MARS: An SDN-Based Malware Analysis Solution

João Marcelo Ceron*, Cíntia Borges Margi*, Lisandro Zambenedetti Granville†
*University of São Paulo
Av. Professor Luciano Gualberto, travessa 3, 158 – São Paulo, SP –Brazil
{ceron, cintia}@usp.br
†Federal University of Rio Grande do Sul –
Av. Bento Gonçalves, 9500 – Porto Alegre, RS – Brazil
granville@inf.ufrgs.br

***Abstract*—Mechanisms to detect and analyze malicious software are essential to improve security systems. Current security mechanisms have limited success in detecting sophisticated malicious software. More than to evade analysis system, many malwares require specific conditions to activate their actions in the target system. The flexibility of Software-Defined Networking (SDN) provides an opportunity to develop a malware analysis architecture integrating different systems and networks profile configuration. In this paper we design an architecture specialized in malware analysis using SDN to dynamically reconfigure the network environment based on malware actions. As result, we demonstrate that our solution can trigger more malware's events than traditional solutions that do not consider sandbox surround environment as an important component in malware analysis.**

I. Introduction

Malwares pose a major threat to computer systems [1]. Malwares are continuously evolving, increasing their complexity and sophistication [2]. New malwares arise extremely often, forcing the constant update of defensive mechanisms. To cope with this situation, the research community has proposed and deployed tools to identify malicious software and improve current security mechanisms. One of the most employed approach is dynamic analysis [3]. That approach consist in executing the suspicious software in a *sandbox* (instrumented system) in a controlled environment and, based on the actions performed in system, identify software's malicious behavior [3].

Modern malwares, however, do not promptly reveal their malicious behavior. Such malwares are able to detect the environment where they are running. Malwares can, for example, fingerprint the infected host by inspecting registry keys and running processes, and detect whether that host is actually a malware detector [4]. Advanced malwares do not restrict their observation to the infected host; in fact, they also consider the whole execution environment that surrounds them. Some malwares, for example, require network connectivity and specific services available in the execution environment before initiating malicious activities. Unless these conditions exist, those malwares expose inoffensive behavior that prevents them from being detected. This is typical of *targeted* malwares, which are malwares carefully designed to compromise specific targets such as companies, governments, and manufacturers. For example, Stuxnet [1] is a remarkable malware that remained dormant until it found a specific hardware component present in the local network.

Given the current context, modern malwares demand flexible malware-analysis solutions. Flexible solutions can provide different analysis environments that can trigger the disguise malware's malicious behavior. Instead of focusing in the sandbox like several studies [4, 3, 5–7] do, we focus in the environment that surrounds the sandbox. In MARS – MAlwaRe Analysis Architecture based on SDN – we specifically tackle the problem of dynamic analyzing advanced malwares that are network environment sensitive.

The hypothesis of our work is that executing the same malware in different networking scenarios could reveal distinct behavioral profiles. Thus, through our solution, we are able to analyze malwares that would otherwise remain dormant in other analysis tools. We implement a prototype as a proof of concept; a software component that can inspect the network traffic and based on a set of rules can modify the network environment and identify new behaviors. To materialize different networking scenarios in a flexible way, we employed Software-Defined Networking (SDN) technologies [8]. More precisely, by using the OpenFlow protocol [9], we managed the malware analysis architecture in a centralized fashion, providing dynamic network resource allocation and containment polices. The main contributions of this work comprises:

- An architecture for malware analysis that dynamically reconfigure the network environment based on malware interaction. The network reconfiguration can dynamically redirect or forward traffic to local server with associated services, *e.g.*, DHCP servers, SMTP servers, shared file systems, SCADA systems.
- The network layer customization can be automated. Using network profiles it is possible to execute the malware sample in multiple network profiles environments (*e.g.*, IPv6 only, industrial network, academic network).
- The assurance of containment rules using fine-grain control policies. Our solution provides a complete control over inbound and outbound per-flow communication.

The remainder of this paper is organized as follows. In Section II, we review related work. In Section III, we present our malware analysis system architecture. In Section IV, we describe how the system was evaluated and, in Section V we present the results. Section VI concludes this paper presenting final remarks and future directions.

