University of Pisa
Department of Computer Science

Data Science and Business Informatics

ICT Risk Assessement

Splunk Enterprise Evaluation

Experimental Analysis of Threat Detection Capabilities

Student: Melissa Abdelkafi - 702815
Instructors: Fabrizio Baiardi, Luca Deri

Academic Year: 2026/2025

Contents

1 Introduction

1.1 Project Context

The project started as an evaluation of Splunk Entreprise's technical capabilities in detecting threats as a security analytics platform. While the main point of this study was to test features using synthetic data, a blind analysis was conducted that revealed additional malicious activity that weren't in the provided alerts file. What started as a straightforward "Can Splunk Enterprise detect threats?" turned into a much more intriguing question: "Can Splunk Enterprise effectively identify additional threats that were missed?" This report documents my attempts to answer this question.

1.2 Objectives

- Ingest log datasets
- Real-time analytics via Search Processing Language (SPL)
- Test AI toolkit
- Leverage Splunk's BI-style dashboards
- Threat detection and risk scoring
- Evaluate performance

2 Methodology

2.1 Experimental Setup

- **Dataset:** WarmCookie Malware Traffic extracted from https://malware-traffic-analysis.net

Dataset	**Start Time**	**End Time**	**Duration**
Security Alerts	00:09:00.000	00:13:00.000	4 minutes
Network Traffic	00:09:00.000	00:41:42.204	32 minutes 42 seconds

Table 1: Analysis Time Windows

  – Datataset came as two files: WarmCookie malware data with a pcap traffic file and the Alerts text file.

- **Data Conversion:** The two files have been converted to CSV files so that they can get uploaded to Splunk (Pcap and txt files are not Splunk readable)
- **Data Cleaning:** The traffic CSV file had no Splunk-readable timestamp (it initially only included decimal seconds, the common format in packet analyzers)

- **Lookup Creation:** I have initially ingested the traffic data in different approaches, as a data source, as an index then eventually set it as a lookup to parse the log effectively.

2.2 Analysis Approach and Evaluation Metrics

The analysis was conducted using Splunk's Search Processing Language (SPL), a specialized query language for log analysis which allows to extract, transform, and analyze security data through commands. The analysis starts with understanding the environment, knowing how many packets the file has, and whether traffic patterns look normal. A blind analysis is combined with quantitative validation, comparing WarmCookie malware traffic with the alerts file to assess Splunk's capabilities across different evaluation metrics.

Table 2: Evaluation Metrics

Metric	**What It Measures**
Alert Coverage Rate	Threats caught by existing alerts (%)
Threat Discovery Rate	New threats found by Splunk (%)
Data Loss	Amount of data exfiltrated (MB)
Attacker Servers	Number of Attacker servers
Malware Delivery	Number of download attempts
Attack Infrastructure	Number of malicious external IPs
Behavioral Patterns	Types of attack communication identified

3 Experimental Evaluation

3.1 Network Discovery

3.1.1 Initial Network Reconnaissance

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| rex ",\"(?<Source>[^\"]+)\",\"(?<Destination>[^\"]+)\",
  \"(?<Protocol>[^\"]+)\",\"(?<Length>[^\"]+)\""
| eval src = replace(Source, "\"", "")
| eval dst = replace(Destination, "\"", "")
| eval proto = replace(Protocol, "\"", "")
| eval size = tonumber(replace(Length, "\"", ""))
| stats
    count as Total_Packets,
    dc(src) as Unique_Sources,
    dc(dst) as Unique_Destinations,
    avg(size) as Avg_Packet_Size
```

Listing 1: Network Baseline Analysis

This query basically loads the dataset, extracts and cleans key fields (source, destination, protocol, packet size), then calculates essential metrics: total packet count, unique com-

municating entities, and average packet size.
Note: While field cleaning here wasn't necessary for this spesific log dataset, I kept the cleaning syntax in the queries since most logs require cleaning in order to be properly parsed.

Results:

Metric	**Value**
Total Packets	18,189
Unique Sources	72
Unique Destinations	78
Average Packet Size	641.82 bytes

Table 3: Network Baseline Results

The results show 18,189 messages between 72 different devices communicating with 78 different destinations. Average message size is about 641.82 bytes.

3.1.2 Noisiest Devices Identification

The logic is basically identifying the noisiest devices because in security the noisiest ones are often the problem, since Malware-infected machines often send/receive way more traffic than normal.

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| rex ",\"(?<Source>[^\"]+)\",\""
| eval src = replace(Source, "\"", "")
| stats count by src
| sort -count
| head 10
```

