SIEM Project

Introduction

In an era of increasing cyber threats and data breaches, organizations are continuously seeking ways to enhance their security posture. Security Information and Event Management (SIEM) systems play a pivotal role in this pursuit by providing a comprehensive solution for monitoring, analyzing, and responding to security events and incidents.

This project aims to create a hands-on experience in designing and configuring a basic SIEM setup. It serves as an opportunity for me as a beginner to gain practical knowledge in the field of cybersecurity and SIEM operations. By the end of this project, I will have developed essential skills related to data collection, incident detection, and security event analysis.

Goals:

- Set up a functional lab environment simulating a network with various log sources.
- Configure a SIEM platform for collecting and analyzing security-related logs.
- Create custom alerts and rules based on security best practices.
- Implement incident detection and response procedures using the SIEM.
- Develop skills in data visualization and reporting.
- Document the project and summarize key findings and takeaways.
- Foster a continuous improvement mindset for future security projects.

Project Setup

It's crucial to establish a well-defined lab environment and select a SIEM platform that aligns with your goals. Here, I'll outline the fundamental aspects of the project setup. I'll split the requirements into 2 categories, as follows:

1. **Hardware**, as in basic computer gear such as:

- A computer with sufficient processing power and memory for running virtual machines (VMs).
- Network equipment (e.g., routers, switches) for creating a simulated network.

2. **Software**

- A hypervisor such as VMware Workstation, VirtualBox, or a cloud-based solution like AWS or Azure for creating VMs.
- Operating systems for the VMs (e.g., Windows Server, Linux distributions).

Next, as network specifications, the home network bill be used for the entire process and in addition to that, we can use multiple devices if necessary. The home network can include: servers, firewalls and routers, endpoints, switches and hubs.

For our SIEM platform selection, we will use **Splunk**, as it is a widely-used commercial SIEM platform with a robust set of features that will give me the chance to experience as close as it can get to the real-world scenarios. The alternatives could be **ELK Stack (Elasticsearch, Logstash, Kibana)** which is a open-source solution known for its scalability and flexibility or **OSSIM (Open Source Security Information Management)** which is a free and open-source SIEM solution.

Data Collection

Step 1: Set Up Log Sources

Before we can begin collecting data, I'll need to configure log sources to generate and send logs to our SIEM platform. Log sources can be varied, but a few examples could be:

- **Servers**
    - **Web Servers**: Configure web servers (e.g., Apache, Nginx) to generate access logs, error logs, and application-specific logs.

    - **Database Servers**: Enable logging on database servers (e.g., MySQL, PostgreSQL) to capture database-related events.
    - **Authentication Servers**: For Windows environments, enable security event logging (e.g., Active Directory logs).
    - **Application Servers**: If applicable, configure application servers (e.g., Tomcat, IIS) to generate logs related to application activities.
- **Network Devices**
    - **Firewalls and Routers**: Configure these devices to log network traffic, security events, and firewall rule violations.
    - **Switches and Hubs**: Set up logging for network switches and hubs to monitor network traffic and port activity.
- **Endpoints**
    - **Desktops and Laptops**: Enable endpoint security software to generate logs related to antivirus, firewall, and system events.
    - **Mobile Devices**: If mobile devices are part of the lab environment, configure mobile device management (MDM) tools for logging.

We will proceed with setting up an Apache Web Server.

Let's set up a web server on Windows 10 and configure it to generate logs. In this example, we'll use XAMPP, which is a package that includes the Apache web server, MySQL database, and PHP:

1. Download and Install XAMPP:

- Visit the XAMPP download page: **https://www.apachefriends.org/index.html**
- Download the Windows version of XAMPP that includes Apache.
- Run the installer and follow the on-screen instructions to install XAMPP. We can choose the default settings for most options during installation.

2. Start the Apache Server:

- After installation, launch the XAMPP Control Panel.
- Click the "Start" button next to "Apache" to start the Apache web server.

3. Create a Sample Web Page:

- Open a text editor (e.g., Notepad) and create a simple HTML file. Save it as "index.html" in the "htdocs" folder within the XAMPP installation directory. By default, this directory is located at `C:\xampp\htdocs`.

```
<!DOCTYPE html>
<html>
<head>
    <title>Sample Web Page for fg SIEM Project</title>
</head>
<body>
    <h1>Welcome to my web page!</h1>
</body>
</html>
```

4. Access the Web Page:

- Open a web browser and enter the following URL in the address bar: `http://localhost/index.html`.
- We should see the sample web page you created.