978-1-5090-0679-3/16/$31.00 ©2016 IEEE

II. Related Work

Malware analysis has been extensively studied in the literature. Two main approaches are used to observe the behavior of suspicious software: static analysis and dynamic analysis. In static analysis, the suspicious software (or just binary) is examined with the goal of extracting information from it, but without actually executing the binary file. In dynamic analysis, instead, the binary file is executed in a controlled environment and its behavior is monitored [4]. Each approach has pros and cons, being feasible to apply to any kind of malware. Because of issues related to scalability and reliability of analysis, the dynamic analysis approach is often considered more effective and prominent in the context of automated malware analysis [5].

A typical dynamic malware analysis solution is composed of a sandbox (in which a suspicious binary file runs) and network middleboxes that provide security and network isolation. The research community has presented a number of studies on sandboxes, taking into account monitoring mechanisms and analysis transparency [10–12]. Usually, a sandbox is based on a modified virtual machine (VM) or machine emulator prepared to monitor system calls and network access. Some of these sandboxes use in-guest techniques to monitor system calls, as it is the case of Cuckoo [12] and CWSandbox [10]. Anubis Sandbox [13], in turn, is based on machine emulation using Qemu [14] to map system calls. In another front, Kirat [3] and Dinaburg [7], proposed a solution to monitor malwares' behavior using hardware virtualization technologies.

In fact, monitoring malware behavior using sandboxes is a convenient way to study the malicious actions against a target system. As a consequence, many malware coders ended up creating new techniques to evade potential malware monitoring systems. Before carrying out with malicious actions, many malwares perform reconnaissance by fingerprinting the guest system, in order to both improve the efficient of an attack and avoid detection. Malwares such as Agobot [15] can change the execution behavior when they detect an analysis systems. Other malwares simply terminate their execution when they detect a virtualized machine [16,17,6]. In addition, a subset of malwares checks for users activities and system configuration to try to detect monitoring systems [4].

Great attention has been given to specific approaches used to detect sandboxes and evade monitoring systems [3, 7, 6, 4]. Fingerprinting a sandbox is not the only approach used by malwares to avoid disclosing their behavior. Advanced malicious software explores characteristics of the whole execution environment that surrounds a sandbox, including the network profiles. These network profiles encompass Internet connectivity, local network connectivity, vulnerable services, and specific vendor devices. To fight against these malwares that outline network characteristics to tune their behavior, it is necessary to provide tools that allow an investigator to create different network profiles in order to identify malwares behavior deviation. Unfortunately, as previously discussed, the majority of researches targets sandbox transparency and information granularity. One exception is presented by Chen [16], where the author observes network activity to identify a remote virtualized system and builds a taxonomy for malware evasion. Kreibich *et al.* [18], on the other hand, describe large scale malware analysis environment focusing in containment policies using customized route protocol. Our solution, in contrast, focus on dynamically reconfiguring the environment, providing network resources based on this network traffic.

To achieve the dynamic scaling of resource and per-flow containment policies, we propose the use of Software-Defined Networking (SDN) technologies. The use of SDN to identify malicious software is not new. Jin and Wang [19], for example, describe a solution to detect mobile malwares using an SDN controller. Such a controller stores pre-defined malwares signatures, and acts like an Intrusion Detection System (IDS) to detect suspicious network flows. In contrast, we implemented a solution where SDN is used to reconfigure the network environment where sandboxes are located, including containment policies.

III. Architecture Overview

In this section, we describe MARS, our architecture for dynamic malware analysis. Our goal is to provide a flexible network layer to compose distinct and dynamic analysis environments that surrounds the sandbox. To achieve that, we have defined a modular event-driven architecture. Each event is associated with actions that directly interact with malware analysis by configuring the network and/or controlling the execution flow. An architecture overview is depicted in Figure 1 and the main elements are:

- *Sandbox* - It is responsible for executing the malware and tracking down its associated behavior. By design, and to simplify our observations, the sandbox executes a single malware sample per analysis. We do not specify any sandbox technologies; it can be a virtual machine or a plain physical system. As a requirement, the sandbox must provide an interface to remotely control the malware analysis execution.
- *SDN controller* - It is responsible for controlling the malware analysis and to manage the network elements. By having a centralized view, the controller can coordinate the malware analysis execution using software modules implemented on top of the controller.
- *Resource pool* - It is composed of network services and devices used to set an analysis scenario where the malware is executed. These elements may be inserted on the analysis environment to stimulate new behaviors or to collect detailed information about a specific service (traffic redirection).