Listing 2: Identification of the noisiest devices

The query counts packets per source IP and sorts in descending order. This reveals which hosts are generating the most traffic.
Results:

Table 4: Top 10 Network Communication Sources

Source IP	Packet Count	Percentage of Total
10.8.15.133	6,646	36.54%
104.21.55.70	2,013	11.07%
72.5.43.29	1,653	9.09%
23.220.103.8	1,142	6.28%
23.220.103.18	1,075	5.91%
10.8.15.4	1,047	5.76%
172.67.170.159	1,021	5.61%
23.220.103.72	496	2.73%
204.79.197.203	415	2.28%
20.42.72.131	222	1.22%

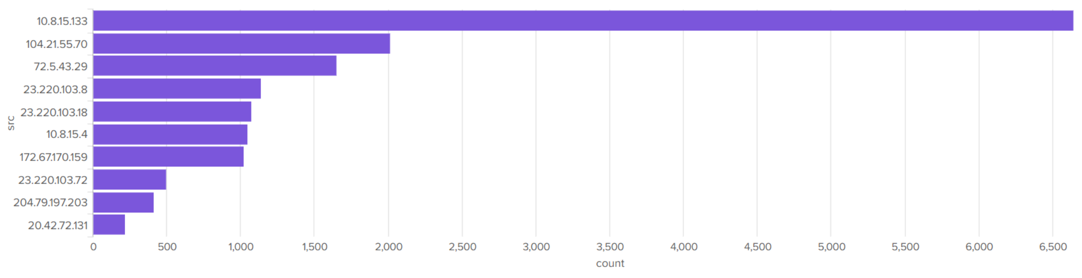


Figure 1: Top 10 Network Communication Sources Bar Plot

Finding the noisiest devices helps spot problems and here it shows one device (10.8.15.133) is doing 36% of all communication which is probably not normal.

3.1.3 External Communication Mapping

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| rex ",\"(?<Source>[^\"]+)\",\"(?<Destination>[^\"]+)\",\""
| eval src = replace(Source, "\"", "")
| eval dst = replace(Destination, "\"", "")
| where NOT dst LIKE "10.8.15.%"
| stats count by dst
| sort -count
```

Listing 3: External Communication Mapping

This query puts focus specifically on communications leaving the internal network (10.8.15.%) by filtering for external destinations, then counts connections to each external IP. This exposes potential command and control servers and malware distribution points.
Results:

Table 5: External Threats Mapping

External IP	Packets Count
72.5.43.29	1,844
23.220.103.18	469
23.220.103.8	389
104.21.55.70	262
204.79.197.203	215

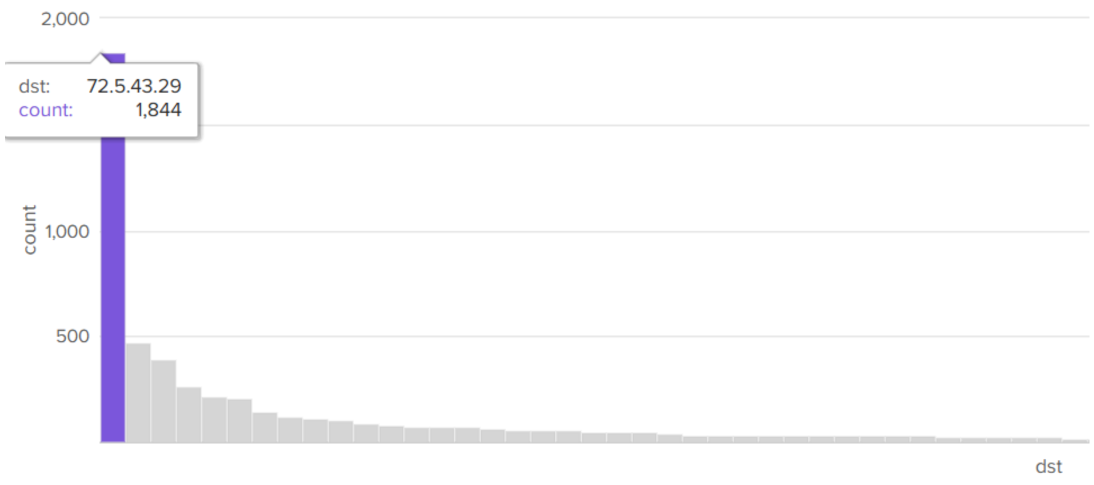


Figure 2: External Threats Mapping Bar Plot

Splunk found most external traffic goes to 72.5.43.29 (not an internal IP) with 1,844 packets, this is the main ”Attacker” server.

3.1.4 Malware Delivery Detection

Here we look if anyone is downloading suspicious files. Large downloads from strangers often mean malware and tracking them helps understand the infection method.

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| rex
  ",\"(?<Source>[^\"]+)\",\"(?<Destination>[^\"]+)\",\"(?<Length>
  [^\"]+)\""
| eval src = replace(Source, "\"", "")
| eval dst = replace(Destination, "\"", "")
| eval size = tonumber(replace(Length, "\"", ""))

| where dst="10.8.15.133" AND size > 1000 AND NOT src LIKE
  "10.8.15.%"
| stats
    count as Large_Download_Attempts,
    values(src) as Malicious_Sources,
    avg(size) as Average_Payload_Size
```

