Detecting Zero-days with SnortML

Contents

Introduction

Intrusion detection/prevention systems (IDS/IPS) are the network security technology of choice for protection against exploits of known vulnerabilities. These systems rely on "signatures" to detect exploits in progress. However, the signatures may not detect exploits that utilize previously unknown variants of known vulnerabilities. We call these exploits "zero-day variants."

Network security researchers and developers have long investigated techniques that would enable IDS/IPS to detect the zero-day variants. Machine learning (ML) – a branch of Artificial Intelligence (AI) – has finally provided a mechanism to extend IDS/IPS detection to these zero-day variants.

SnortML is a machine learning-based exploit detection framework that can detect (classify) zero-day variants without requiring new signatures or classifier updates. SnortML is part of the Snort distribution and runs in parallel to the signature-based detection engines in Snort.[1]

IDS/IPS basics

Typically, IDS/IPS sits behind an access control engine. Whereas the access control engine blocks or permits network traffic based on layer 2-7 attributes of a traffic flow, IDS/IPS detects exploit attempts (attacks) within the traffic flows allowed by access control. IDS/IPS use protocol decoding engines and specific traffic flow characteristics (sometimes via regular expression pattern matching) to detect and block incoming attacks. The traffic characteristics used for identifying vulnerability exploitation are known as signatures and are the workhorses of IDS/IPS.

Snort is an open-source IDS/IPS implementation. It was initially developed in 1998 and has been available for free since then. Snort technology is also available as part of Cisco Secure Firewall, a commercial product.[2] Note that in the Snort world, signatures are called "Snort rules." This document will exclusively use the term "signature" to ensure consistency in the exposition.

Challenges with IDS/IPS signatures

IDS/IPS signatures are usually crafted by humans. After creation, they undergo an extensive testing cycle to build confidence in their efficacy. Depending on the signature provider, signatures may also be deployed in limited live settings before broader deployment. Again, the limited deployments aim to increase confidence in the signatures.

Signatures are generally written to tightly fit known exploitation of a known vulnerability. This is done deliberately to lower the probability of matching legitimate traffic, thereby keeping false positives low. The flip side of the endeavor to maintain a low false positive rate is that signatures may miss zero-day variants.

Thus, the human signature writer must manually tune the generalizability of a signature. If the signature is too tight, it will not catch even modest variations of a known attack, let alone zero-day variants. If the signature is too general, it will result in false positives. Finding the right balance is a tedious process that requires frequent trial and error. Frequently, false positives are enough of a concern that only tightly matching signatures are deployed.

Enter SnortML

SnortML provides an automated mechanism to find the right balance between generalizability and false positives. As mentioned above, SnortML uses machine learning – in particular, a deep neural network – to detect exploits.[3]

Machine learning techniques are an alternate way to "learn" the signature of a class of related exploits. Here, a deep neural network is trained on malicious and benign traffic corpora. The neural network infers generalized versions of the exploit patterns in the malicious corpora and learns to distinguish between malicious and benign traffic. For example, the malicious traffic may be an SQL injection exploit, whereas the benign traffic may be legitimate SQL queries.

Note that a neural network has many parameters that are tuned during the training process. Effectively, during training, generalized inferred patterns of attacks are embedded in the parameters of the neural network. These generalized patterns enable the neural network to detect zero-day variants. Continuing the SQL example above, a neural network can learn the pattern of related SQL injection exploits and detect a new exploit even if it has never seen it before.

SnortML has two components. The first is the machine learning engine, which loads machine learning classifiers (trained over malicious and benign traffic as discussed above) and makes them available for detection. The second is an inspector, which subscribes to data provided by the underlying Snort architecture, passes the data to classifiers, and then acts on the classifiers' output. The SnortML classifiers run in parallel to traditional signature-matching engines within Snort, as shown in Figure 1.

