," in *Proc. 2nd Int. Workshop Math. Methods, Models Architectures Comput. Netw. Secur.*, St. Petersburg, Russia, 2003, pp. 1–16.
[19] R. J. Walls, E. Learned-Miller, and B. N. Levine, "Forensic triage for mobile phones with DEC0DE," in *Proc. 20th USENIX Conf. Secur.* Berkeley, CA, USA: USENIX Association, 2011, p. 7.
[20] Y. Li *et al.*, "Experimental study of fuzzy hashing in Malware clustering analysis," in *Proc. 8th Workshop Cyber Secur. Experimentation Test (CSET)*, 2015, p. 52.
[21] Y. Liao and V. R. Vemuri, "Using text categorization techniques for intrusion detection," in *Proc. 11th USENIX Secur. Symp.*, 2002, pp. 1–10.
[22] C. Ko, T. Fraser, L. Badger, and D. Kilpatrick, "Detecting and countering system intrusions using software wrappers," in *Proc. 9th USENIX Secur. Symp.*, 2000, pp. 1–13.
[23] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *Proc. NDSS*, 2003, pp. 1–16.
[24] N. Joseph, S. Sunny, S. Dija, and K. L. Thomas, "Volatile Internet evidence extraction from windows systems," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res.*, Dec. 2014, pp. 1–5.
[25] F. Jiang *et al.*, "Deep learning based multi-channel intelligent attack detection for data security," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2018.2793284.

ZHIHONG TIAN received the Ph.D. degree from the Harbin Institute of Technology. From 2003 to 2016, he was with the Harbin Institute of Technology. He is currently a Professor, a Ph.D. Supervisor, and the Dean of the Cyberspace Institute of Advanced Technology, Guangzhou University. His current research interests include computer network and network security. He is a member of the China Computer Federation. He is the Standing Director of the Cyber Security Association of China.



YU CUI was born in Harbin, China, in 1985. He received the Ph.D. degree from the Harbin Institute of Technology. He is currently a Senior Engineer and the Technical Manager with the Cyber Range Department, Heetian Information.

LUN AN was born in Xi'an, China, in 1985. He is currently pursuing the Ph.D. degree with the Harbin Institute of Technology. He is currently with the Beijing University of Posts and Telecommunications. His current research interests include cyber range and network security.

SHEN SU was born in Harbin, China, in 1985. He received the Ph.D. degree from the Harbin Institute of Technology. He is currently an Assistant Professor with Guangzhou University. His current research interests include inter-domain routing and security.

XIAOXIA YIN received the Ph.D. degree in electronics engineering from The University of Adelaide, Australia. She was a Visiting Scholar with the University of Reading, Reading, U.K., under the supervision of S. Hadjiloucas, and with the University of Cambridge, Cambridge, U.K., under the supervision of L. F. Gladden. She involved in tumor detection via DCE-MRI with The University of Melbourne, Australia, under the supervision of Prof. Kotagiri. She has an existing collaboration with Prof. M.-Y. Su with the Center for Functional Onco Imaging, University of California at Irvine, USA, and with Prof. T. Kron with the Peter MacCallum Cancer Centre, Australia. She is currently a Research Fellow in high-dimensional medical image analysis with Victoria University, Australia. Her research interests include multiresolution analysis, segmentation, image reconstruction and classification, and their applications to high-dimensional medical imaging. She was a member of the Organizing Committee and the Publication Chairperson of the 3rd International Conference of Health Information Science, and a member of the Organizing Committee and the Program Co-Chairperson of the 4th and 5th International Conference of Health Information Science. She received the Post-Doctoral Research Fellowship from the Australian Research Council in 2009. She is the Managing Editor of the *Health Information Science and System Journal*.

LIHUA YIN was born in Harbin, China, in 1973. She received the Ph.D. degree from the Harbin Institute of Technology. She is currently a Professor and a Ph.D. Supervisor with Guangzhou University. Her current research interests include computer network and network security. She is a member of the China Computer Federation.

XIANG CUI received the master's degree from the Harbin Institute of Technology in 2003 and the Ph.D. degree in information security from the Institute of Computing Technology, Chinese Academy of Sciences, in 2012. He focused on the cybersecurity emergency response. From 2007 to 2017, he was with the CAS Institute of Computing, Chinese Academy of Sciences, where he focused on the cyber-attack and defense technologies. He is currently a Guangzhou University Cyberspace Advanced Technology Research Fellow. He participated in a series of major cyber-security incident handling and 863-917 platform construction. His research areas include network offensive and defensive techniques.