AWESOME - Automated Web Emulation for Secure Operation of a Malware-Analysis Environment

Martin Brunner, Christian Martin Fuchs and Sascha Todt
Fraunhofer Research Institution for Applied and Integrated Security (AISEC)
Munich/Garching, Germany
{*martin.brunner,christian.fuchs,sascha.todt*}*@aisec.fraunhofer.de*

***Abstract*—We present AWESOME, a novel approach for integrated honeypot-based malware collection and analysis which extends the functionalities of existing approaches. In contrast to purely network-based approaches, the goal of our collection and analysis system is runtime retrieval of internal malware logic information. This approach allows us to provide analyzed malware with all requested resources in real time, despite the fact that it is executed within an isolated environment. Our assumption is that being able to track the entire malware execution life-cycle will enable a better understanding of current and emerging malware. This paper introduces our design, outlining its contributions and design considerations. An in-depth description and evaluation of each component will be discussed in separate work. While still under development, we expect our approach to make a significant contribution to enhanced analysis of current malware.**

Keywords-malware collection; malware analysis and defense.

I. Introduction and Related Work

Some of today's most disruptive cyber threats can be attributed to malware, usually organized within a botnet in large-scale scenarios. This results in a fundamental need to track the rapid evolution of malware, which, in turn, depends on collection and examination of current real-world attack data, commonly acquired through meticulous analysis of the most recent samples. The evolution of malware over time has led to the development of intensive obfuscation and anti-debugging mechanisms, as well as a complex and multi-staged malware execution life-cycle [16]. Usually, this life-cycle can be partitioned into three phases: (i) *propagation and exploitation*, which covers the spread of malicious payloads and unpacking and deobfuscation of the corresponding shellcode; (ii) *infection and installation* covers the deployment of a binary (e.g., dropper) on the victim host followed by possible preparatory activities and the retrieval of the actual malware body; and, (iii) the *operation and maintenance* phase, in which the malware's core components harvest valuable information and attempt to establish a command-and-control (C&C) channel awaiting further instructions. Each phase may include numerous measures aimed at maximizing installation success and reliability. In order to comprehensively analyze the full life-cycle, the malware under analysis must have unhindered access to all requested resources during runtime. While this could easily be achieved by allowing full interaction with the Internet, this is not a viable approach in setups which are forced to consider liability issues. Advanced large-scale malware collection and analysis infrastructures, such as [1][5][8], can satisfy the requirements for automated tracking of malware, but suffer from several limitations:

(i) Despite being executed within an isolated environment, samples must be supplied with requested network services during analysis. Otherwise, achieving high-quality results is impeded, potentially causing different malware behavior or even a refusal of execution. While certain services can be offered using sinkholing techniques, existing approaches are purely network-based, and reactions to malware-initiated connection attempts remain static during runtime. Predefined commonly-used services, which are usually queried, are offered by these infrastructures to the malware; however, other requests can not be handled accordingly.

(ii) High-interaction (HI) honeypots pose, aside from their complexity and maintenance issues, high operational risks. These are often inadequately addressed. While there are many methods of mitigation, the remaining risk is still higher than with low-interaction (LI) honeypots, resulting in ethical questions and possibly even legal and liability issues for the operating organization.

(iii) Existing approaches separate collection and analysis, thus forfeiting the system context (i.e., file handles, requests, sockets) of the victim host. While such separation is not necessarily a limitation (it may not be mandatory to gain qualitative analysis results), we argue that this loss of information hinders analysis and may degrade analysis results or prevent analysis of certain malware altogether.

II. Approach

A. Goals

Our overall goal is to capture and dynamically analyze novel malware on a large scale. To identify trends of current and emerging malware, we aim to cover the entire life-cycle. That is, we want to track malware communicating via unknown (e.g., C&C) protocols in an automated way within a controlled environment. In order to minimize harm to third parties, malware should by default have no Internet access during analysis. The whole procedure intends to trick

