The EXPMON System - Advanced Detection and Analytics for File-based Exploits
- Detecting zero-day/unknown exploits through designing the system right

Methodology and Architecture

Introduction

EXPMON (exploit monitor) is a sophisticated system that analyzes file-based samples in depth. It uses crafted sandboxing and static analysis techniques to check if a file sample could be a potential exploit. Essentially, it analyzes and saves metadata in an approach we call "environment binding". This allows us to perform future advanced "Big Data" analytics to find undetected exploits and continuously improve the system.

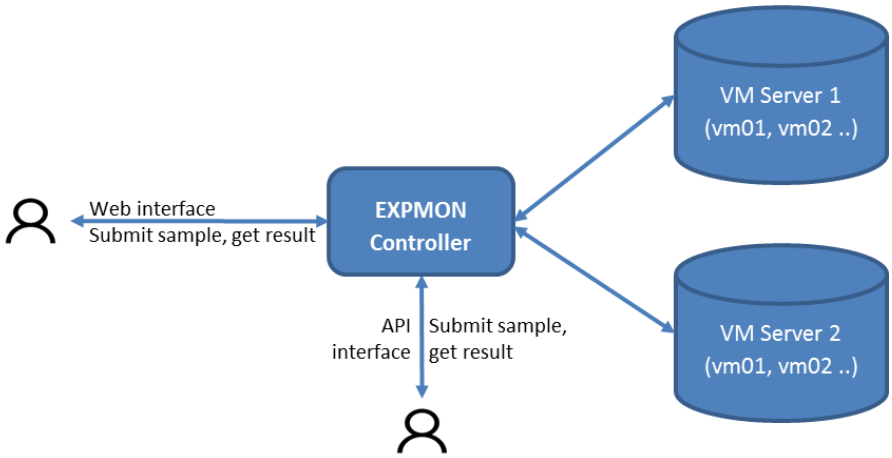
The core idea of EXPMON is that it runs the sandboxes based on a concept we call "environment binding." This is because exploits run very differently than malware. Exploits are highly dependent on the specific software environment; we can't examine exploits without considering the environment. For example, a .pdf exploit for Foxit Reader may not function as a working exploit on Adobe Reader. If you use a sandboxing environment with Adobe Reader to test the .pdf sample, you will miss out on the exploit. Sometimes, an exploit may target specific software versions and act normally on others.

EXPMON is designed and built from the perspective of vulnerability research, in contrast to others that are all built based on malware research. This, in fact, constitutes the core difference between this system and other sandbox-based systems. There is a significant need in the security community for a pure, crafted system designed solely for advanced exploit detection, and that's exactly the reason why we built this system.

The Architecture

We want EXPMON to be a system that is as open as possible, allowing it to benefit the security community the most. This is why we are sharing the architecture of the system, enabling users to better understand our approach and leverage the system in their daily operations to combat advanced exploits.

The high-level architecture is illustrated in the following figure:



As we can see, at a high level, the system appears quite simple (although in detail, it is a highly sophisticated system). It comprises the Controller and the VM Server(s). The controller serves as the core component, it does many jobs including calling the VMs in the VM Servers for sandbox-based analysis.

EXPMON system can only have one Controller but may feature one or multiple VM Servers. This depends on the required "analysis speed" and the computing power the VM Servers can provide. For instance, if you possess only one VM Server capable of hosting 8 VMs, but you have a substantial number of samples exceeding the capacity of the 8 VMs, adding another VM Server becomes necessary to augment the available computing power. The VM Server(s) is basically a pool of VMs.

Following the data flow, the Controller executes the following steps:

1. Accept sample submissions.
2. Perform static analysis.
3. Find the correct VMs (in the VM Servers) and perform sandbox-based analysis.
4. Collect and analyze the sandbox analysis data.
5. Output analysis result to user.
6. Save "file object" samples and raw/meta sandbox analysis data for future analytics.