5. Generate Apache Access Logs:

By default, Apache logs access events to a file called "access.log." You can find this log file in the "logs" folder within the XAMPP installation directory (`C:\xampp\apache\logs`).

To generate some log entries, we can simply access the web page through the browser a few times. Each time we access the page, Apache will log the request in the "access.log" file.

6. View the Log Entries:

- Open the "access.log" file using a text editor (e.g., Notepad) to view the logged entries. Our log entries will look like the following:
-

```
::1 - - [03/Oct/2023:09:01:55 +0300] "GET /index.html HTTP/1.1" 404 295 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
::1 - - [03/Oct/2023:09:01:56 +0300] "GET /favicon.ico HTTP/1.1" 200 30894 "http://localhost/index.html" "Mozilla/5.0 (Windows NT 10.0; Win
::1 - - [03/Oct/2023:09:03:33 +0300] "GET /index.html HTTP/1.1" 200 160 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
```

These log entries contain information about the IP address, request method, requested URL, HTTP status code, and more.

We've now configured Apache to generate access logs that can be used for our SIEM project. In addition, we can explore Apache's documentation for more advanced logging options and configurations if needed.

Supplementary, we can create a rule that will direct incoming network traffic from outside the home network (using IPv4) to our locally hosted Apache Web Server using Dynamic Domain Name System(DDNS). By setting this up, we can have a normally looking web address such as “petricamarius.home.ro” that will update it’s host IP address every time the host’s IP changes. That’s mainly because the home network has additional security features included, like interchangeable IPv4 after a certain amount of time.

This time, we can use the curl command to fast-generate some log data. We can simulate additional access requests by using this tool or scripts with this tool. The steps are:

1. **Install Curl (if not already installed):**
   If we didn’t have `curl` installed on our Windows system, we could’ve downloaded it from the Curl website (**https://curl.se/windows/**).
2. **Use Curl to Simulate Access Requests:**
   Next, we open a Command Prompt (CMD) or PowerShell window and use `curl` to access our web page multiple times. The following command simulates 10 requests to our web page:

```
for /l %i in (1,1,10) do curl http://localhost/index.html
```

This loop will send 10 GET requests to our web page, and each request will be logged in the Apache access logs.

We will also generate some “file not found” GET requests using the following code:

```
for /l %i in (1,1,10) do curl http://localhost/login.php
```

By using this command, we try to access a non-existing file in our Web Page (login.php). We will get the response as follows:

```
127.0.0.1 - - [03/Oct/2023:09:29:32 +0300] "GET /login.php HTTP/1.1" 404 295 "-" "curl/8.0.1"
127.0.0.1 - - [03/Oct/2023:09:29:32 +0300] "GET /login.php HTTP/1.1" 404 295 "-" "curl/8.0.1"
```

3. **Check the Apache Access Logs:**
   After running the above command, we can check the "access.log" file again to see the increased number of log entries.

We can adjust the loop parameters as needed to generate the desired amount of data for our SIEM project.

Step 2: Data Parsing and Normalization. Search queries, tables, reports.

Now that we already have some data in the access.log file, the next step is to configure Splunk to understand and extract structured information from these log entries.

It’s generally better to create specific indexes, so we need to go into “Settings” → “Indexes” → “New Index” and give it a name such as “apache_web”.

2.1.Log Parsing

- **Configure inputs.** In Splunk, log data is ingested through inputs. Access the Splunk web interface and click "Settings" → "Data Inputs".
- **Select Input Type**: Choose the appropriate input type based on our log source. For Apache access logs, we can use the "Files & Directories" input type.
- **Specify Log File Location**: Configure the input by specifying the path to the Apache access log file (`access.log`) or the directory where it resides. Ensure that the input is enabled.
- After this, we need to select the index that we would like to use, in our case it's the "apache_web" index we previously created. We'll leave the source type as default: access_combined.

