halcyon

Halcyon Vs. EPP/EDR
The Future of Resilience

An endpoint protection platform (EPP) or endpoint detection and response (EDR) product are both critical control points in an organization. Understanding the depth and breadth of examination with these tools was a critical aspect when we began designing Halcyon. As we examine these controls, we will look at the architectural differences, some potential overlaps, and finally, the compliments halcyon has employed to create a better together environment for our client base.

Phases of Analysis

EPPs & EDRs provide two primary inspection layers in their product. They inspect files with a static analysis (pre-execution) engine and a dynamic analysis (behavioral) engine. The static analysis engine provides an inspection that looks to stop threats prior to or at the time of execution and uses cloud lookups to determine a basic attribution score based on the sample. The way to describe these engines are primarily as a signature engine of threats that were able to evade or bypass detections that have since been found, hashed, and placed in a lookup table upon execution of a binary.

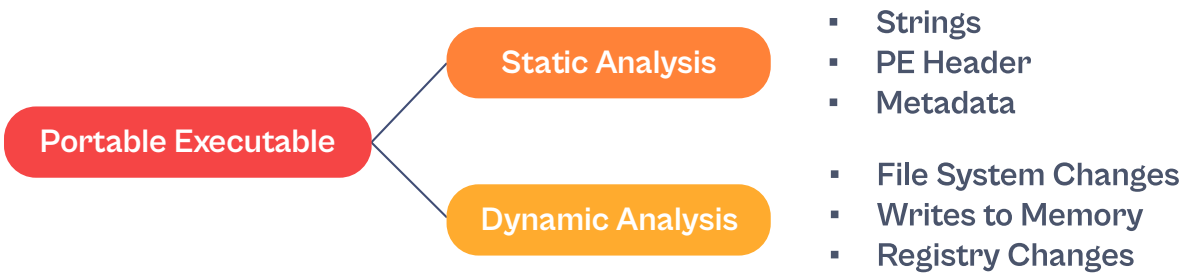

Fig 1: Examples of Analysis Types

They also look at common attributes of the sample to create a general conviction score of that sample. Dynamic analysis engines inspect behaviors using ML. The models most companies employ are based on the concept of a Convolutional Neural Network (CNN) where they train on all known attributes of all known malware profiles using an immense corpus of data to have their model return high-fidelity results. Most train the models once a year as it is expensive and time consuming. These behavioral models analyze the samples firing in real-time to determine if they match the pattern associated with the corpus of data they trained against.

THE CHALLENGE

While the telemetry to these tools is incredible the challenge in the dual inspection engine comes down to one primary challenge. **Loss of context upon inspection is one of the largest limiters for EPPs and EDRs.** The first inspection occurs, the threat is scanned and scored, and if it is under a designated threshold the threat moves on to start back at zero and is inspected in the behavioral engine. If the inspection threshold is 80 and a threat scores 79, you lose the attribution as it flows to the next inspection, despite that sample looking mostly 'bad'.

THE HALCYON ANSWER

We overcame the context challenge by weaving together an inspection fabric where the sample must be tested against four unique inspection gates, and each inspection carries the context of all former inspections before it. This allows us to examine a sample in a much deeper manner than most tools can. Based on the inspection chain we can build context through the attribution model and infer intent with higher fidelity.

DUAL DRIVER ARCHITECTURE

EPP/EDR is built on a dual driver package consisting of an application driver and a kernel driver. They register their driver in a callback table and when specific calls of interest are made, they receive the information that the access to the callback tables allows them. They then take that information and pass it to the application tier where they enrich the data. In the application tier of the OS these tools hook many common DLLs that provide detailed telemetry that allows them to provide analysis of IOCs and create attribution that can help a client detect and respond to patterns that deviate from norms their models are trained on.

THE CHALLENGE

Bypasses and evasions have become routine amongst these advanced threat actors. Amongst these techniques are libraries of application tier bypasses and evasions where threat actors look to actively use exploitation to blind the endpoint protection. Userland unhooking is a common technique employed today. Amongst more sophisticated threat actors, they look to manipulate kernel cal back tables where they can alter a vendor's registration on that table so they see no data from this tier. While we are calling out specific examples, the list of EPP/EDR architecture manipulation to bypass or evade their detection is vast.

THE HALCYON ANSWER

We have developed a primary single driver architecture where we set our hook fully into the kernel. We do not register or hook callback tables, so the common techniques used to bypass or evade those other controls cannot be used against us. We developed a patchguard-compliant driver that allows us to persist and see full information into the kernel directly. This allows us to help protect EPPs and EDRs as we can monitor for bypass and evasion techniques that otherwise go unseen and keep those tools on. If those tools miss, we compliment their detections with our own and will microfilter the environment so that if a sample skirts by your primary agent we should be able to, through attribution, convict that sample so it cannot continue to run. Our architecture helps to armor your current environment, complement your primary agents' detections and thwart the common bypasses and evasions current architectures are susceptible to.

