RRPOT: A Record and Replay Based Honeypot System

Chunlai Du¹, Shichuan Zhao¹², Wei Wang²*

¹ School of Information Science and Technology, North China University of Technology, Beijing, China
² Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
*duchunlai@ncut.edu.cn

Abstract. The honeypot system is a critical defense facility in cybersecurity. It captures and analyzes the attack behavior for constructing a more robust defense system. However, considering the overhead caused by analysis, the system cannot support fine-grained analysis. This paper proposes RRPOT, a novel honeypot system based on the record and replay (RnR) framework of virtual machine (VM). By combining the native honeypot system with an RnR framework, the system supports to postpone the time-consuming analysis to the replay stage, thereby enables fine-grained analysis. Moreover, we propose an on-demand record mechanism to reduce performance and storage costs during the record stage and employ an on-demand replay mechanism to improve replay efficiency. Leveraging these approaches, we develop a flexible and efficient honeypot system, and it was proven useful in the practical scenario.

Keywords: Virtual Machine, Record and Replay, Cybersecurity, Honeypot

1. Introduction
Due to the rapid growth of internet technology, cybersecurity issue occurs frequently, thus attracts the attention of many people. The honeypot system is an essential defense facility in cybersecurity, which is compromised and placed in a vital area of the network to get information about the attacker [1]. Leveraging preset baits, honeypot induces the attacker to attack it. Then it detects and captures malicious traffic, and records them for further analysis. In this way, the honeypot system could assist the analyzer in tracing the source of malicious programs and attack traffic. Furthermore, it helps the defender to find security threats they are facing, thus provides effective guidance for improving defense strategy. In recent years, with the development of virtualization technology, virtualization infrastructures have become a mainstream platform for the honeypot system [2, 3]. On this platform, honeypots are VMs that provides an isolated and scaled computing paradigm, rather than physical machines [4], thereby inheriting advances of VM, such as flexibility, elasticity, cost efficiency, and isolation.

Unfortunately, there are shortcomings in the practical honeypot system. First of all, due to the uncontrollable factors during the system execution, many attack processes cannot be reproduced in experimental environments. For example, caused by the data structure characteristics of the heap, there are many uncontrollable factors in the attacks against the heap. Thus, it is difficult to reproduce these attacks accurately. Secondly, considering that the analysis would introduce a performance penalty, the honeypot system cannot support fine-grained analysis.
With these in mind, we propose to construct our honeypot system based on the record and replay (RnR) framework of virtual machine (VM) called RRPOT. The RnR framework provides the ability to reproduce the previous execution of the whole VM system deterministically. It logs all non-deterministic events that the system receives in the hypervisor and enables the reproduction of the system’s execution. Based on the RnR framework, RRPOT is capable of replaying the execution flow of the recorded honeypot, thereby supports to reproduce the attack process against zero-day vulnerability efficiently. Moreover, we can decouple the diagnosis from the normal execution of honeypot and postpone it to the replay stage. It would moderate the performance penalty of the honeypot system and enable fine-grained analysis at the same time.

The rest of the paper is organized as follows. Section 2 introduces the overall architecture of RRPOT. Then, Section 3 presents the implementation details. Section 4 illustrates how RRPOT works in practical usage. And we conclude RRPOT in Section 5.

2. Design of RRPOT

RRPOT is a honeypot system based on the RnR framework of the VM system. Its architecture consists of five modules, i.e., Controller, Introspection Component, Record Component, Replay Component, and Analyzer, as illustrated in Fig.1. Specifically, the honeypot system is composed of several honeypots, which are VMs. Moreover, the Controller is an interface for users to control the honeypot system through VMM. The introspection Component is responsible for detecting sensitive behavior generated in the honeypot. Upon receiving a record command from Controller or discovering a sensitive behavior by the Introspection component, the Record Component starts working. It captures all non-deterministic events of honeypots during the execution and logs these events into the storage system. According to the practical requirement, the Controller could send a replay signal to Replay Component. Once receiving the signal, the Replay Component would create a Replayer, read the corresponding logfile from the storage system, and then reproduce the execution honeypot according to the information recorded in log files. Thus, Replayer has the same execution flow as the recorded honeypot. Therefore, Analyzer can detect malicious behaviors in honeypot by analyzing the execution of the Replayer.