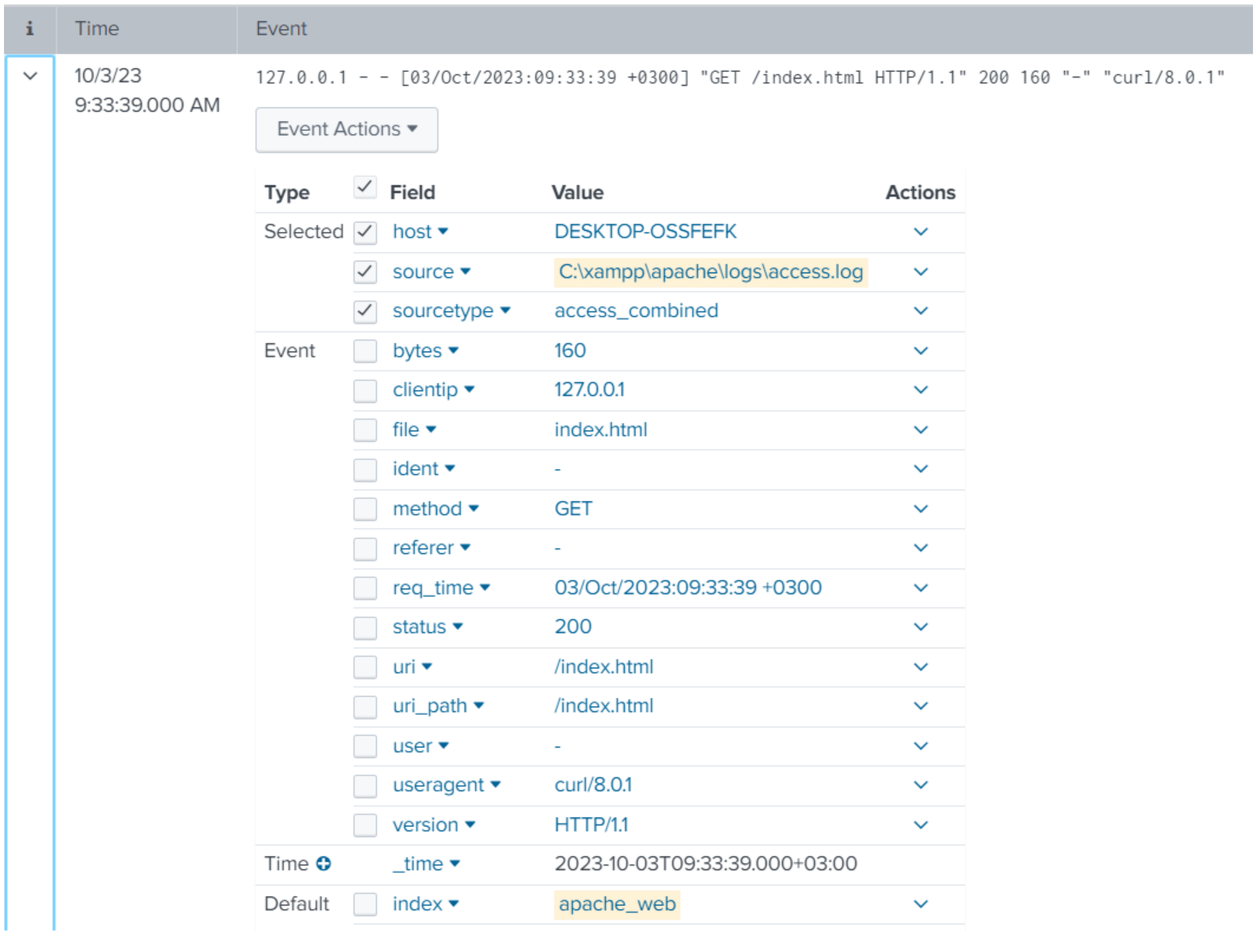
2.2.Splunk Configuration

- **Create Searches and Reports**: After parsing and normalizing log data, create saved searches, alerts, dashboards, and reports in Splunk to monitor and analyze the Apache access logs effectively.
- **Custom Dashboards**: Design custom dashboards that visualize log data, showing metrics like top URLs, client IP addresses, response codes, and more.
- **Scheduled Reports and Alerts**: Set up scheduled reports and alerts based on specific log events or conditions. Splunk can proactively notify you of security incidents or anomalies.

We can perform a simple search as an example using the following search structure and filters in the search bar of Splunk:

```
index="apache_web" source="C:\\xampp\\apache\\logs\\access.log"
```

Using this, we get the following events captured in our log file.