In the next sub-sections we present in details the modules implemented on top of the SDN controller.

A. Inspection

This module is responsible for analyzing the network traffic looking for pre-defined patterns. These patterns could be a particular byte sequence or malware's family characteristics (i.e.

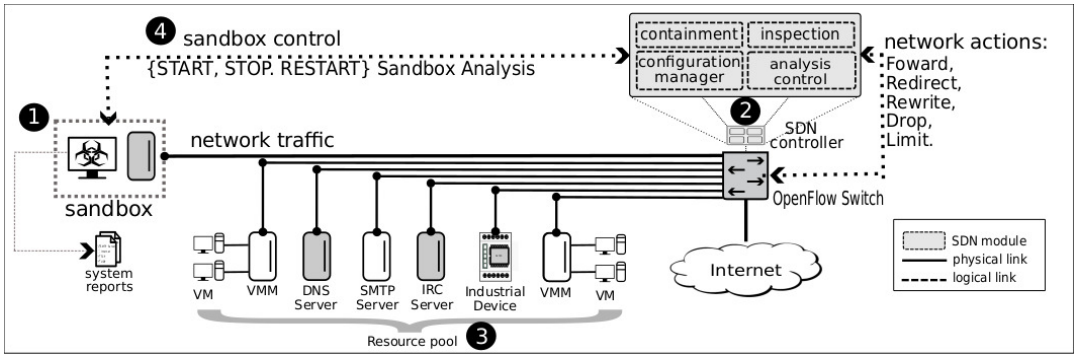

Fig. 1. **Malware analysis's flow**: The sandbox ❶ executes the malwares and performs connections to the Internet and/or to local network elements. All network traffic is inspected ❷ generating events to the SDN modules implemented on the SDN Controller. The controller, based on these events, modifies the network topology by forwarding or redirecting network flows to internal device ❸. Every connection can be managed (packet-rate, throughput, dropped) by the controller to enforce pre-defined containment policies. If required, the controller can interact with the sandbox and restart the analysis in order to investigate new network behavior ❹. If new traffic behavior occurred by changes in the environment, the analysis can be repeated until it achieves the stop condition (*i.e.*, number of rounds, specific behavior found, network threshold).

regular expressions, ports, IP address). The SDN controller is able to observe all communications – internal and external – and to examine full stack packets details. Once traffic that matches a pre-defined patterns is found, an event is issued and the appropriate module is responsible for handling that event is triggered. Events can be generated in response to network patterns that may take into account: number of packets, number of TCP/UDP sessions, packet payload, throughput, source/destination address, and TCP/UDP port.

In our proposed solution, we associate each event to an action that is translated into OpenFlow rules and incorporated into the SDN controller. The generated events are handled by one of the following three modules of the architecture, based on their functionality: *Containment module* for network policies, *Configuration Manager module* for topology configuration, *Analysis Control module* for sandbox interaction. For example, when a specific destination IP address is identified, one event can be triggered and the Containment module can block that connection in the whole environment.

B. Containment

The Containment module is responsible for applying containment rules related to network activity. The analyzed malware sample can communicate with other systems across the Internet as well as within local environment elements. Thus, it is appropriated to implements mechanisms to protect system elements in order to prevent the running malware from harming other systems too. Therefore, this module implements this protection in the core of the network. Using a centralized view, it is possible to provide a set of containment rules considering:

- *Throughput* - To determine characteristics of the affected systems, some malwares carry out tests. A common test is to download large files from the Internet and evaluate the connection parameters, such as the elapsed time. Systems connected to high-speed networks are more valuable for the attackers being used in malicious activities, such as spam propagation. Therefore, limiting the system throughput is a beneficial protection.
- *Packets rate* - Similarly to throughput, it is possible to configure the packet rate associated to each element in the architecture. This is useful in Denial of Service attacks (DoS): a class of malwares performs DoS using a high rate of small packet size.
- *Connection limit* - Other malwares perform connections at a higher rate than benign systems, especially in brute force and scanning attacks. In consequence, limiting the number of connections can be effective to avoid this kind of attacks.
- *Blacklist* - Sometimes, it is interesting to avoid access to few destinations from the analysis environment. In this way, malwares cannot attack the pre-defined destinations.