Different from the native honeypot system, our RRPOT supports the execution flow reproduction of honeypot when needed, and enables us to postpone analysis to the replay stage. Although the idea is simple, there exist several challenges to realize RRPOT. Firstly, the overhead introduced by the recording process affects the practicability of the system. To reduce the costs, we propose an on-demand record strategy to minimize unnecessary records. Secondly, analysis is usually time-consuming. However, the analysis performance directly affects the system response time to the attack. To improve the analysis performance of the honeypot system, we employ an on-demand replay mechanism to narrow down the replay range. In the following subsections, we will present the details.
Fig. 1 Architecture Overview of RRPO T

2.1. On-demand Record
As we mentioned above, it would introduce a lot of unnecessary costs if the system records the whole execution flow of honeypot. Therefore, we proposed two kinds of modes to trigger the record on demand, i.e., manual mode and automatic mode.

In manual mode, the operator determines the starting point according to their experience. When needed, operators manually send recording commands to VMM by Controller, and then the VMM forwards the signal to the Record Component.

In automatic mode, the record is triggered by the Introspection Component. The introspection Component is responsible for monitoring the behavior of honeypot in a coarse-grained way. Introspection Component detects malicious behavior referring to preset rules, e.g., execution of sensitive commands, creation of sensitive threads, access to sensitive files, etc. Once some suspicious behavior is found, Introspection Component sends a signal to notify the Record Component to start the record.

Upon receiving the record command from VMM or Controller, Record Component creates a record thread. The record thread takes charge of capturing non-deterministic events during the honeypot running and then records them to the storage system for later replay.

2.2. On-demand replay
To reduce unnecessary replay, the system should determine the replay start point at first before replay. To solve this problem, we use an automatic tool to conduct a preliminary analysis of the honeypot log. Through the analysis, we can get the range of execution fragment that we are interested in. With this result, we use the interface provided by the Controller to notify the Replay Component. Upon receiving the replay command, the Replay Component creates a Replayer and initializes its state to the time point just before the execution fragment. Then, the Replay Component fetches the events from the log file and feeds them into the Replayer to replay the execution of honeypot. While replaying, Analyzer can analyze the behavior of the whole honeypot system leveraging analysis tools. It is worth noting that the execution of the Replayer and the recorded honeypot are entirely independent, and the analysis process is performed on the Replayer. Therefore, the analysis process would not affect the original honeypot system. Based on this framework, the system can support analysis in various granularities and multi-dimension.

3. Implementation
We implement the RRPO T based on Qemu-KVM. In this section, we will present some key details.

3.1. Record and Replay for VM
The execution of a system is uncertain, which is affected by a lot of non-deterministic events. Record Component is responsible for recording those non-deterministic events during the honeypot execution. This component records the non-deterministic events captured by VMM in a buffer, and then flush them into the permanent storage system when the buffer is full. In the replay stage, Replay Component creates a honeypot Replayer to replay the previous execution of the honeypot according to the recorded information. It fetches non-deterministic events from the log file and feeds them into Replayer through VMM. According to logged content, the execution of the honeypot will be rebuilt.

According to the replay approach, non-deterministic events can be classified into three types, i.e., synchronous events, asynchronous events, and compound events [5]. They are recorded and replayed in the following different manners.

1) Synchronous events, such as I/O events and RDTSC instructions, are issued by the guest VM to devices. We call these events synchronous since they will always be triggered during the replay stage by the associated instructions at the same time. However, their contents are non-deterministic, thus the only thing we have to record is the content of these events. That is to say, for I/O events
such as PIO (Port-based I/O) events, MMIO (Memory-mapped I/O) events, guest VM automatically replicates these events during execution. For RDTSC events, which provides precise time measurements in a virtual environment, the value of each "RDTSC" instruction would be recorded as a timestamp for the guest operating system, so that the same time guest state can be rendered during a replay. In the replay stage, to replay synchronous events, the Replay Component only needs to fill the corresponding area of the Replayer with the logged content when they are triggered.