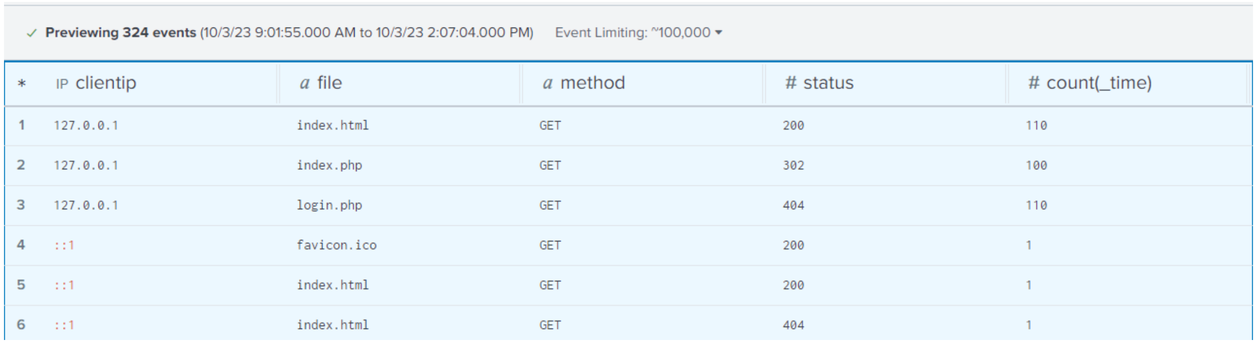


i	Time	Event
	10/3/23 9:33:39.000 AM	`127.0.0.1 - - [03/Oct/2023:09:33:39 +0300] "GET /index.html HTTP/1.1" 200 160 "-" "curl/8.0.1"`

Type	✓	Field	Value	Actions
Selected	✓	host	DESKTOP-OSSFEFK	
	✓	source	C:\xampp\apache\logs\access.log	
	✓	sourcetype	access_combined	
Event		bytes	160	
		clientip	127.0.0.1	
		file	index.html	
		ident	-	
		method	GET	
		referer	-	
		req_time	03/Oct/2023:09:33:39 +0300	
		status	200	
		uri	/index.html	
		uri_path	/index.html	
		user	-	
		useragent	curl/8.0.1	
		version	HTTP/1.1	
Time		_time	2023-10-03T09:33:39.000+03:00	
Default		index	apache_web	

Splunk has already understood the data according to the logs and created relevant fields to help our analysis. For example, it has discovered that the client's IP address is present in each row of our file and has created a field called "clientip" that stores those values such as "127.0.0.1" as our localhost machine. It's got the HTTP method ("GET") as "method", the status received "200" and more.

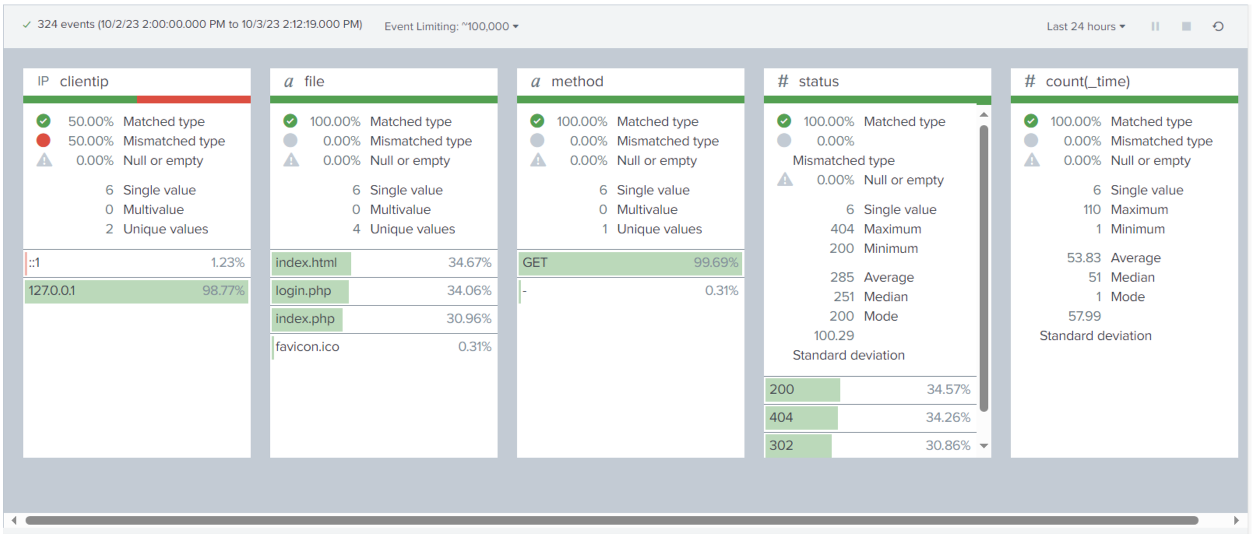
Even more, we can create summarized tables like the one here:



✓ **Previewing 324 events** (10/3/23 9:01:55.000 AM to 10/3/23 2:07:04.000 PM) Event Limiting: ~100,000

*	clientip	file	method	status	count(_time)
1	127.0.0.1	index.html	GET	200	110
2	127.0.0.1	index.php	GET	302	100
3	127.0.0.1	login.php	GET	404	110
4	::1	favicon.ico	GET	200	1
5	::1	index.html	GET	200	1
6	::1	index.html	GET	404	1

Or, views that look like dashboards, where we can see the percentages and values of every "split by" variable:


✓ 324 events (10/2/23 2:00:00.000 PM to 10/3/23 2:12:19.000 PM) Event Limiting: ~100,000 Last 24 hours

IP clientip
50.00% Matched type
50.00% Mismatched type
0.00% Null or empty
6 Single value
0 Multivalue
2 Unique values

::1	1.23%
127.0.0.1	98.77%

a file
100.00% Matched type
0.00% Mismatched type
0.00% Null or empty
6 Single value
0 Multivalue
4 Unique values

index.html	34.67%
login.php	34.06%
index.php	30.96%
favicon.ico	0.31%

a method
100.00% Matched type
0.00% Mismatched type
0.00% Null or empty
6 Single value
0 Multivalue
1 Unique values

GET	99.69%
-	0.31%

status
100.00% Matched type
0.00% Mismatched type
0.00% Null or empty
6 Single value
404 Maximum
200 Minimum
285 Average
251 Median
200 Mode
100.29 Standard deviation

200	34.57%
404	34.26%
302	30.86%

count(_time)
100.00% Matched type
0.00% Mismatched type
0.00% Null or empty
6 Single value
110 Maximum
1 Minimum
53.83 Average
51 Median
1 Mode
57.99 Standard deviation

Even more, we can create complex search queries and filter out the data that we do not need for the current view, such as:

```
index="apache_web" source="C:\\xampp\\apache\\logs\\access.log" clientip="127.0.0.1" status="200" method="GET"
```

By using this, only 110 events will be shown out of 324.