DETECTION LOGIC

Primary EPP/EDRs are based on detection logic associated with all families of malware. They are looking at malware samples for common behaviors and attributes that will match patterns associated with that threat in their ML models and automatically block it. The logic behind these tools as referenced above is based on two engines and three primary inspections. Those inspections first lookup hash tables for known bad malware samples, conduct initial attribution scoring of the sample, and finally use their ML models to inspect the samples in a more granular fashion against known malware behaviors. It is important to note these logic engines look primarily at the behaviors of malware.

THE CHALLENGE

Most of the destructive threat actors in the wild today are moving away from deploying their weapons using malware. We have experienced a surge in exploitation using stolen, harvested, or purchased credentials and access. Once in they continue to use exploitation techniques that do not trigger primary endpoint detection engines such as DLL sideloading and common user behavior to build persistence in the environment. Standard endpoint tools, even the most sophisticated, are blind to many of the exploitation techniques employed based on how their models are built to detect. The other challenge comes in looking at the vast array of malware profiles in the wild. This can be an encumbering task based on the volume and baseline changes that are constant in the threat landscape. You routinely find daily samples that circumvent these tools as protecting against all things all the time is an onerous task.

THE HALCYON ANSWER

Having come from an exploitation background our primary premise was to work on building detection logic that was complimentary and plugged the exploitation gaps that current tooling has. We did this in several ways. The first was to build a contextually aware detection fabric that looked to aggravate the common conditions many of these threats employ to remain undetected. We built an exploitation engine primarily geared toward exploitation of the conditional statements advanced malware and operators work under such as geographic bailouts, anti-analysis checking, conditional system/software baiting and many others.

We were now for the first time able to employ models that would exploit the exploits that threat actors use forcing them to expose unkowingly themselves. We also built our approach on a concept of micro-modeling that allows us to update our ML logic multiple times a day and create client-specific models vs. the one-size approach that others take. That allows us to stay ahead of baseline changes in real-time vs. once a year. The final item that was critical in our compliments was the general notion that our logic will fail, and when it does what happens next?

We focused on resilience and have worked to build that concept into multiple layers of the security fabric as oppossed to simply trying to do more. Our goal with this was to plug the common gaps in current tooling and focus on reducing impact as close to zero as possible so in the event of a client being hit, we could rapidly recover them with much lower downtime than normally associated with rollbacks or backups. We provided these compliments and recovery solutions in the following manner:

DEFENSE RESILIENCE

- **Inoculation:** To combat propagation and re-infection from threats like ransomware we designed inoculation technology that can profile the sample and provide preventative protection against that attack and any mutations that match its behavior.
- **Bypass and Evasion Protection:** With the libraries of evasion techniques and bypasses being employed against critical controls such as EDR, we built protections for those controls so that attackers cannot blind or turn those controls off.
- **Micro-filtering:** If sophisticated threats make it past current control frameworks we act as a microfilter inspecting low level processes in memory to prevent infection.

OPERATIONAL RESILIENCE

- **Key Capture & Automated Decryption:** We have built a proprietary key capture & decryption technology that allows us to recover one or many systems in an environment in rapid fashion. It is the fastest form of recovery available from encryption-based attacks.
- **VSS Protection:** We also protect and can restore the machine based on VSS snapshots as a secondary form of recovery.
- **Real-time Resilience Module:** We are currently working on a solution to be able to rewind the system and registry changes a persistent threat will make to machines in order to backdoor them. This will give us the capability to undo these changes in real time.

DATA RESILIENCE

- **Data Exfiltration:** With extortion and data exfiltration skyrocketing we began looking for ways to make data more resilient. We correlate kernel level process data for suspicion with network and data behavior to reduce exfiltration of any given suspicious process to various volumetric thresholds and alarms meaning a bad actor may get the first few files but can no longer mass exfiltrate files.

HALCYON PRODUCT DEEP DIVE

PRE-EXECUTION BLOCKING

Coupled with the existing EPP, the Halcyon agent is the last line of proactive defense against ransomware (and other malicious/evasive code) from successfully completing its routines or functions on the system. The Pre-Execution Blocking acts as the application/executable/binary is first called by the system to be loaded into memory and before the first lines of code are run.

The Pre-Execution layer includes the Attribution engine and first implementation of our ML logic. The Attribution leverages several components to analyze the process prior to it running:

- Commercial threat feeds as well as proprietary ones.
- PE static analysis.
- Hard drive scanning.