The containment rules are applied on the SDN controller affecting the architecture. Consequently, it is possible to define access policies to the whole analysis environment and for individual elements. For instance, individual elements like a Web Server can be limited to access a specific domain.

C. Configuration Manager

This module is responsible for handling events related to the network environment configuration. The essential actions consist in building a network topology and dynamically modifying it based on traffic characteristics. This configuration is highly customized enabling to insert new services into the analysis environment, implementing a different addressing policy, or reconfiguring the current network topology.

The analysis environment that surrounds the sandbox is instantiated before the analysis starts. For convenience, our solution provides different environment profiles, including: 'simple network with few Web Servers'; 'IPv6 only', 'industrial network', 'academic network', 'governmental institution'. Nevertheless, a custom environment profile could be instantiated by the user to create specific analysis scenarios. During the analysis the initial environment can be changed to incor-

porate new services or add more elements. An environment template example is presented bellow:

```
# initial analysis environment
web-server = {ip=10.0.0.1,ip=10.0.0.2,ip=10.0.0.3}
smtp-sever = {ip=10.0.0.4,status=waiting}
sandbox = {ip=10.0.0.10}
```

The above configuration sets 3 Web servers and one sandbox. The SMTP server is available but actually it is not part of the analysis environment in the initial stage. However, the SMTP server is in 'waiting mode' and could be incorporated into the environment depending on the malware's actions.

D. Analysis Control

In our architecture, the malware analysis process might be composed of multiple cycles which means executing a malware sample multiple times. Thus, this module is responsible for controlling theses cycles by handling events where the interaction with the sandbox is required. Such events are generated in the presence of network events and trigger actions that can: conclude the analysis, restart the malware execution, or revert the sandbox to a clean state for next reinfection. For example, when the analysis environment is changed – new services are added –, the controller can interact with the sandbox and restart the analysis to investigate new network behavior with the new element present. If new traffic behavior is presented because of changes in the environment, it is possible to repeat the process until achieving the stop condition (number of rounds, specific behavior found, network threshold).

IV. System Evaluation

To validate our architecture, we have implemented a prototype of our solution. We identify distinct malware network behaviors by changing the network environment that surrounds the sandbox. To achieve that, we have specified three network environment profiles. Although, our architecture enables dynamic topology configuration we choose to implement a fixed topology and diversify the access policy to carry out our evaluation. The three network profiles environment configured are:

- *open*: no restrictions, *i.e.*, all protocols in any direction (inbound or outbound) are allowed;
- *partial*: no restrictions on the local network, but the access is limited to elements out of the analysis environment (Internet). The allowed ports are: 80/TCP, 6667/TCP, 53/TCP, 53/UDP, 67/UDP, 68/UDP;
- *close*: only local network access is permitted, no Internet connectivity.

The malwares samples were executed in the three network profiles and all network access attempts were examined. The 50 malwares are classified as APT (Advanced Persistent Threat), publicly available at the VirusShare repository [1]. Every malware was instantiated in a Windows XP for 3 minutes and exposed to the three environment profiles, producing 150 analysis reports. Our analysis scenario is shown in Figure 2 and describe bellow:

- *POX Controller* - The controller used in our prototype is POX [20]. All implemented SDN modules was developed using the POX API and the OpenFlow 1.0 protocol.
- *OpenVSwitch* - An OpenFlow specialized [21] switch that interconnects all elements in the analysis environment.
- *Web Server* - The Web Servers are elements in the environment where the malware can access end exploit possible vulnerable Web application (Wordpress, for example).
- *Sandbox* - We used the Cuckoo Sandbox [12] with minor customization. The sandbox instantiates the malware in a Windows XP virtual machine.
- *Malware repository* - It is responsible for submitting each malware sample to the architecture.