Listing 4: Malware Delivery Detection

The query detects suspicious download activity targeting the infected host (10.8.15.133) by identifying large payloads set above (1000 bytes) from external sources then counts download frequency, malicious sources, and average payload size.
Results:

Table 6: Malware Delivery Detection Results

Metric	Value
Large Download Attempts	6,275
Average Payload Size	1,430.69 bytes
Unique Malicious Sources	+60

The infected device received 6,275 suspicious packages from +60 different delivery services.

3.1.5 Attacker Servers Detection

Here is a basic check if the infected computer make regular calls with command servers. Regular calls mean active malware control. Finding this pattern confirms active compromise.

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| rex ",\"(?<Time>[^\"]+)\",\"(?<Source>
  [^\"]+)\",\"(?<Destination>[^\"]+)\",\""
| eval time_num = tonumber(replace(Time, "\"", ""))
| eval src = replace(Source, "\"", "")
| eval dst = replace(Destination, "\"", "")

| where src="10.8.15.133" AND (dst="72.5.43.29" OR dst LIKE
   "23.220.103.%")
| stats
    count as Attacker_checkins,
    dc(dst) as Attacker_Servers,
    avg(time_num) as Average_Interval
```

Listing 5: Command Servers Detection

This query identifies the communiation pattern between the infected host and known malicious servers to eventually deduce the command and control servers, it calculates the check-in frequency with these servers, and determines its average time interval.
Results:

Table 7: Command and Control Beaconing Analysis

Metric	Value
Attacker Check-ins	2,843
Attackers Servers	4
Average Check-in Interval	658.91 seconds (11 minutes)

The infected device is communicating 2,843 times to 4 different servers, checking in every 11 minutes on average.

3.1.6 Data Exfiltration Assessment

The question here is "Does the data get stolen?". Large outgoing transfers to unknown servers mean data theft which is in many cases the ultimate goal of many attacks "stealing data". Quantifying data loss helps understand the impact of the attack. The following query measures outbound traffic from the infected host "10.8.15.133" to external destinations, focusing on larger packets above (500 bytes) which likely contains stolen data. Then computes data loss volume, exfiltration frequency, and receiving servers.

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| rex ",\"(?<Source>[^\"]+)\",\"(?<Destination>[^\"]+)\",
  \"(?<Length>[^\"]+)\""
| eval src = replace(Source, "\"", "")
| eval dst = replace(Destination, "\"", "")
| eval size = tonumber(replace(Length, "\"", ""))
| where src="10.8.15.133" AND size > 500 AND NOT dst LIKE
  "10.8.15.%"
| stats
    sum(size) as Total_Bytes_Exfiltrated,
    count as Exfiltration_Packets,
    values(dst) as Receiving_Servers
```

Listing 6: Data Exfiltration Assessment

Results:

Table 8: Data Exfiltration Analysis Results

Metric	Value
Total Bytes Exfiltrated	1,142,564 bytes (1.14 MB)
Exfiltration Packets	1,039
Unique Receiving Servers	50+

Primary Exfiltration Destinations:
72.5.43.29, 104.21.55.70, 23.220.103.8, 23.220.103.18,
20.42.72.131, 52.109.0.136, 40.126.29.10, 23.205.110.12

1.14 MB of our data was stolen sent out in 1,039 separate pieces to 50+ different drop points.

3.1.7 Attack Timeline Reconstruction

This part shows when the attack happened, when it peaked, and how it evolved. That's crucial for incident response and helps understand attacker methodology.