 ISBN: 978-1-61208-209-7

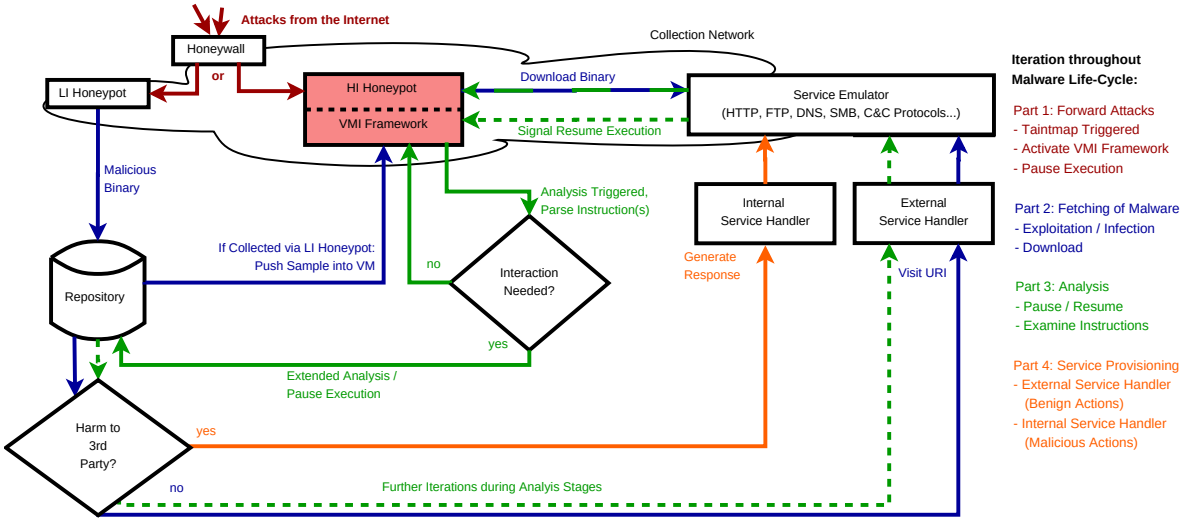

Figure 1. General Design of the Presented Approach

a sample into believing it is running on a real victim host with full Internet access.

B. Basic Concept

To identify the services and protocols required in the next step of the life-cycle, we intend to harvest information on internal malware logic during execution. In contrast to purely network-based approaches, our method also operates at the binary level, directly interacting with the malware's host system. It therefore aims to integrate network-based analysis and binary analysis, as in [21]. As depicted in Figure 1, the presented approach is based on a HI honeypot and a virtual machine introspection (VMI) framework. We enhance our architecture with a transparent pause/resume functionality, which is instrumented to determine and, if needed, interrupt the program flow. Hence, we enable the extraction and alteration of program logic and data within the victim environment during runtime. This is specifically valuable for extracting protocol information and cryptographic material embedded within malware in order to determine the protocol type and intercept encrypted communication. After checking, extracted information is forwarded to a service handler (SH) and sinkholing service in order to maintain full control over all interactions between the malware and the outside world. For handling unknown traffic as well, finite state machines (FSM) are automatically derived from the observed traffic and used for service emulation. An important goal of automating the whole collection and analysis process is to handle large amounts of malware while allowing scalability.

C. Added Value

The system context of the malware collection facility persists and is also used in the subsequent analysis. The capabilities resulting from the merge of collection and analysis is similar to the approach used in HI honeypots. Thus, it is more closely aligned to real-world scenarios than LI honeypots. In addition, we achieve increased transparency during analysis due to the use of VMI. We consider this to be a benefit, since we argue that VMI based analysis is more likely to remain undetected by malware. Compared to other techniques, VMI requires no trusted support components which could be compromised [7] inside the sample's context of execution. Hence, the approach is more likely to observe the entire malware execution life-cycle. Furthermore, we are able to extract and inject data as well as instructions from or into the memory of the infected virtual machine (VM) during runtime (for example, in order to tap and manipulate encrypted C&C traffic). Since our approach does not depend on analysis components within the VM, we believe it to be more secure while also expecting better overall performance. Moreover, we are able to control any interaction between malware and third party systems. Thus, our architecture can fulfill legal and liability constraints. Since our approach is applied directly at the instruction level, we are aware of the actions initiated by the malware, thus allowing us to provide matching services and even to service novel communication patterns. Subsequently, the risk resulting from HI honeypot operation is minimized.

III. Design and Implementation

A. Components

Our approach utilizes the following components:

For *malware collection*, a modified HI honeypot [18] is used. *Malware analysis* is conducted based upon Nitro [17], a KVM-based framework for tracing system calls via VMI. In particular, we determine whether a given action initiated by the currently-analyzed malware requires Internet access and thus apply a complex rule-set to the tracing component.

 ISBN: 978-1-61208-209-7

Our *service provisioning* component manages all malware-initiated attempts to request Internet resources. Malicious attempts are handled via an appropriate sinkholing service spawned by Honeyd [19], and unknown traffic patterns may be handled utilizing ScriptGen [13].

