GQ: Practical Containment for Measuring Modern Malware Systems

Jeff Malavasi
Dept. of Computing Security
Rochester Institute of Technology
Email: jm3378@rit.edu

I. Review

The authors of the paper aim to develop a framework that can be applied to dynamic malware analysis environments in order to safely contain samples. While the field has proven that there is a need to execute malware in order to properly reverse engineer it, little progress has been made in defining the proper controls needed to do this without risking harm to other machines or users. The authors propose a containment architecture that allows researchers to start with a deny-all environment. Next, they show how to methodically apply policies that allow specific communications while ensuring that there is no potential for leakage. Finally, the authors demonstrate how building a containment profile can actually improve our understanding of the binary being studied [1].

When designing their system, the authors defined specific goals that would be necessary for successful containment. For example, containment policies would need to be flexible enough to allow the minimum amount of traffic to flow, while also convincing the binary that it was not in a sandbox. To achieve this, each device was placed on a separate VLAN, and all traffic flowed through a centralized containment server and router [1]. These machines are responsible for analyzing all incoming traffic and then deciding where and how to move the traffic to a destination in the containment network. Traffic can be manipulated in a number of ways such as rate-limiting, rewriting or even dropping specific packets. Typically, traffic would be sent to an inmate server, which would be infected with the malware of interest.

The authors utilize a shimming protocol in order bind the router and containment server which allows them to inject metadata into the packets that is later used to apply policies. In order to define the policies for a specific binary, a configuration file is used. This file defines which inmates will receive a policy, which binaries they will be infected with, lifecycle actions and a definition of the containment environment. Finally, they implement a reporting service that can be used to verify that the gateway and server are correctly enforcing policies and fine tune the current profile.

After creating a containment profile for a binary, the researchers found that the system could be used for analysis of malware as well. For example, unknown samples can be tested against a known profile to determine if it is a member of the same phylogeny. Additionally, they found that by varying the specifications of the environment and policy set applied to a sample, novel behaviors could be observed [1].

The researchers were able to successfully design a containment system that can easily be adjusted to many different types of malware samples. Through the use of robust reporting, the authors were able to verify the success of their design in real time which helped to accelerate the development of a specific containment profile. However, there is still additional work that is needed to be done. For example, the authors noted that the system struggled with scalability. This was both due to logical limits, such as VLAN exhaustion, as well as availability issues due to the fact that all traffic must flow throw the containment server and router. This was especially true as the number of inmates increased in the environment. In the future, the researchers may want to consider adding multiple containment servers behind a load balancer to help reduce bottlenecks.

While it is extremely important to dynamically study malware, it is important that researchers only do this in heavily contained environments. The authors highlight the fact that there is no standardized framework for designing a contained environment and introduce an application that can be used to quickly and automatically spin up a robust sandbox. Future research will still be needed to create containment profiles for novel malware families and additionally improve the scalability of GQ.

References

[1] C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson, "Gq: Practical containment for measuring modern malware systems." New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2068816.2068854