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| rex
  ",\"(?<Time>[^\"]+)\",\"(?<Source>[^\"]+)\",\"(?<Destination>
 [^\"]+)\",\"(?<Length>[^\"]+)\""
| eval time_num = tonumber(replace(Time, "\"", ""))
| eval src = replace(Source, "\"", "")
| eval dst = replace(Destination, "\"", "")
| eval size = tonumber(replace(Length, "\"", ""))
| bin time_num span=0.1
| eval is_malicious = if(src="10.8.15.133" OR dst="10.8.15.133",
  1, 0)
| stats
    sum(is_malicious) as malicious_traffic,
    count as total_traffic
    by time_num
| sort time_num
```

Listing 7: Attack Timeline Reconstruction

This query creates a time-series visualization of malicious activity by grouping traffic into time buckets and marking communications involving the infected host.
Results:

Table 9: Attack Timeline Peak Analysis

Metric	**Value**
Peak Time Window	80.2 - 80.3 seconds
Peak Traffic Volume	946 packets
Attack Duration	4 minutes (00:09 - 00:13 UTC)

Peak activity happened at time: 80.2 - 80.3 (between 00:10:20.200 and 00:10:20.300 UTC).
The attack peaked at that specific moment with 946 packets representing the highest concentration of malicious activity.

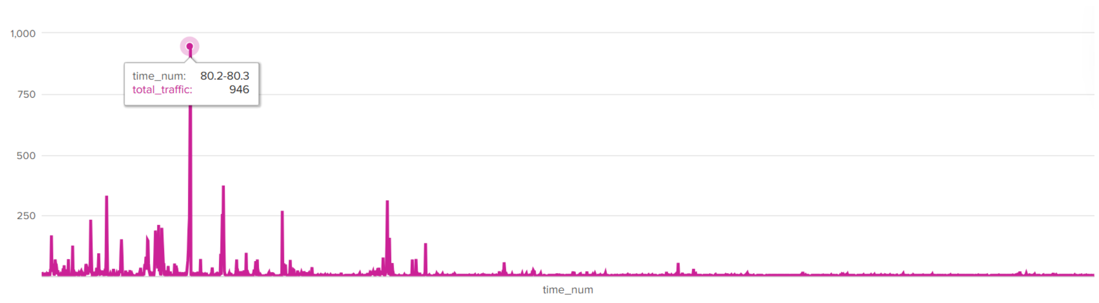


Figure 3: Attack Timeline Plot-line

3.2 Protocol Threat Assessment

Previously, specific attack behaviors were seen (attacker servers, data exfiltration). Now in this section the interest is about what protocols are attackers leveraging and eventually see if Splunk can identify protocol-based threat patterns.

3.2.1 Protocol-based Threat Categorization

The following SPL Query groups all network traffic by protocol, then calculates for each protocol how many packets, average size and maximum size. Then classifies each protocol into a threat category based on some behavioral rules.

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| eval proto = replace(Protocol, "\"", "")
| eval length_num = tonumber(Length)
| stats
    count as packet_count,
    avg(length_num) as avg_size,
    max(length_num) as max_size
    by proto
| sort - packet_count
| eval threat_category = case(
    proto == "QUIC", "ENCRYPTED_EVASION_CHANNEL",
    proto == "TLSv1.3" OR proto == "TLSv1.2",
       "ENCRYPTED_EVASION_CHANNEL",
    proto == "HTTP" AND avg_size > 300, "WEB_BASED_EXFILTRATION",
    proto == "DNS" AND avg_size > 120, "DNS_TUNNELING_SUSPECT",
    proto == "OCSP" AND max_size > 500, "PROTOCOL_ABUSE_HIGH",
    proto == "LDAP" OR proto == "SMB2" OR proto == "DCERPC",
       "LATERAL_MOVEMENT_RISK",
    proto == "TCP" AND packet_count > 10000,
       "HIGH_VOLUME_ATTACK_TRAFFIC",
    1=1, "STANDARD_TRAFFIC"
)
```

Listing 8: Protocol-based Threat Categorization

Table 10: Protocol Threat Mapping Results

Protocol	Packet Count	Avg Size (B)	Max Size (B)	Threat Category
TCP	11,812	655.65	1,514	High volume attack traffic
QUIC	2,151	895.49	1,292	Encrypted Evasion channel
TLSv1.3	1,560	751.45	1,514	Encrypted Evasion channel
HTTP	657	385.27	1,236	Web-based Exfiltration
TLSv1.2	449	460.95	1,514	Encrypted Evasion channel
DNS	419	134.57	328	DNS tunneling suspect
LDAP	256	377.88	1,435	Lateral movement risk
SMB2	235	253.14	1,482	Lateral movement risk
DCERPC	102	301.79	858	Lateral movement risk
OCSP	1	981	981	Protocol abused

Key Findings

- **QUIC, TLSv1.2 and TLSv1.3 protocols** (4,160 packets), encrypted evasion channels;
- **DNS protocol** (419 packets, avg 134 bytes, max 328 bytes) that shows abnormally large query sizes, indicative of DNS tunneling;
- **OCSP** a certificate status protocol was abused to deliver a 981-byte payload, a classic protocol abuse technique;
- **LDAP/SMB2/DCERPC** traffic to and from the infected host. These are internal network protocols that indicates active lateral movement attempts;
- **HTTP** was the only protocol partially detected, confirming that legacy signature-based detection focuses on clear-text web traffic while ignoring modern encrypted and protocol-abuse vectors.