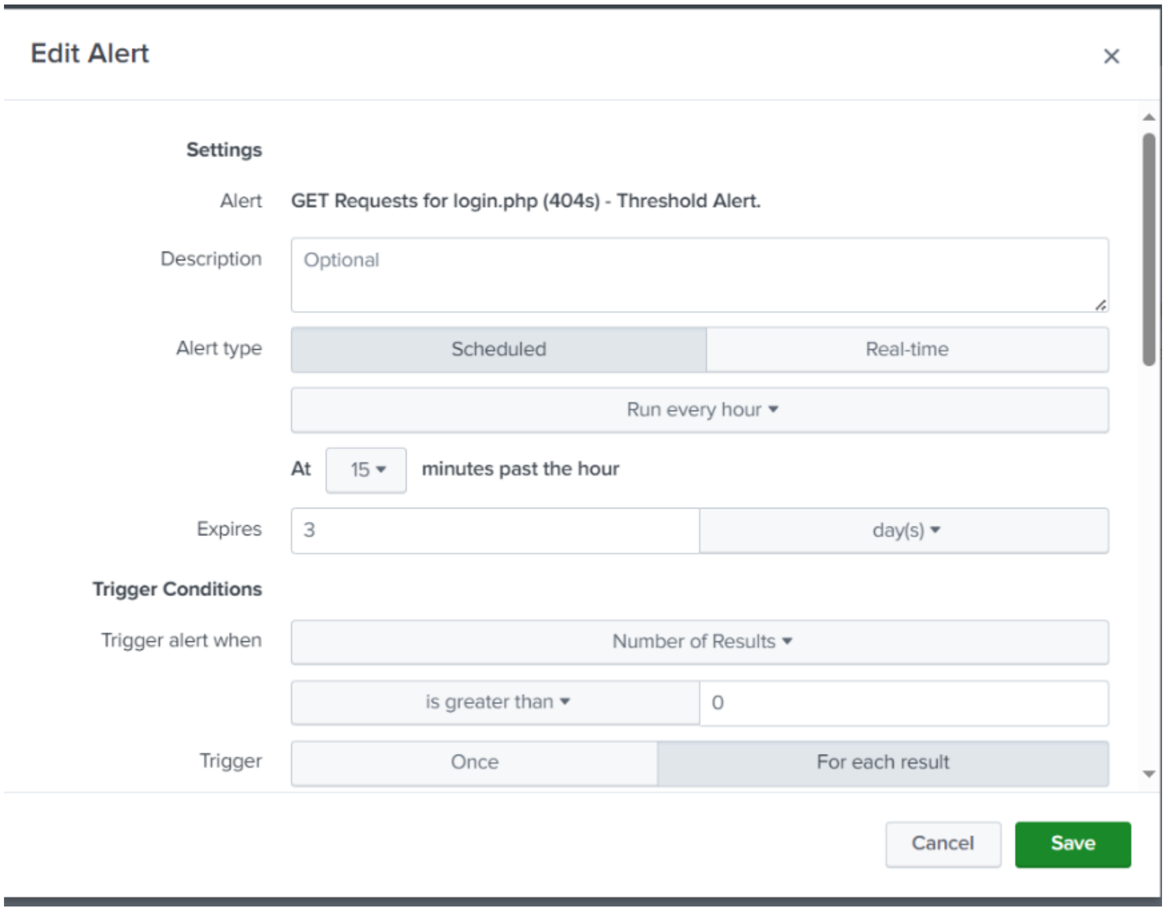
Custom Alerts and Rules (for Incident Detection)

Step 3: Create Custom Alerts

In Splunk, we can create custom alerts based on our queries. For example, we can use the following syntax:

```
index="apache_web" status=404 method="GET" "/login.php" | timechart span=1h count as requests | where requests> 100
```

This will only show the number of GET requests on our Apache Web Server and for the file login.php if the requests are counting at least 100 in an 1 hour window.


