ProTego

Creating a SIEM with the Elastic Stack or with OpenSearch

Whitepaper

ProTego: DATA-PROTECTION TOOLKIT REDUCING RISKS IN HOSPITALS AND CARE CENTERS

Project Number 826284

Call: H2020-SU-TDS-02-2018. Trusted digital solutions and Cybersecurity in Health and Care

Executive Summary

Nowadays every organization is exposed to the intrinsic risks associated to the use of Information Technology. Cyber threats are more powerful and dangerous all the time. Data breaches cost organizations millions of euros every year. Therefore, they must prepare and protect infrastructures for such attacks even anticipating them if possible.

A SIEM implements a set of technologies able to help detect, respond, and neutralize cyber threats. The main objective is to give an organization a global vision of IT security, allowing to have its complete control. By collecting and managing information about events that take place it is easier to detect trends and focus on anomalies.

But the cost involved in purchasing, deploying, and customizing a commercial SIEM is high and beyond the budget of many organizations. This document is not intended to be an in-depth dissertation about the matter, rather it is a starting point that gives organizations with limited budgets ideas on how to use the Elastic Stack or the OpenSearch project, based on the experience of the ProTego.

Introduction

The underlying principle of a SIEM is that security-relevant data in an organization takes place in multiple locations. By being able to see all that data from a "single pane of glass" makes it a lot easier to detect trends and uncommon patterns.

The concept of a SIEM rises from combining the functions of two different kinds of systems:

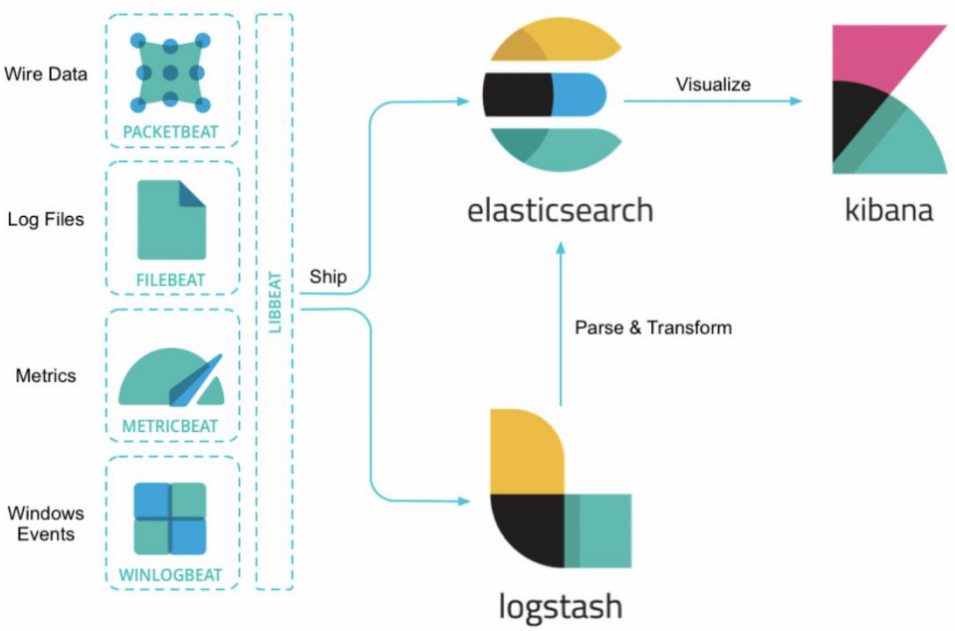
- SEM: Security Event Management. A system that centralizes the storage of information and allows a near real-time analysis of what is taking place in security management, detecting and analyzing abnormal patterns.
- SIM: Security Information Management. A system that collects long-term data on a central repository to be analyzed later, providing automated reports.

Elastic Stack / OpenSearch

To accomplish that objective a SIEM data is collected from many different sources, like logs, metrics, network packets, etc. This will evidently generate large amounts of data. A platform that has become the *de facto* standard to store that kind of information is Elastic Stack (also known as ELK or simply Elastic). In fact, worldwide cyber-intelligence sites such as Cisco Talos use this platform.

Starting February 2021, the upstream versions of Elasticsearch and Kibana have experienced changes in the license [1]. The Elastic license limits how these components can be used. As a consequence of this, a new project called OpenSearch [2] has forked from Elasticsearch and Kibana. OpenSearch is maintained by the community and released under the Apache License, and is supported by organizations such as Amazon, Red Hat, SAP, and others. Although the Elastic Stack was initially used in ProTego, it was later substituted by OpenSearch because of this new licensing model. Nevertheless, what is explained in this document equally applies to both platforms.