3.2.2 Protocol-based Anomaly Detection (Baseline Comparison)

Here a simple baseline was set: normal packet size = 300 (a typical threshold for alerts), which helps measure how far each protocol deviates from it. This threat categorization framework maps protocol usage to specific attack techniques and evaluates their detection status.

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| eval Protocol = replace(Protocol, "\"", "")
| stats avg(Length) as avg_size by Protocol
| eval normal_size = 300
| eval anomaly = (avg_size - normal_size) / normal_size
| eval risk = if(anomaly > 1, "HIGH", if(anomaly > 0.5, "MED",
  "LOW"))
```

Listing 9: Protocol-based Anomaly Detection

This SPL sets a ”normal” packet size at 300 bytes, then computes avg packet size for each protocol to measure the deviation from the 300 bytes baseline using an anomaly score, then assigns risk levels.

Table 11: Static Baseline Anomaly Detection Results

Protocol	**Avg Size (B)**	**Anomaly Score**	**Risk Classification**
OCSP	981.00	2.27	**HIGH**
QUIC	895.49	1.98	**HIGH**
TLSv1.3	751.45	1.50	**HIGH**
TCP	655.65	1.19	**HIGH**
TLSv1.2	460.95	0.54	**MED**
...			
DNS	134.57	-0.55	LOW
LDAP	377.88	0.26	LOW
SMB2	253.14	-0.16	LOW
DCERPC	301.79	0.01	LOW

Key Observation

A conventional threshold-based alert system would have only flagged OCSP, QUIC, TLSv1.3, and TCP as HIGH risk, while completely missing:

- **DNS tunneling indicators** (134 bytes average classified LOW);
- **Lateral movement protocols** (LDAP, SMB2, DCERPC all classified LOW);

This shows that static threshold alerts are insufficient for detecting malwares like this one, which intentionally maintains protocol characteristics within ”normal” statistical bounds.

3.2.3 Encrypted Traffic Analysis

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| eval proto = replace(Protocol, "\"", "")
| where proto="TLSv1.3" OR proto="TLSv1.2" OR proto="QUIC"
| stats
    count as encrypted_packets,
    sum(Length) as encrypted_volume
| eval encrypted_percentage = round(encrypted_packets/18189*100,
  1)
| eval encrypted_mb = round(encrypted_volume/1048576, 2)
```

Listing 10: Encrypted Traffic Analysis

This SPL filters only encrypted protocols (TLSv1.2, TLSv1.3, QUIC), then counts total encrypted packets and sum their bytes, to then calculate what percentage of total traffic is encrypted.

Table 12: Encrypted Traffic Analysis Results

Metric	**Value**
Total Encrypted Packets	4,160
Total Encrypted Volume	3,305,427 bytes
Encrypted Traffic Volume	3.15 MB
Percentage of Total Traffic	22.9%
Encrypted Threats included in Alerts	**0%**

Results

- **22.9%** of all network traffic was encrypted (QUIC + TLSv1.2 + TLSv1.3);
- This represents 3.15 MB of data that cannot be inspected;
- WarmCookie malware abused **QUIC** which is a modern encrypted transport protocol that many security tools do not fully decrypt or inspect;
- The existing security alerts didn't include most encrypted threats.

3.3 Statistical Detection

3.3.1 Behavioral Clustering Analysis

Understanding behavior types helps categorize attacks: data theft vs attacker servers communication vs malware delivery. What patterns of behavior do we see? Grouping similar activities reveals attack strategies.