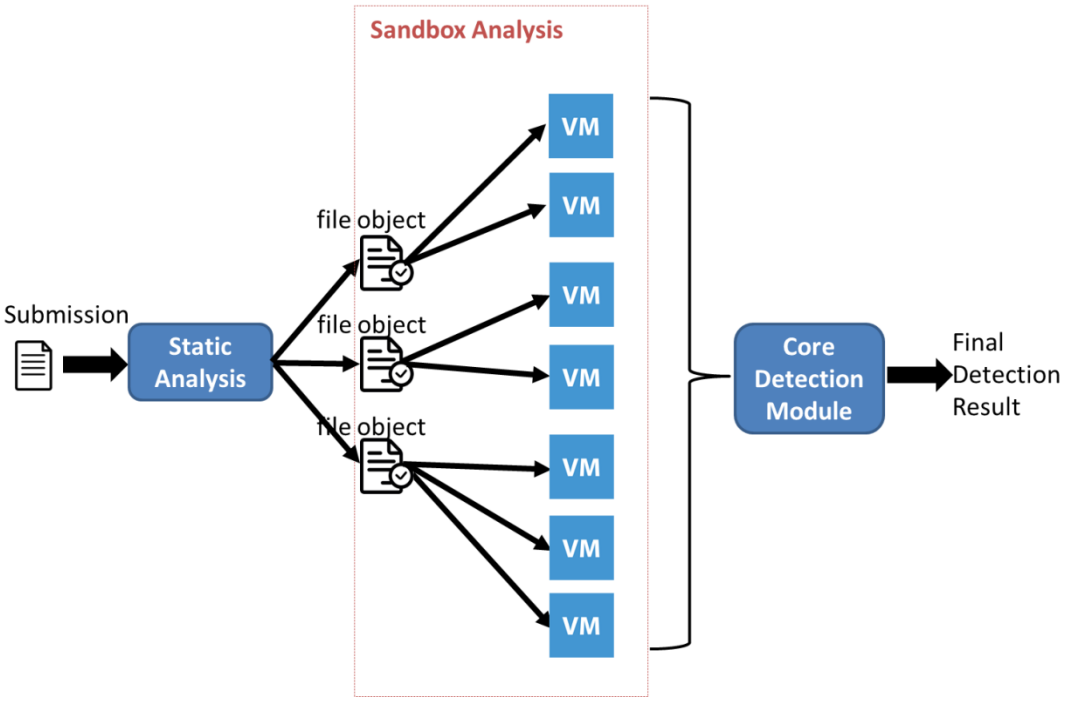
The above is still a high-level description of the Controller. The detailed process is as follows:

1. Analyze the submitted sample with the static analysis module. This may produce one or several "file objects" as a result. Each file object is tagged with meaningful metadata to assist the next sandbox analysis, ensuring more precise execution.
2. For every file object, based on the Controller's configuration, the "scheduler" module locates and runs the correct VMs for sandbox-based analysis. One file object may be sent to more than one VM for analysis.
3. Invoke the "detection module" to analyze the sandbox analysis data. The detection module is our core logic for exploit detection, utilizing a rule-based, environment-binding, and "plugin-

like" approach. Maintained by EXPMON, it undergoes periodic updates. We employ Big Data analytics to enhance the detection module and uncover the most elusive zero-day exploits. The subsequent step enables us to conduct Big Data analytics.

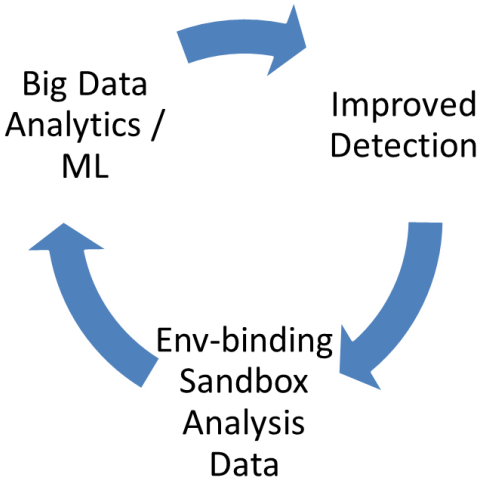
4. After reporting the analysis results to the user, the Controller saves the raw/meta sandbox analysis data, tagged with environment-binding information. This facilitates meaningful Big Data analytics to continuously enhance our detection logic and identify any oversights.

The entire process could be illustrated by the following figure:



For instance, a single submitted sample may be expanded into M file objects, and each file object is dispatched to N VMs for dynamic sandboxing analysis. The final detection result is determined based on all the file objects tested across all VMs. Consequently, the total number of running VMs for a single sample is M*N. This extensive approach ensures a more thorough and in-depth analysis compared to other sandboxing systems.

Crucially, this is a "detect-analytics-feedback-improve" cycle. Thanks to our environment-binding approach, we can conduct meaningful Big Data analytics, enabling continuous learning from what we may have missed and identifying areas for improvement. As we analyze more samples, the system becomes more precise and advanced, surpassing even the most undetectable file-based zero-days.



For example, when testing the system in August 2021, the system was able to detect the CVE-2021-40444 zero-day exploit with using just a few hundred samples from MalShare (acknowledged by Microsoft for the discovery).