The architecture of the Elastic Stack is the following:

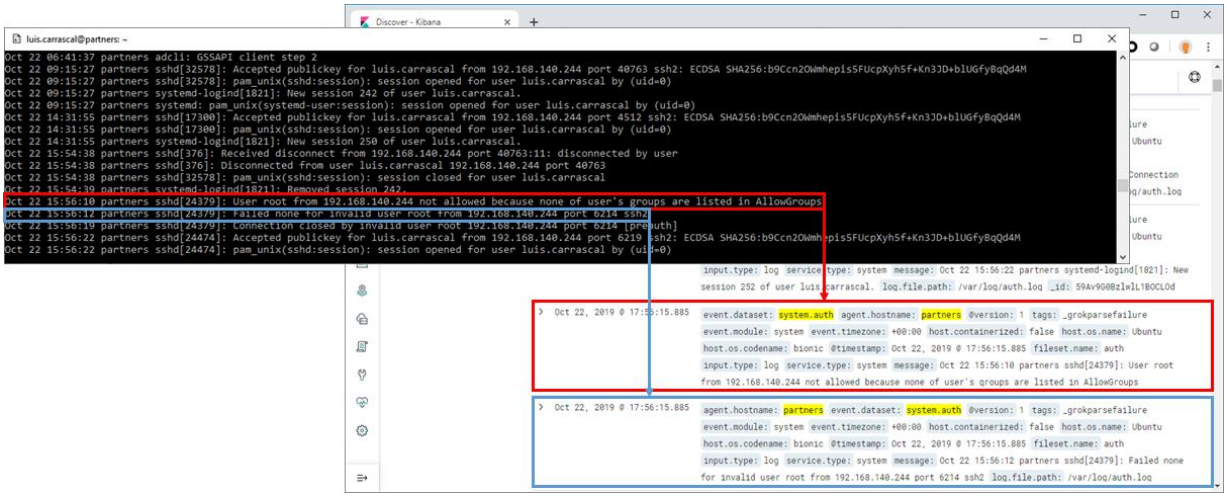


The main components that make up the Elastic Stack are the following:

- **Beats**: this is the component responsible for data ingest. These are light-weight agents that ship data to other components in the Elastic Stack. Beats are specialized in just one specific type of data:
    - *Filebeat*: it monitors log files or locations and collect log events. It is one of the main Beats, since logs are a main data source.
    - *Packetbeat*: it works by capturing network traffic between different assets in the infrastructure and decoding the various protocols.
    - *Winlogbeat*: it reads Windows event logs and filters events.
    - *Metricbeat*: it collects metrics and statistics from the operating system and services running.
    - *Heartbeat*: if periodically checks the status of services to determine whether they are available.
- **Logstash**: this component is a data collection engine with real-time pipelining capabilities. It can dynamically unify data from disparate data sources and normalize data. Any type of event can be enriched and transformed with a broad array of input, filter, and output plugins, simplifying the ingestion process. LogStash can clean and transform data with many aggregations, mutations, along with pattern-matching, geo mapping, and dynamic lookup capabilities.
- **Elasticsearch**: this component is at the heart of the Elastic Stack. It is a distributed search and analytics engine. Logstash and Beats facilitate collecting, aggregating, and enriching your data and storing it in Elasticsearch. It is where the indexing, search, and analysis magic happen. Elasticsearch provides real-time search and analytics for all types of data. In the OpenSearch project this is the component that is called OpenSearch in itself.
- **Kibana**: this component provides an analytics and visualization platform. Kibana is used to search, view, and interact with data stored in Elasticsearch indices. It makes it easy to understand large volumes of data. Its simple, browser-based interface enables to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time. In the OpenSearch project this component is called OpenSearch Dashboards.

Testing Experiments

After setting up the different components in the stack and configure beats in some systems to be monitored, data starts to be collected. As expected, log events are processed and stored:



Nevertheless, there are loads of information stored and needs to be refined. For example, after logging in to the system of a particular Windows virtual machine five different events are generated that contain the account name of the logged user:

> Oct 22, 2019 @ 18:23:33.357 message: The computer attempted to validate the credentials for an account. Authentication Package: MICROSOFT_AUTHENTICATION_PACKAGE_V1_0 Logon Account: luis.carrascal Source Workstation: Alzheimer Error Code: 0x0 @version: 1 winlog.process.thread.id: 2,816 winlog.event_data.TargetUserName: luis.carrascal winlog.task: Credential Validation winlog.channel: Security winlog.computer_name: Alzheimer agent.type: winlogbeat agent.hostname: Alzheimer event.action: Credential Validation event.created: Oct 22,