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| rex ",\"(?<Source>[^\"]+)\",\"(?<Destination>[^\"]+)\",
  \"(?<Length>[^\"]+)\""
| eval src = replace(Source, "\"", "")
| eval dst = replace(Destination, "\"", "")
| eval size = tonumber(replace(Length, "\"", ""))
| stats
    count as frequency,
    avg(size) as avg_size,
    stdev(size) as size_variation
    by src, dst
| eval behavior_type = case(
    frequency > 10 AND avg_size > 1000,
       "HIGH_VOLUME_DATA_TRANSFER",
    frequency > 10 AND avg_size < 100, "HIGH_VOLUME_CONTROL",
    frequency <= 3 AND avg_size > 500, "TRANSFER_SURGES",
    true(), "NORMAL_COMMUNICATION")
| stats count by behavior_type
```

Listing 11: Clustering Analysis

This SPL query computes frequency of the communication, the average message size, the consistency size for each src-dst pair, then evaluates the behavior-type: "High volume data transfer" for lots of big data, "High volume control" for lots of small control messages, and "Transfer Surges" for few but big transfers and "Normal communication" for everything else. Finally, it counts each behavior type.
Results:

Table 13: Behavioral Communication Pattern Analysis

Behavior Type	**Count**	**Security Interpretation**
High Volume Data Transfer	10	Malware delivery and data exfiltration
High Volume Control	5	Attacker servers communications
Normal Communication	132	Background network activity
Total Communication Patterns	**147**	

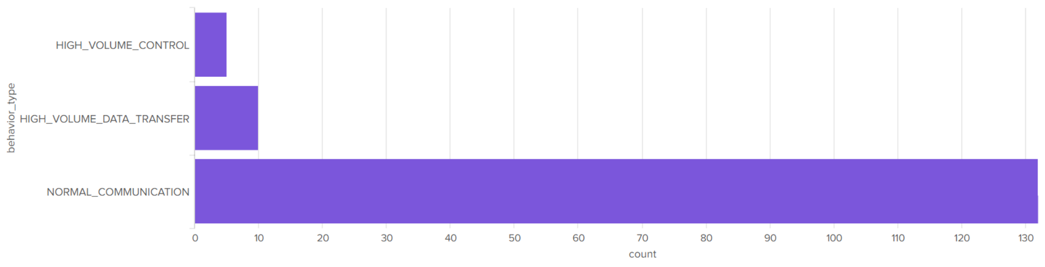


Figure 4: Behavioral Communication Pattern Bar plot

So it finds 10 different "delivery routes" for malware and 5 different "control channels" for the attackers to send commands.

3.4 AI Toolkit

To evaluate Splunk's AI Toolkit (previously Machine Learning Toolkit) few tests were run on the infected host (10.8.15.133) traffic.

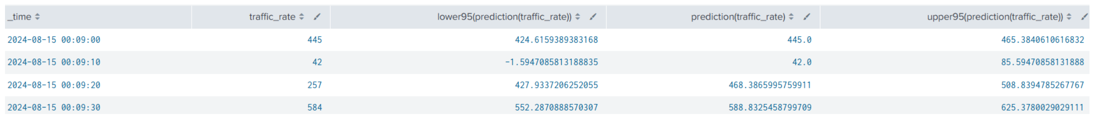
3.4.1 Time Series Forecasting

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| where Source="10.8.15.133"
| eval _time = strptime(Absolute_Time, "%Y-%m-%d %H:%M:%S.%f")
| timechart span=10s count as traffic_rate
| predict traffic_rate algorithm=LLT
| eval prediction_error = abs(traffic_rate -
  'prediction(traffic_rate)')
```

Listing 12: Time Series Forecasting

The query uses the **LLT = Local Level Trend** algorithm (exponential smoothing), which learns the normal traffic pattern and predicts what the next value should be.

Then it compares actual vs. predicted to measure accuracy.



_time	traffic_rate	lower95(prediction(traffic_rate))	prediction(traffic_rate)	upper95(prediction(traffic_rate))
2024-08-15 00:09:00	445	424.6159389383168	445.0	465.3840610616832
2024-08-15 00:09:10	42	-1.5947085813188835	42.0	85.59470858131888
2024-08-15 00:09:20	257	427.9337206252055	468.3865995759911	508.8394785267767
2024-08-15 00:09:30	584	552.2870888570307	588.8325458799709	625.3780029029111

Figure 5: Time Series Forecasting results

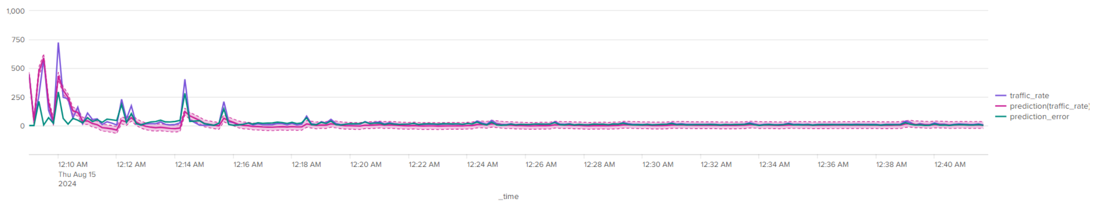


Figure 6: Time Series Forecasting Plot-line

The query presented 195 statistics, among these lines we have some interesting observations such at 00:09:20, actual traffic was 257 packets, but the algorithm predicted 468 packets. This shows that it successfully anticipated the upcoming spike at 00:09:30 (584 packets). We can eventually observe from the line chart that the algorithm does a pretty good job at forecasting spikes.