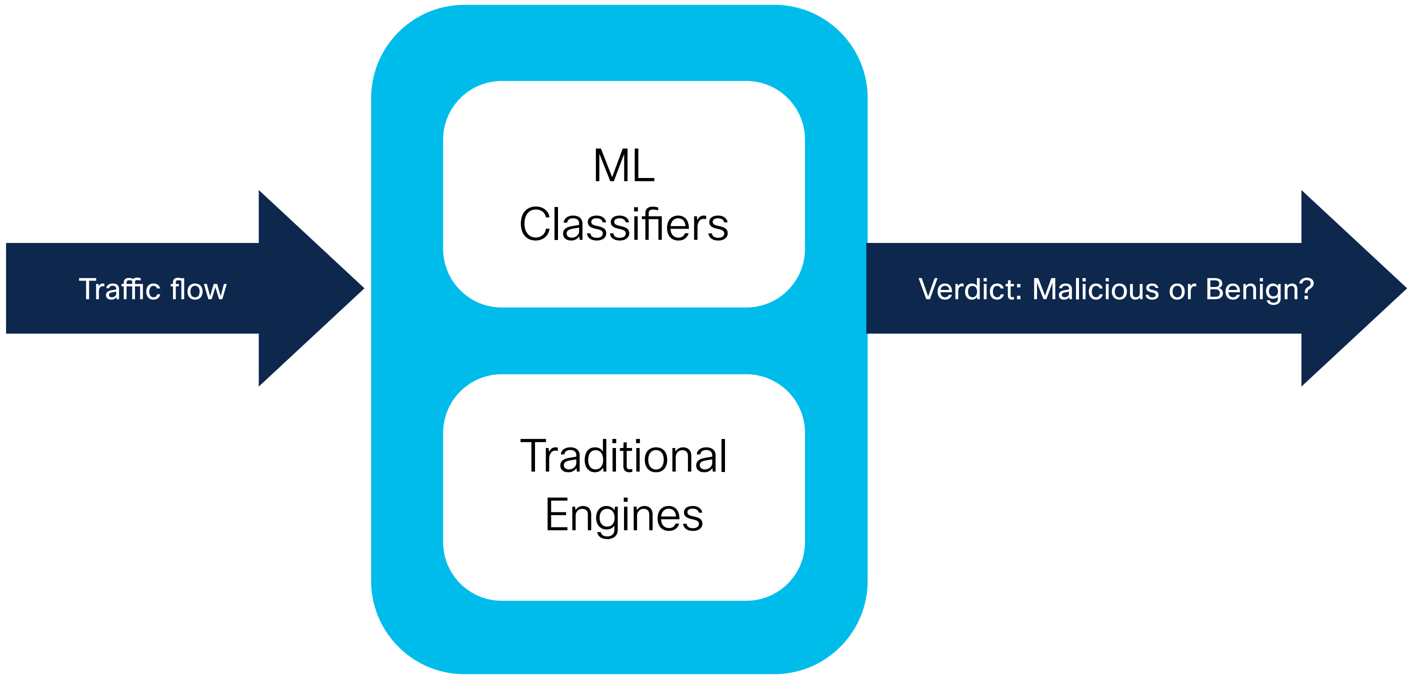

Figure 1: Machine learning classifiers and traditional signature-based detection in parallel.

The SnortML inspector obtains and passes HTTP data to a previously trained binary classifier. This classifier then returns the probability that it saw an exploit. Similar to traditional Snort, an alert can be generated based on the probability returned. In addition to alerting, SnortML can be configured to block malicious traffic.

SnortML performance

How well does SnortML work? How well does it work relative to Snort's venerable performance? To answer these questions, the developers behind SnortML carried out a battery of tests, which we discuss below.

First, the developers wanted to assure themselves about SnortML's low false positive rates over benign traffic. They selected three well-known datasets of URL queries that produce mostly benign HTTP traffic. These datasets are the Alexa 1000, Alexa 5000, and Common Crawl. Alexa 1000 is a list of the top 1000 visited websites worldwide.[4] Similarly, Alexa 5000 lists the top 5000 visited websites. Common Crawl is a repository of publicly accessible raw web pages and some related data.[5] Passing URLs from these datasets through SnortML results in high accuracy and very low false positives, as seen in Table 1.

Second, SnortML's developers wanted to ensure that malicious URL queries detected by Snort are also detected by SnortML. They used BreakingPoint – an industry-standard traffic generation tool – to demonstrate a high detection rate (see the last row of Table 1).[6] Note that Snort has a human-crafted signature for every BreakingPoint test and thus has a high accuracy rate for these tests.

Finally, the developers tested the latency – the time needed for the deep neural network to produce a verdict – of SnortML. They found that on a 64-bit AMD CPU running at 4.7 GHz, SnortML only needed 350 microseconds to complete its computation. CPUs on most Cisco Secure Firewalls are of higher capacity. As such, SnortML is expected to run even faster on these firewalls.

Source Data	# URL Queries	# False Positives	# False Negatives	# True Positives	# True Negatives	Accuracy (%)
Alexa 1000	14,148	52	0	0	14,096	99.63
Alexa 5000	279,271	35	0	0	279,236	99.99
Common Crawl	9,836,863	243	0	220	9,836,400	99.99
Breaking Point	540	0	1	539	0	99.81

Table 1: SnortML efficacy on benign and attack traffic.

An astute reader may ask, “What if a false positive was encountered in a real deployment?” Indeed, such a scenario is plausible. Typically, a security administrator will either have SnortML in alerting mode so that the false positive doesn’t interfere with live traffic or temporarily put SnortML in alerting mode. In the latter case, the security administrator would also submit a false positive report to Cisco for further analysis.

Similarly, a reader may ask, “What if a false negative gets through in a real deployment?” This is also a plausible situation. While SnortML will catch many more exploits against variants of known vulnerabilities than the equivalent Snort signatures, some exploits may indeed get through over time. Defenders in these situations will need to deploy additional security technologies that work substantially differently to detect such exploits. Example technologies include network sandboxing and endpoint detection and response.

Conclusion

Cisco has been working on IDS/IPS for over thirty years. In recent years, it has invested heavily in "Artificial Intelligence for Security."[7] Cisco uses AI to assist security administrators, augment human ability to detect incoming threats, and automate mundane and repetitive security tasks.

SnortML is an example of using AI techniques (machine learning) to augment human ability. Here, machine learning enables IDS/IPS to detect zero-day variants of exploits, significantly cutting down customers' exposure to new exploits. SnortML is also an example of automation. The SnortML framework enables the automated learning of signature patterns by a deep neural network, eliminating the need for the manual creation of traditional IDS/IPS signatures. As a result of the above-mentioned augmentation and automation, IDS/IPS continues to be a significant network security threat detection technology inside the Cisco Secure Firewall.

References

[1] Snort, retrieved June 25, 2024.

[2] Cisco Secure Firewall, retrieved June 25, 2024.

[3] Talos launching new machine learning-based exploit detection engine, Brandon Stultz, March 15, 2024.

[4] Alexa Top 1000 Most Visited Websites, retrieved June 25, 2024.

[5] Common Crawl, retrieved June 25, 2024.

[6] BreakingPoint, retrieved June 25, 2024.

[7] A New Age of AI-Enhanced Cybersecurity, retrieved June 25, 2024.