Edit Alert

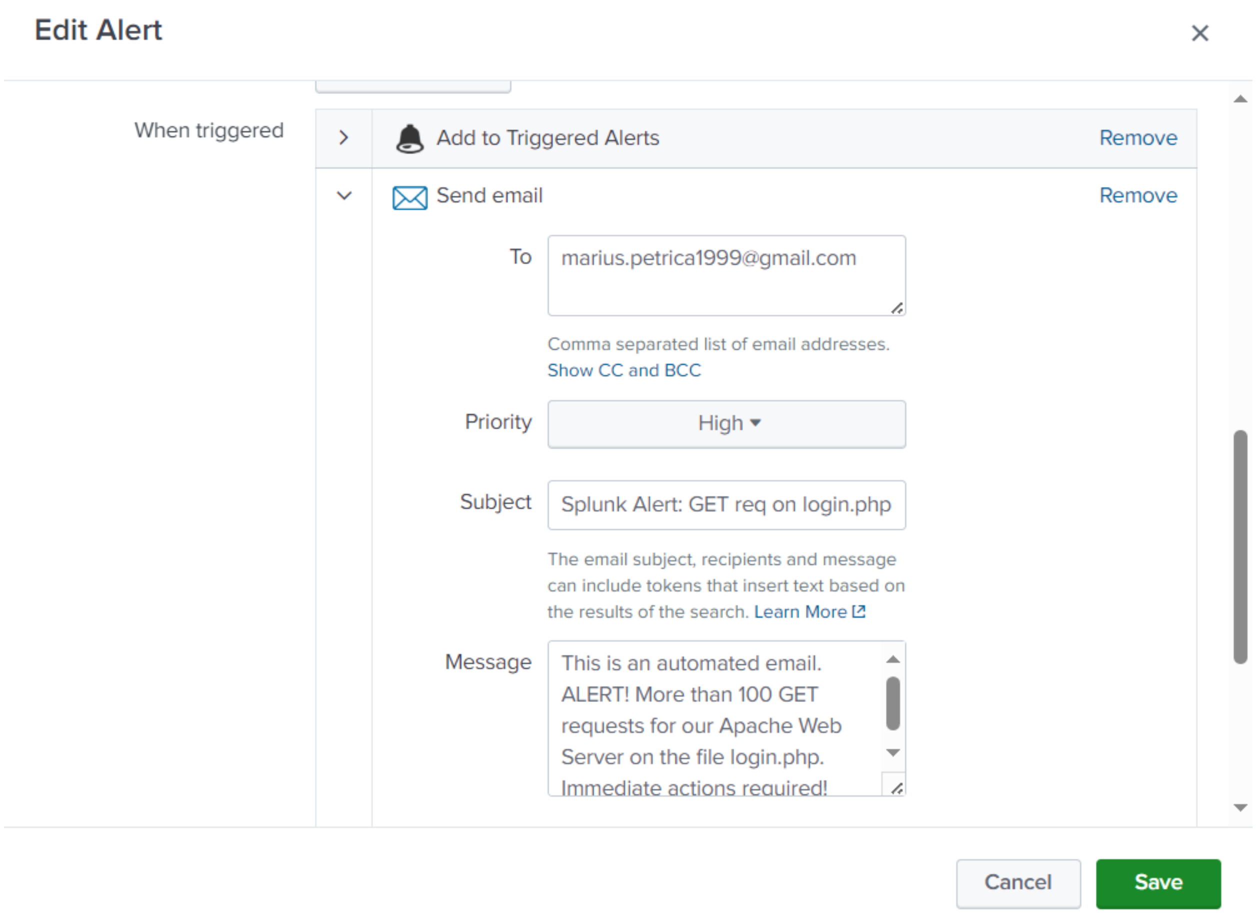
Settings

Alert	GET Requests for login.php (404s) - Threshold Alert.
Description	Optional
Alert type	[✓ Scheduled] Real-time
	Run every hour
	At 15 minutes past the hour
Expires	3 day(s)

Trigger Conditions

Trigger alert when	Number of Results
	is greater than 0
Trigger	Once [✓ For each result]

We set it up with a specific Alert name, Description(optional), schedule it to run every hour and set trigger conditions as the number of results>0.


Edit Alert

When triggered

Add to Triggered Alerts — Remove

Send email — Remove

To: marius.petrica1999@gmail.com

Comma separated list of email addresses.
Show CC and BCC

Priority: High

Subject: Splunk Alert: GET req on login.php

The email subject, recipients and message can include tokens that insert text based on the results of the search. Learn More

Message: This is an automated email. ALERT! More than 100 GET requests for our Apache Web Server on the file login.php. Immediate actions required!

Cancel | Save

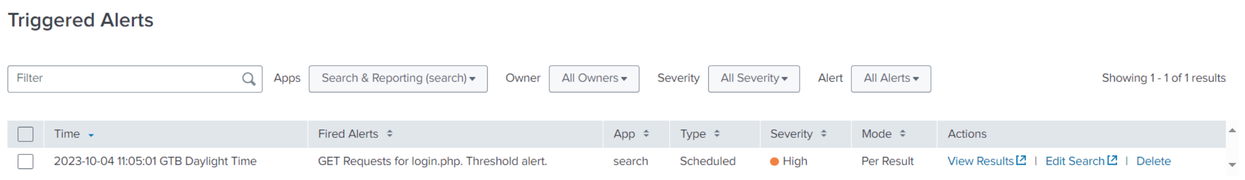
As an action, we want to send an email to the recipients with High Priority and the subject “Splunk Alert: GET req on login.php!”. The message can be a little bit detailed but we will try to be as short as possible. We will use the following text:

```
This is an automated email.
ALERT! More than 100 GET requests for our Apache Web Server on the file login.php.
Immediate actions required!
```

So now, if we try to trigger this alarm by using the same command as before, we will generate 101 curl requests on login.php and will receive an email of the alert:

```
for /l %i in (1,1,101) do curl http://localhost/login.php
```