2) Asynchronous events, such as external interrupts, are generated from external devices. These events may occur at an arbitrary time, which means the timestamp is also non-deterministic. We should record the timestamp along with the contents. In the replay stage, the Replay Component should wait until the execution of the Replayer reaches the logged timestamp, and then injects these events into the Replayer.

3) Compound events are relatively complicated. Compound events refer to those events whose time and content are both non-deterministic. Different from asynchronous events, compound events can only be seen after the synchronous events or asynchronous events occur. Therefore, compound events are tied to synchronous events or asynchronous events. Thus, in the replay stage, we can inject a Compound Event following the manner of either Synchronous Event or Asynchronous Event. Here we take the network packet as an example. The arrival time and payload of the network packet are non-deterministic, but its arrival will first trigger an interrupt, which means the network packet is tied to an interrupt event. The network packet would be replayed following the injection of the interrupt event to which it is tied to. As a result, the content of Compound Events is the only thing to be recorded. In the replay stage, these events are injected following the injection of events it is tied to in the replay stage.

3.2. Honeypot system
In cybersecurity, mainly caused by information asymmetry, the defender is always weaker than the attacker. Specifically, the tools used by attackers are becoming more and more advanced, and the software vulnerabilities keep popping up, while the defender only performs passive defense. The honeypot technology is put forward, aiming at combining strategic defense and tactical offense. It actively sets up decoys and traps to attract attackers and then captures their behaviors for analysis. In this way, the defender could have a comprehensive understanding of vulnerabilities in the system. Furthermore, the defender could obtain a lot of information about the attacker, including identity, intentions, approach and etc.

For a honeypot system, the first thing we need to do is disguising it as a real business server. Specifically, we open service ports, including web service (port 80) and mail service (port 110), and set up fake databases. Inside the honeypot system, we put some phishing files, such as some encrypted confidential files, to induce attackers to download and view. In addition, we disguise operating system information, erase the virtualization features of the honeypot, such as the process of vmtools, the MAC address of the virtual network card. These disguising methods make it is difficult for the attacker to identify the authenticity of the honeypot.

After deploying the honeypot environment, the next step is to wait for the attacker and collect their behaviors. The following process module can be divided into an information collector, replay controller, and data analyzer. The details are as follows.

1) The information collector collects packets from the sniffer of a honeypot and stores them into the database after processing. Processing methods include BP neural network [6], support vector machine [7], and regular expressions [8]. This information will serve as an essential reference for whether the Record Component would save the recorded behavior of honeypot.

2) The replay controller mainly provides analysts with the replay of attack execution scenarios and offers the interface to control the start, pause, and stop of the replay process. Moreover, it provides a debugging environment, such as gdb [9].

3) The data analyzer analyzes the malicious data from the honeypot and visualizes them. Then it feeds the results back to the manager through Web service. In this system, it plays a crucial role in dis-
covering the characteristics of attackers' behavior, identifying potential risks, and providing a significant reference for whether to maintain the logs.

4. Use Case
The honeypot system is vital defense equipment in cybersecurity. With the assistance of RnR technology, the honeypot system becomes more powerful. In this section, we will present the application of our RRPO in the practical defense environment.

As illustrated in Fig.2, our honeypot system RRPO is deployed in the intranet and disguised as a regular server. And in the honeypot, we have preset some of the vulnerabilities, e.g., MS17-010, which would attract the attack to exploit. Usually, before attacking, attackers would scan all hosts in the network to search for hosts with vulnerabilities (step1). Upon detecting the scanning traffic, RRPO starts the Record Component to log the execution flow of honeypot and save them into the storage system (step2). Then, the attacker discovers the vulnerability and tries to exploit our honeypot (step3). All this behavior would be recorded by the Record Component, and a warning would be sent to the Honeypot monitor (step4). The honeypot monitor is responsible for monitoring these warnings and notify the administrator if necessary. After the completion of the emergency response, the administrator or auditor can replay the recorded honeypot for further analysis through the interface offered by Honeypot Manager (step5).