3.4.2 Anomaly Detection

```
| inputlookup warmcookie_traffic_with_absolute_time.csv
| where Source="10.8.15.133"
| eval _time = strptime(Absolute_Time, "%Y-%m-%d %H:%M:%S.%f")
| timechart span=10s avg(Length) as avg_packet_size
| anomalydetection avg_packet_size
```

Listing 13: Anomaly Detection

The query uses **anomalydetection** (Probabilistic Autoregressive) algorithm to look for statistical outliers in packet sizes.
The algorithm is returning 91 statistics with identical sores. So the algorithm executes the query giving our data but outputs constant values (a flat line) even when unusual events occur, which we can observe in the following figure, bluntly ignoring obvious anomalies.

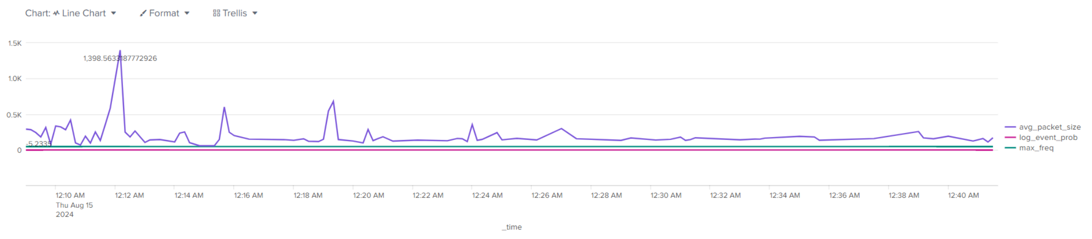


Figure 7: Anomaly Detection Plot-line

- Line chart: avg packet size (shows actual attacks: 1398 bytes!)
- Second Y-axis: log event prob (shows flat line = algorithm failure)

From these tests, we can definitely notice that Splunk AI toolkit shows selective utility, and in some cases, it isn't only giving unsatisfactory results but refuses entirely to process our security incident data, such as with ARIMA (Autoregressive Integrated Moving Average) algorithm that returns *command="predict" No data* errors when presented with this data traffic. And the main reason is that it requires many data points over hours or days and our 32.7 minutes attack window isn't enough for ARIMA to process.
This part shows that it is not always possible and simple to point AI Toolkit Algorithms at security data and expect results. The mathematical requirements of some algorithms is (large n, low variance, long baselines) which are incompatible with security incident characteristics (small n, high variance, short duration). These cases are simply a tool-data mismatch.

- **LLT:** Actually useful (predicted attack earlier)
- **Anomalydetection:** Silent failure (looked like it worked, but it didn't)
- **ARIMA** "No data" errors (impossible to process)

3.5 Validation

3.5.1 Alerts Correlation

Here comes the validation phase: a comparison between what Splunk discovered against what we already knew with the Alerts file. It's essentially asking how much of the threats we initially knew, and how much new threats did Splunk discover.

The alerts file had:

- **3 alerts** about HTTP traffic to/from 72.5.43.29 (the main attacker server). These caught the malware download and some POST backs.
- **2 alerts** about HTTP traffic to 23.205.110.48 (another attacker server). These caught some early attack reconnaissance and connectivity tests.
- **1 alert** about TLSv1.0 (outdated encryption).

- **2 alerts** about internal stuff (DNS errors and RPC kerberos traffic). These represented noise, not the main event.
- **8 alerts** in total that actually related to the attack.

What wasn't in the alerts file:

- **3,711 encrypted packets** (TLSv1.3 + QUIC) - **0 alerts**
- **593 lateral movement packets** (LDAP, SMB2, DCERPC) - **0 alerts**
- **419 DNS tunneling packets - 0 alerts**
- **OCSP protocol abuse** (981-byte payload) - **0 alerts**
- **28+ minutes of post-compromise activity - 0 alerts**

The alerts caught what they were designed to catch. They caught HTTP malware downloads. They caught some HTTP attacker servers traffic. They caught outdated TLS. However it wasn't broad enough. The alerts didn't mention encrypted traffic or DNS unusual packet sizes. Also LDAP, SMB, and OCSP were most probably trusted to be harmless.