B. Setup

While most popular LI honeypots have proven to be efficient for malware collection, their knowledge-based approach has also drawbacks regarding the quantity and diversity of the collected malware [23]. With respect to our primary goal (to handle unknown malware), we chose to apply the taint-map-based approach of ARGOS, since it allows the detection of both known and unknown (0-day) attacks. In addition, it is independent of special collection mechanisms. Moreover, it can cooperate with the KVM based VMI framework Nitro. Hence, several components were modified: (i) the victim VM's RTC is detached from the host's clock, since ARGOS is more time-consuming than traditional approaches and thus detectable by an abnormal latency and timing-behavior; (ii) once the taint-map reports tainted memory being executed, we activate the analysis functionality provided by the VMI framework; and, (iii) simple interpretation and filtering of system calls and their parameters is conducted directly within hypervisor space, while more complex analysis is performed via the VMM in the host environment [10]. The entire process consists of three parts (collection, analysis, and service provisioning) and is structured as described below. The steps are repeated iteratively throughout the entire life-cycle of the malware.

C. Malware Collection

To overcome the poor performance of ARGOS, we build a two-staged malware collection network. That is, we deploy a hybrid honeypot system similar to existing approaches, such as [2][11][22]. We then take advantage of our preexisting honeyfarm infrastructure [3], which utilizes a large-scale network telescope employing various different LI honeypots. This infrastructure is used to filter noise and known (and thus uninteresting) attacks. Only novel incidents are forwarded to ARGOS, thus reducing the overall load on it.

D. Malware Analysis

Dynamic malware analysis utilizing virtualization can be recognized and thus evaded by environment-sensitive malware [9][10][14][20]. Hence, our goal is to achieve a reasonably transparent dynamic malware analysis; however, we also consider VMI as the most promising available approach to evade malware's anti-debugging measures. Thus, in order to provide the best chance at evading detection while still gaining the benefits of VMI, we have chosen Nitro since it offers several advantages regarding performance and functionality when compared to other publicly available tools such as Ether (see [17]). As Nitro is based on KVM, we have, in addition to guest portability, full virtualization capability, thanks to the host CPU's virtualization extensions; thus, we can expect reasonable performance. During the analysis process, we expect a malicious binary to be shellcode or a dropper rather than the actual malware binary. This initially retrieved binary is then decoded and usually contains a URL pointing at the resource used for deploying the next stage of the malware. In the second iteration, execution of this binary continues after it has been downloaded and the VM has been resumed. The resulting system call trace is then examined for routines related to connection handling (e.g., NTConnectPort). If present, we transparently pause execution of the VM and forward related traffic to the service provisioning component.

E. Service Provisioning

Malware-driven outbound requests are evaluated to prevent harm to third party systems. For these checks, we rely upon existing measures, such as IDSs or a web application firewall. We are aware that such measures will not be sufficient to tell benign and malicious flows apart in every case; thus, we may build on existing approaches [12]. We assume that a purely passive request (e.g., a download) does not cause harm to a third party. It is thus considered to be benign and handed over to the *external service handler* (see Figure 1). Since the external SH has Internet access, it resides in a dedicated network segment separated from the analysis environment. If a given request can not be determined to be benign, it is redirected to the *internal service handler*. The sole task of these SHs is to fetch, prepare and provide information for the *service emulator* (SE). The SE launches the requested service in order to deliver the appropriate payload supplied by the SH. Afterwards, execution is transparently resumed. Since these services can be extremely heterogeneous, the SE is based on Honeyd. It is a very flexible and scalable tool which is able to emulate or spawn arbitrary services, given that a protocol template exists.

The creation of templates for novel protocols is a much more challenging task. Therefore we plan to instrument a tool, which derives FSMs from observed traffic, such as *ScriptGen* or a similar approach [4][6][15]. Each FSM represents the behavior of a given protocol at an abstract level while not depending on prior knowledge or protocol semantics. Based on the generated FSMs, service emulation scripts for the SE can be derived. By integrating such a tool into our approach, we aim toward adding 'self-learning capabilities' to the service provisioning element. Obviously this requires (at least) one-time observation of a given communication between the honeypot and the external system. Hence we need a (supervised) back-channel for learning about novel protocols. Once a corresponding communication has been recorded and the appropriate FSM has been generated, we are able to handle the new protocol as well. While the need

 ISBN: 978-1-61208-209-7

for a back-channel is a clear limitation, we consider it to be a reasonable trade-off.

IV. Summary and Future Work

In this paper, we have presented a novel approach for integrated honeypot-based malware collection and analysis which extends existing functionalities. Specifically, it addresses the separation of collection and analysis, the limitations of service emulation, and the operational risk of HI honeypots. The key contribution of the approach is the design of the framework as well as the integration and extension of the stated tools. While this is an ongoing research activity and thus still under development, several modifications to ARGOS and Nitro have already been implemented and successfully tested, indicating the feasibility of our approach. Future work will include the completion and evaluation of the service emulator and the measures to prevent harm to third party systems.