Upon receiving the command, the Honeypot Manager would start a Honeypot Replayer to reproduces its previous execution (step6). During replaying, Analyzer analyzes the behavior of the Honeypot Replayer (step7) and returns the results to the administrator (step8).

In the practical environment, the system may not be able to judge whether the data traffic is malicious in a short period, such as zero-day attacks. In this case, comprehensive judgment should be made based on its subsequent actions in a period. When it is judged harmless, the following record makes little sense and would bring extra overhead to the system. Therefore, the record could be stopped. Based on the above ideas, we propose a feedback-based recording mechanism.

Feedback Based Recording Mechanism. As illustrated in Fig.3, the mechanism involves three components, i.e., Honeypot, Honeypot Manager, and Analyzer. Once the monitor in Honeypot receiving unrecognized data traffic, the system starts recording right away and reports the warning to the Honeypot Manager. Receiving the warning, the Honeypot Manager creates a Honeypot Replayer im-
mediately and starts an Analyzer thread to perform analysis to the Honeypot Replayer. After a while, the Analyzer returns the results to the Honeypot Manager. If the results show that the traffic is harmless, the Honeypot Manager will notify the Record Component to stop the record. If it is malicious, the Honeypot Manager reports the administrator to adopt some defensive strategies. Otherwise, if it remains undetermined, the Honeypot Manager will ask the Analyzer to continue its analysis.

![Diagram of the feedback based recording mechanism]

**Fig. 3** Feedback based recording mechanism

5. **Conclusion**

This paper presents an efficient honeypot system named RRPOt. Cooperating with the record and replay technology of VM, it supports to reproduce the previous execution of recorded honeypot deterministically. Leveraging this advance, the system can support a more precise analysis by postponing the analysis process to the replay stage. In addition, we propose an on-demand record mechanism to reduce costs introduced by the record and employ an on-demand replay mechanism to improve replay efficiency. In practical application scenarios, the RRPOt system is proven to be useful and flexible. This paper proposes to combine virtualization technology with network defense technology, thereby develops a new idea in the construction of defense systems.

**Acknowledgments**

This work was supported in part by the "Practical Training Plan" for Cross Training of High-Level Talents in Beijing Universities (Grant no. PXM2020_014212_000020).
References

[1] Mairh, Abhishek and et al 2011 Honeypot in network security  a survey International conference on communication computing & security

[2] Weizhe Z, Hui H and Tai-hoon Kim 2015 Xen-based virtual honeypot system for smart device Multimedia Tools and Applications 74 pp8541-8558

[3] Jianchao H and Ying H 2018 Research on network defense strategy based on honey pot technology IOP Conference Series Materials Science and Engineering

[4] Hamdaqa M and L Tahvildari 2012 Cloud computing uncovered: a research landscape in Advances in Computers Elsevier 86 pp41–85

[5] Shalabi Y, Yan M and NHonarmand 2018 Record-replay architecture as a general security framework in Proceedings of HPCA IEEE

[6] A Joshi, S T King, G W Dunlap and P M Chen 2005 Detecting Past and Present Intrusions Through Vulnerability-specific Predicates in Proceedings of SOSP

[7] R Senthilkumaran and P Kulkarni 2012 Implementation and applications of virtual machine record and replay Department of Computer Science and Engineering Indian Institute of Technology Tech Rep 113050055

[8] Aiping Q 2005 Network Behavior Analysis Based on Multi-distribute Agent Radial Basic Function Neural Net Computer Engineering and Applications

[9] Santos R, Souza D, Santo W and et al 2019 Machine learning algorithms to detect DDoS attacks in SDN J. Concurrency and Computation: Practice and Experience 5402

[10] Sovilj D, Budnarain P, Sanner S and et al 2020 A Comparative Evaluation of Unsupervised Deep Architectures for Intrusion Detection in Sequential Data Streams Expert Systems with Applications 113577