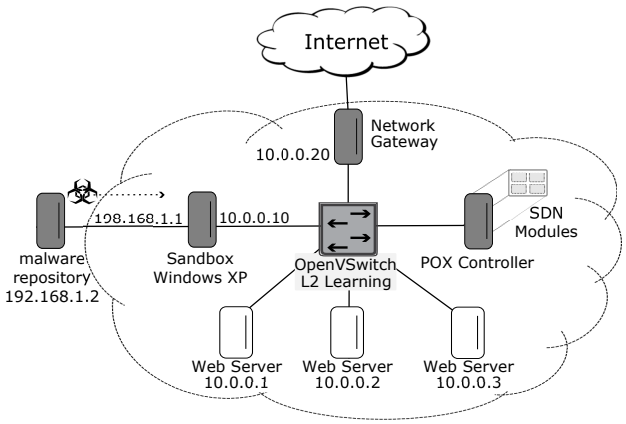

Fig. 2. System implementation overview.

In our evaluation methodology we focus the analysis on the network connections. We classified the network connections as *internal events* and *external events*. One event is defined as an unidirectional flow composed of timestamp, protocol, source IP, source port, destination IP, destination port, and the action applied by the controller (drop or accept). The following listing shows an example:

```
#time,proto,src-ip,src-port,dst-ip,dst-port,action
1444772807,udp,10.1.1.1,55858,20.X.34.1,53,accept
1444772811,tcp,10.1.1.1,1505,54.X.X.220,80,accept
1444772824,udp,20.X.X.1,53,10.1.1.1,55858,accept
1444772832,tcp,10.1.1.1,1506,20.X.X.6,80,accept
1444772843,tcp,10.1.1.1,1507,20.X.X.6,80,accept
1444772850,tcp,17.X.X.12,80,10.1.1.1,1504,accept
1444772855,tcp,54.X.X.22,80,10.1.1.1,1505,accept
```

Flows destined to external IPs are *external events* and flows destined to internal network are *internal events*. We are interested to map network behaviors available in the different network profiles, consequentially all repeated event in the same report were discarded to avoid inconsistency results. When the sandbox downloads files, for example, many flows associated to this specific behavior were observed. Thus, avoiding redundancy events is a feasible way to map distinct

[1] http://www.virusshare.com/

malicious behavior. In the next section we describe the findings and a detailed discussions.

V. Results

The set of analyzed malwares presented interesting behavior and corroborates with our hypothesis. Our first analysis consists in examining the total number of events – internal and external – in every network profiles. Results are summarized in the Table I, where we present the arithmetic mean and standard deviation of events. Notably, more network events are observed in 'open' and 'partial' profiles. The 'close' profile, in particular, shows a reduced in the number of events. The first two profiles are permitted to access the Internet with minor restrictions. In opposite, in the profile 'close', only local access is permitted. So, it is possible to infer that the behavior of the analyzed malwares is high dependent on the Internet access. The majority of malwares tries to access the Internet in order to download other malicious software or to associate to a remote administrative control. When this is not possible, malwares stayed dormant and did not try to explore the local network.

Network profile	Mean	Std Var.
open	38,94	25,52
partial	30,04	21,67
close	18,90	78,69

TABLE I
Network events by profile.

In Figure 3, Figure 4, Figure 5 we present the results from the 50 analyzed malwares in three graphs. Each one represents the network events in the respective network profiles: open (Figure 3), partial (Figure 4), close (Figure 5). In the X axis of each graph, we show the same 50 MD5 hashes from analyzed malwares. It is interesting to see the behavior deviation from particular malwares in the tree profiles. For example, the number of events presented by malware '002…b8e' is the same in partial and open profile, but in close profile the number of events is significantly low. This malware in particular uses port 80/TCP as the main vector of attack. So, the partial profile – where 80/TCP is allowed – does not affect the behavioral profile for this malware. However, the close profile where 80/TCP is not allowed, the number of events is low.

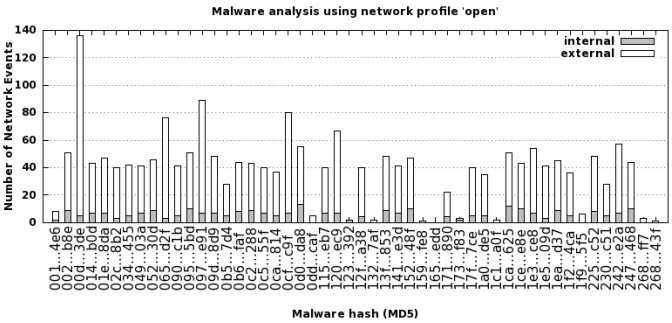

Fig. 3. Network events mapped using the network profile 'open' where restrictions are not imposed.