References

[1] M. Apel, J. Biskup, U. Flegel, and M. Meier. Early warning system on a national level - project amsel. In *Proc. of the European Workshop on Internet Early Warning and Network Intelligence (EWNI 2010)*, January 2010.

[2] M. Bailey, E. Cooke, D. Watson, F. Jahanian, and N. Provos. A hybrid honeypot architecture for scalable network monitoring. In *Technical Report CSE-TR-499-04*, 2006.

[3] M. Brunner, M. Epah, H. Hofinger, C. Roblee, P. Schoo, and S. Todt. The fraunhofer aisec malware analysis laboratory - establishing a secured, honeynet-based cyber threat analysis and research environment. Technical report, Fraunhofer AISEC, September 2010.

[4] J. Caballero, P. Poosankam, C. Kreibich, and D. Song. Dispatcher: enabling active botnet infiltration using automatic protocol reverse-engineering. In *Proc. of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 621–634, New York, NY, USA, 2009. ACM.

[5] D. Cavalca and E. Goldoni. Hive: an open infrastructure for malware collection and analysis. In *proc. of the 1st Workshop on Open Source Software for Computer and Network Forensics*, 2008.

[6] W. Cui, V. Paxson, Nicholas C. Weaver, and Y H. Katz. Protocol-independent adaptive replay of application dialog. In *13th Annual Network and Distributed System Security Symposium (NDSS)*, 2006.

[7] M. Dornseif, T. Holz, and C.N. Klein. Nosebreak - attacking honeynets. In *Information Assurance Workshop, 2004. Proc. from the Fifth Annual IEEE SMC*, June 2004.

[8] M. Engelberth, F. Freiling, J. Göbel, C. Gorecki, T. Holz, R. Hund, P. Trinius, and C. Willems. The inmas approach. In *1st European Workshop on Internet Early Warning and Network Intelligence (EWNI)*, 2010.

[9] P. Ferrie. Attacks on virtual machine emulators. In *AVAR Conference, Auckland*. Symantec Advanced Threat Research, December 2006.

[10] C. M. Fuchs. Deployment of binary level protocol identification for malware analysis and collection environments. Bacherlor's thesis, Upper Austria University of Applied Sciences Hagenberg, May 2011.

[11] X. Jiang and D. Xu. Collapsar: a vm-based architecture for network attack detention center. In *Proc. of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, Berkeley, CA, USA, 2004. USENIX Association.

[12] C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson. Gq: practical containment for measuring modern malware systems. In *Proc. of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 397–412, New York, NY, USA, 2011. ACM.

[13] C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. In *Proc. of the 21st Annual Computer Security Applications Conference (ACSAC)*, Washington, DC, USA, 2005. IEEE.

[14] M. Lindorfer, C. Kolbitsch, and P. Milani Comparetti. Detecting environment-sensitive malware. In *Recent Advances in Intrusion Detection (RAID) Symposium*, 2011.

[15] P. Milani Comparetti, G. Wondracek, C. Kruegel, and E. Kirda. Prospex: Protocol specification extraction. In *Proc. of the 30th IEEE Symposium on Security and Privacy*, pages 110–125, Washington, DC, USA, 2009.

[16] G. Ollmann. Behind today's crimeware installation lifecycle: How advanced malware morphs to remain stealthy and persistent. Whitepaper, Damballa, 2011.

[17] J. Pfoh, C. Schneider, and C. Eckert. Nitro: Hardware-based system call tracing for virtual machines. In *Advances in Information and Computer Security*, volume 7038 of *Lecture Notes in Computer Science*. Springer, November 2011.

[18] G. Portokalidis, A. Slowinska, and H. Bos. Argos: an emulator for fingerprinting zero-day attacks. In *Proc. ACM SIGOPS EUROSYS'2006*, Leuven, Belgium, April 2006.

[19] Niels Provos. A virtual honeypot framework. In *Proc. of the 13th USENIX Security Symposium*, 2004.

[20] J. Rutkowska. Red pill... or how to detect vmm using (almost) one cpu instruction, 2004. http://invisiblethings.org [retrieved: July, 2011].

[21] D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. Gyung Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena. Bitblaze: A new approach to computer security via binary analysis. In *Proc. of the 4th International Conference on Information Systems Security.*, 2008.

[22] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. In *Proc. of the 20th ACM Symposium on Operating Systems Principles*, SOSP '05, New York, NY, USA, 2005. ACM.

[23] J. Zhuge, T. Holz, X. Han, C. Song, and W. Zou. Collecting autonomous spreading malware using high-interaction honeypots. In *Proc. of the 9th International Conference on Information and Communications Security*, ICICS'07, Berlin, Heidelberg, 2007. Springer-Verlag.

 ISBN: 978-1-61208-209-7