For this email alert to work, we would have to setup an email in Splunk Settings, using Mail Server Settings. We will not cover this in the project. Instead, we will use the secondary action which is → Show alert in Triggered Alerts section, as follows:


Triggered Alerts

Filter | Apps: Search & Reporting (search) | Owner: All Owners | Severity: All Severity | Alert: All Alerts | Showing 1 - 1 of 1 results

Time	Fired Alerts	App	Type	Severity	Mode	Actions
2023-10-04 11:05:01 GTB Daylight Time	GET Requests for login.php. Threshold alert.	search	Scheduled	● High	Per Result	View Results | Edit Search | Delete

Using some alerts like the one we created, we can monitor even in real-time by creating dashboards and alerts on critical security events as they happen. We can go further to communicate with the security team and ensure that our incident detection process is aligned with the organization's standards and response plan.

Incident Detection and Response

Step 5: Incident Response

t's the process of taking action when an incident is detected to mitigate its impact and prevent further harm. In this context, we'll focus on how to use Splunk for incident response to address the incidents detected in Step 4. Here's how you can set up incident response using Splunk:

5.1.Incident Triage

- When an incident alert is triggered by Splunk based on your defined criteria, the first step is to triage the incident. This involves quickly assessing the incident's severity, scope, and potential impact.
- Use the alert details and any available context from the alert to understand what occurred, including the source IP, affected systems, and the nature of the incident.

5.2. Investigate the Incident

- Access the Splunk search and reporting interface to investigate the incident further. We can refine our search queries to gather more information about the incident.
- For example, if we would receive an alert about multiple failed login attempts, we can search for related log entries to determine the source IP addresses, timestamps, and any additional details about those login attempts.

5.3. Contain the Incident

- Once we have a clear understanding of the incident, we can take appropriate containment actions to prevent further damage or unauthorized access. Depending on the incident type, containment actions may include:
    - Blocking the source IP addresses associated with the incident.
    - Disabling compromised user accounts.
    - Isolating affected systems from the network.
    - Changing credentials or passwords.

5.4. Communicate and Document

- Communication is vital during incident response. Notify relevant stakeholders, such as the security team, IT department and management, about the incident.
- Document all actions taken during the incident response process. This includes details of the incident, steps taken for containment, communication with stakeholders, and any changes made to systems or configurations.

5.5. Remediate and Recover

- After containment, we need to focus on remediating the root cause of the incident. Address the discovered vulnerabilities, update security configurations, and implement changes to prevent similar incidents in the future.
- Work on restoring affected systems or services to normal operation. This may involve applying patches, restoring backups, or rebuilding compromised systems.

5.6. Review and Lessons Learned

- Conduct a post-incident review to analyze what happened, why it happened, and what can be done to prevent similar incidents in the future.
- Identify lessons learned and update your incident response procedures and security measures accordingly.

5.7. Automate Incident Response (Optional)

- For repetitive or well-defined incident types, we could consider automating parts of the incident response process. Splunk provides integration with automation tools that can perform predefined actions in response to specific alerts. One example Splunk can do is run scripts when an alert is triggered.

5.8. Continuous Improvement

- The last thing we can do is use the insights gained from incident response to continuously improve our security posture. In addition, we can regularly review and update our incident response plan, security policies, and detection rules in Splunk.

Conclusion

As a short conclusion, Splunk can be efficiently used for incident response to address security incidents and vulnerabilities identified in our Apache web server logs. Splunk's capabilities for real-time monitoring, search, and alerting can help streamline the incident response process and enhance your organization's security resilience.

As we have seen, SIEM means Security Information and Event Management (SIEM) and this action can be done using different methods, tools and procedures. In my opinion, I think it’s crucial to experience with the systems in a way that we would not be caught off guard. We need to use the most out of the past incidents, responses and see exactly how we managed each event. As an organization, we should have well-defined terms and conditions in a way that we can take actions quick, as alerts appear. Even a short instruction plan will help in these kinds of situations and it does not matter if it’s an organization-level plan, team plan or personal plan.

Next Steps

My personal next-steps regarding this project are based on developing or adapting it to a SOAR, which is a software solution that enables security teams to integrate and coordinate separate tools into streamlined threat response workflows. This would be very beneficial because in large organizations, security operations centers (SOCs) rely on numerous tools to track and respond to cyberthreats.