4 Results and Findings

4.1 Quantitative Findings

Table 14: Quantitative Findings Summary

Metric	**Value**
Total Malicious Connections Identified	5,426
Alert-Covered Connections	1,849
Splunk-Discovered Connections	**3,577**
Threat Discovery Rate	**65.92%**
Data Exfiltration Volume	1.14 MB
Attacker Servers Identified	4
Malware Delivery Attempts	6,275
Attack Infrastructure Scale	60+ IPs
Encrypted Traffic (TLSv1.3/QUIC)	3,711 packets (22.9%)
Lateral Movement Attempts	593 packets
DNS Tunneling Indicators	419 packets
Protocol Abuse (OCSP)	981-byte payload

4.2 Qualitative Findings

Beyond the metrics, this analysis revealed several qualitative insights which answered our initial question, **yes, Splunk found a lot that the alerts file missed.** About 65% more malicious. But it is even more interesting:

- The malware wasn't particularly sophisticated, it just prioritized encryption and abused protocols that security tools trust.
- Distributed Infrastructure: The attack utilized 60+ external servers rather than focusing on centralized Attacker servers.
- Splunk's AI Toolkit was inconsistent, some algorithms worked, some failed silently, and some couldn't even run on 32 minutes of data.
- The biggest wins came from manual SPL queries, not automation.

5 Splunk Evaluation

5.1 Strengths

- **Detection Gap Quantification:** Splunk SPL analysis identified 3,577 more malicious connections (65.92%) which represents a big improvement in threat coverage and demonstrates the platform's value as a threat detection layer.
- **Comprehensive Threat Visibility:** Unlike signature-based alerts which detected only HTTP-based malware delivery, Splunk analysis revealed entire categories of undetected threats.
- **Flexible Query Language:** SPL enabled complex correlation analysis across multiple dimensions (source, destination, protocol, packet size, timing) that would be difficult or impossible with signature-based detection alone.
- **Real-time Analytics Potential:** Although conducted retrospectively, the analysis successfully reconstructed the full attack progression within the 4-minute attack window, demonstrating Splunk's capability for temporal threat analysis.

5.2 Limitations

- **Data Preparation:** Even though our study case didn't essentially need such data cleaning, many logs require extensive manual data cleaning (`rex` + `replace` operations in nearly every query). Splunk's dependence on well-structured data inputs necessitates additional preprocessing tooling or standardized log formats.
- **Silent Failure:** The `anomalydetection` algorithm executed successfully but returned constant scores for 91 intervals, completely missing obvious packet size anomalies (e.g., 1,398 bytes at 00:12:10).
- **Incompatibility:** Complex algorithms (ARIMA,) failed with `"No data"` errors on 32.7 minutes of attack traffic. This is not a defect but a fundamental mathematical constraint. Algorithms such as ARIMA require hours/days/weeks of baseline data that not all security incidents can provide.
- **Skill Barrier:** Advanced SPL expertise is required, representing training investment.

6 Security Recommendations

1. **Isolate infected host** (10.8.15.133) to prevent further compromise.
2. **Block identified Attack infrastructure** by implementing firewall rules for 72.5.43.29, 23.205.110.48, 23.33.138.184, and 60+ additional malicious IPs.
3. **Investigate domain controller** (10.8.15.4) for movement evidence. There were kerberos overflow attempts and LDAP traffic to external IPs (someone might be trying to steal credentials).
4. **Encrypted Traffic Profiling** for QUIC and TLSv1.3 to external destinations. While packet contents cannot be inspected, metadata like (frequency, size distribution, destination) can identify anomalous encrypted channels.
5. **Query Depth Monitoring** by tracking query sizes and responses payload lengths. Any response exceeding a certain threshold of bytes from external resolvers should trigger investigation for potential tunneling.
6. **Protocol Abuse Detection** by monitoring non-web protocols (OCSP, CRL, NTP) for anomalous payload sizes, since these protocols are rarely used for data transfer.
7. **Lateral Movement Baseline** by establishing communication baselines for LDAP, SMB, and DCERPC traffic and any deviation from normal patterns involving the infected host should trigger immediate containment.

7 Conclusion

This evaluation demonstrated Splunk Enterprise's capability to identify more malicious traffic than the existing security alerts, especially in an environment where missing even one threat can lead to catastrophic breaches and where threat visibility is no longer acceptable.
Despite the limitations of the tool, its effectiveness provides actionable security enhancement. The platform's true value lies not in automated detection but in enabling human analytical reasoning through its flexible query languages. Therefore a key takeaway from this study is is to prioritize SPL proficiency over AI Toolkit automation.
Finally, The 65.92% discovery rate represents not merely more alerts, but detection of entire categories of threats that the initial security alerts didn't cover and this validates Splunk's effectiveness as a security analytics platform.