Functions to:

- Determine ‘known bad’ to kill the process prior to execution.
- Determine ‘known good’ to allow processes to continue without intervention.
- Determine level of suspiciousness or trust weighting that carries throughout the process lifecycle and drives the subsequent analysis, monitoring, and antagonism by the Halcyon agent.

EXPLOITATION

Suspicious processes that are allowed to continue to execute are then passed to the Exploitation layer in which the Halcyon agent leverages a variety of techniques to take advantage of the conditional statements that are commonly written into ransomware code to perform functions like avoiding analysis (exploiting their anti-analysis or evasion techniques), avoiding compromising systems in their host countries (by presenting language files/keyboard/IP space/etc.), or avoiding compromising already encrypted systems (forced confliction through registry keys or other IOCs).

By injecting artifacts that make these conditions present within the view of the running/suspicious process, we can take advantage of routines already built into the ransomware code to prevent initial detonation.

By contrast, rather than trigger an exit of the process, the second engine in the Exploitation layer is designed to trigger as much of the bad behavior that the code is programmed to do as possible.

Behavioral engines are only able to detect the actions of a malicious process. In the case of ransomware, many of the behaviors will only be exhibited if certain conditions are present and when those conditions are not present, the routines simply will not run. Halcyon injects artifacts (false processes, bait files, etc.) into the view of the running process that entice as much bad behavior as possible.

Functions to:

- Attempt to get the process to trigger an exit and 'kill' itself through its anti-analysis and evasion routines that are built into the code.
- Attempt to elicit as much 'bad' or suspicious behavior as possible to amplify signal for not only the Halcyon's behavioral engine, but also the EPP that Halcyon is installed alongside

NEXT-GENERATION BEHAVIORAL

The Behavioral layer ties into the Exploitation engine in the layer before. The Next Generation Behavioral engine is where the bulk of the Halcyon micro-model ML capabilities are housed and is where we monitor and react to suspicious processes activities in real-time.

Most modern endpoint protection products leverage behavioral analysis, typically in the form of convolutional neural networks and decision tree algorithms.

These conventional machine learning capabilities still suffer from inherent flaws based on the size of the datasets required to train them and their ability to remain contextually aware.

The contemporary implementation of endpoint ML/AI models are unable to provide ongoing enhancements to the protection capabilities of system and can't incorporate new data previously learned by the endpoint. The Halcyon behavioral engine employs an industry-first micro-model architecture designed on the principle of capsule network-based machine learning that enables broad benefits over previous behavioral analysis methods.

NEXT-GENERATION BEHAVIORAL (CONT.)

- Allows for data-driven supervised and unsupervised learning engines that work together across endpoints and the
- Enables efficient analysis by using several models, each trained on specific feature sets, in parallel as opposed to one monolithic detection model.
- Provides highly accurate decisions with a limited dataset as each micro-model is specialized and assigned behavior sets that feed data into other micro-models to weigh in their decisions.
- Permits robust process tracing to detect and combat against process injection to further harden against attacks on the security products deployed on the system.

Exfiltration Module (GA July 2023)

- Provides examination of network and data behavior and cross correlates with endpoint process data to look at signaling on illegitimate data exfiltration in the enterprise. The goal is to look at behavioral attributes of the requestor, process and destination to stop exfil and return data leverage back to the client.

ISOLATION & RESILIENCY

For far too long the security products industry has targeted a goal of threat elimination – solutions that mitigate 100% of the threats 100% of the time. Halcyon was built with failure in mind and with the goal of reducing business impact as the first objective and threat mitigation second. Our experience over the last few decades has taught us that the threat is always evolving and while we must evolve and continually adapt to that threat – mitigating business impact is the first priority.

Ensuring that the business can continue to function is the first goal of the cybersecurity program and reducing the business impact of a ransomware event is the primary function of the Halcyon platform.

The Isolation and Resiliency layer provides protection when all other detection and prevention logic fails and provides the following functions:

- Protection of the Volume Shadow Service (VSS) from the MS kernel
- Control of the local system firewall rules to provide the ability to quarantine the affected system
- Symmetric encryption key recovery for encryption events
- Currently recovering/duplicating keys when ransomware is using native Microsoft AES encryption, which is commonly leveraged based on strength, speed and efficacy
- Will recover keys from imported libraries (such as salsa/chacha20) in release Q3 2023

Real Time Resilience Module (Release Fall 2023)

- Provides process specific rewind of all activities executed on the system in a function outside of the VSS. The ability to rewind system changes based on tracking that system's process tree provides the most robust and timely recovery in the market.