Malwares executed in the network profile 'partial' exposes a minor pattern deviation from the 'open' profile (Figure 4). Despite the traffic restriction, imposed by the network profile, we continue to observe the high number of external events (connections destined to the Internet). Results show that a set of malwares can adapt themselves trying to connect in other ports, ports that are not restricted by the system. One example is the sample '001…4e6', where it is possible to see more events in the 'partial' profile than in the 'open' profile.

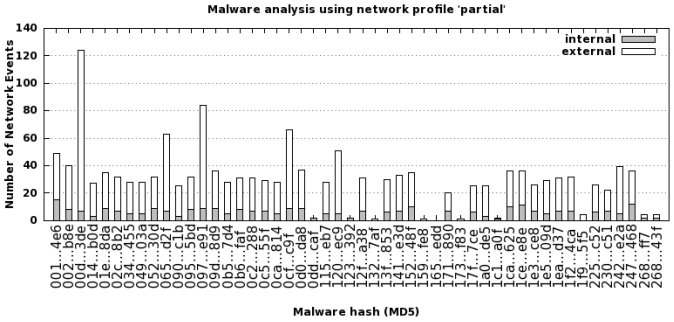

Fig. 4. Network events mapped using the network profile 'partial'. In this environment there are no restrictions on the local network, but Internet access is limited.

The behavior presented by the malware '159…fe8' is a typical behavior of malwares that probe the environment before execution. In this case, the network profiles were not decisive to trigger network events. This class of malwares may require specific systems (operational system, softwares, network environment) to enable its malicious behaviors.

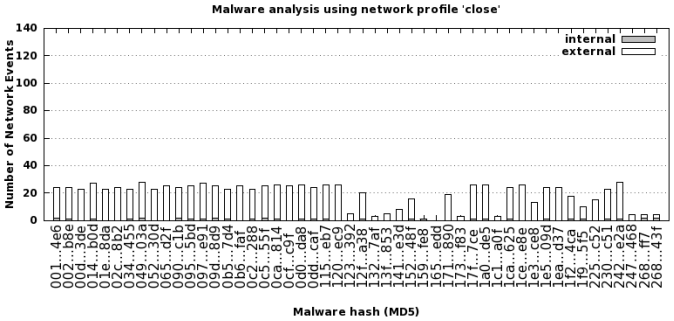

Fig. 5. Network events mapped using the network profile 'close' where only local network connections are permitted, no Internet connectivity.

In the 'close' scenario, we might expect that malwares with no connections will try many ways to access the Internet. However, as can be observed, the number of internal and external events is very low in comparison to other profiles. Again, this behavior supports the argument that without the specific network and system profile, malwares stay dormant without performing substantial network activity.

VI. Conclusions and Future Work

In this paper we presented MARS, a novel architecture for malware analyzing based on SDN technologies. The proposed architecture integrates dynamic topology configuration and per-flow containment polices. Using our solution, it is possible to execute a malware in distinct network environments with

their own containment rules in an automated-fashion. As discussed, the network environment where the malware is executed directly influences the malware behavior. So, using an integrated and automated solution contributes to improve the malware analysis process, dispensing traditional solutions based on middleboxes.

To prove the concept, we have analyzed 50 malwares in different network environments and we have identified behavioral deviations based on network properties. In our analyses, we implemented three different network profiles that lead us to the following conclusions:

- The network profile indeed affect the malware analysis and the quality of information collected in the sandbox. When the malware achieves the proper networks connections, more activity can be traced in the host system by the sandbox.
- In the analyzed set of malware, it is possible to identify few samples that tries to access different ports when one specific port is unreachable. The majority of malwares does not do that, but they try to access different destination IP address using the same port.
- Using the proposed architecture, it is possible to map behavior deviations in an automated way. Our analysis has identified particular malware actions in different network profiles.

We do have limitations on this architecture. As others SDN-based solutions, MARS is also vulnerable to threats and attacks targeting SDN models. If the controller is compromised the system can not properly execute the malware analysis. The elements that are used to compose the analysis scenario (resource pool), likewise can potentially be compromised by one malware executing in the environment. So, we should implement in the future mechanisms to ensure system integrity. On the other hand, the implemented prototype only materializes a set of functions proposed by our architecture. In that way, we argue that our solution can be effective to what we proposed, but given a set of characteristic that can influence our results. For examples, there is a class of malwares that try to connect in a dynamic IPs (P2P administration controller, fast-flux domains, CDNs) generating distinct network events in each analysis. So, this is not a limitation of our architecture, but it may introduce deviations in our results. To address this limitation, in the next step of research, we will improve the evaluation methodology considering 'system malware behavior' – signature [12] – besides the network connections. In addition, we plan to implement new features of the architecture using dynamic configuration to stimulate new malwares behaviors improving the quality of reports. By doing that, we expect to analyze more extensively the malware ecosystem including mobiles malwares.

VII. Acknowledgments

This work is partially funded by São Paulo Research Foundation (FAPESP) under grant #2013/15417-4.

References

[1] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *Security & Privacy, IEEE*, vol. 9, no. 3, pp. 49–51, 2011.

[2] C. dos Santos, R. Bezerra, J. Ceron, L. Granville, and L. Tarouco, "Botnet master detection using a mashup-based approach," in *Network and Service Management (CNSM), International Conference on*, Oct 2010, pp. 390–393.

[3] D. Kirat, G. Vigna, and C. Kruegel, "Barecloud: bare-metal analysis-based evasive malware detection," in *Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14). USENIX Association, Berkeley, CA, USA*, 2014, pp. 287–301.

[4] D. Fleck, A. G. Tokhtabayev, A. Alarif, A. Stavrou, and T. Nykodym, "Pytrigger: A system to trigger & extract user-activated malware behavior," in *2013 International Conference on Availability, Reliability and Security, ARES 2013, Regensburg, Germany, September 2-6, 2013*, 2013, pp. 92–101.

[5] A. Moser, C. Kruegel, and E. Kirda, "Exploring multiple execution paths for malware analysis," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 231–245.

[6] T. Holz and F. Raynal, "Detecting honeypots and other suspicious environments," in *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. IEEE, 2005, pp. 29–36.

[7] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: malware analysis via hardware virtualization extensions," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 51–62.

[8] S. Sezer, S. Scott-Hayward, P.-K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 36–43, 2013.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[10] C. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using cwsandbox," *IEEE Security & Privacy*, no. 2, pp. 32–39, 2007.

[11] T. Blasing, L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak, "An android application sandbox system for suspicious software detection," in *Malicious and unwanted software (MALWARE), 2010 5th international conference on*. IEEE, 2010, pp. 55–62.

[12] D. Oktavianto and I. Muhardianto, *Cuckoo Malware Analysis*. Packt Publishing Ltd, 2013.

[13] C. Kruegel, E. Kirda, and U. Bayer, "Ttanalyze: A tool for analyzing malware," in *Proceedings of the 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference*, vol. 4, 2006.

[14] F. Bellard, "Qemu, a fast and portable dynamic translator." in *USENIX Annual Technical Conference, FREENIX Track*, 2005, pp. 41–46.

[15] L. McLaughlin, "Bot software spreads, causes new worries," *Distributed Systems Online, IEEE*, vol. 5, no. 6, p. 1, 2004.

[16] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario, "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware," in *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*. IEEE, 2008, pp. 177–186.

[17] T. Garfinkel, M. Rosenblum *et al.*, "A virtual machine introspection based architecture for intrusion detection." in *NDSS*, vol. 3, 2003, pp. 191–206.

[18] C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson, "Gq: Practical containment for measuring modern malware systems," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 397–412.

[19] R. Jin and B. Wang, "Malware detection for mobile devices using software-defined networking," in *Proceedings of The 2014 IAJC/ISAM Joint International Conference*, ser. IAJC/ISAM '14. Orlando, FL, USA: IEEE Computer Society, 2014.

[20] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of sdn/openflow controllers," in *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*. ACM, 2013, p. 1.

[21] B. Pfaff, "Open vswitch manual," *Manual, Open vSwitch, [Accessed 10 October, 2015]*.