> Oct 22, 2019 @ 18:23:33.357 message: An account was successfully logged on. Subject: Security ID: S-1-5-18 Account Name: Alzheimer$ Account Domain: WORKGROUP Logon ID: 0x3E7 Logon Type: 10 Impersonation Level: Impersonation New Logon: Security ID: S-1-5-21-2349389593-285953731-3326121726-1001 Account Name: luis.carrascal Account Domain: Alzheimer Logon ID: 0x5256E622 Logon GUID: {00000000-0000-0000-0000-000000000000} Process Information: Process ID: 0xa0c Process Name: C:\Windows\System32\winlogon.exe Network Information: Workstation Name: Alzheimer

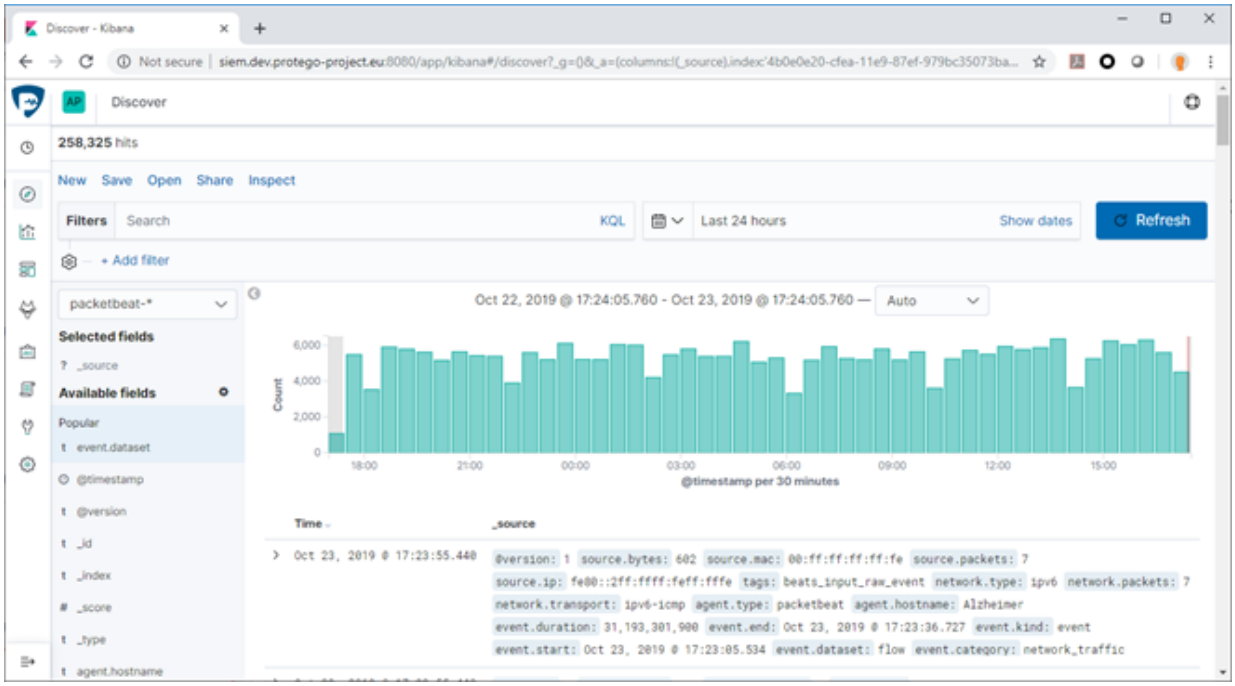
> Oct 22, 2019 @ 18:23:33.357 message: Special privileges assigned to new logon. Subject: Security ID: S-1-5-21-2349389593-285953731-3326121726-1001 Account Name: luis.carrascal Account Domain: Alzheimer Logon ID: 0x5256E622 Privileges: SeSecurityPrivilege SeTakeOwnershipPrivilege SeLoadDriverPrivilege SeBackupPrivilege SeRestorePrivilege SeDebugPrivilege SeSystemEnvironmentPrivilege SeImpersonatePrivilege @version: 1 log.level: information event.action: Special Logon event.created: Oct 22, 2019 @ 18:23:34.557 agent.type: winlogbeat

> Oct 22, 2019 @ 18:23:33.357 message: A logon was attempted using explicit credentials. Subject: Security ID: S-1-5-18 Account Name: Alzheimer$ Account Domain: WORKGROUP Logon ID: 0x3E7 Logon GUID: {00000000-0000-0000-0000-000000000000} Account Whose Credentials Were Used: Account Name: luis.carrascal Account Domain: Alzheimer Logon GUID: {00000000-0000-0000-0000-000000000000} Target Server: Target Server Name: localhost Additional Information: localhost Process Information: Process ID: 0xa0c Process Name: C:\Windows\System32\winlogon.exe Network

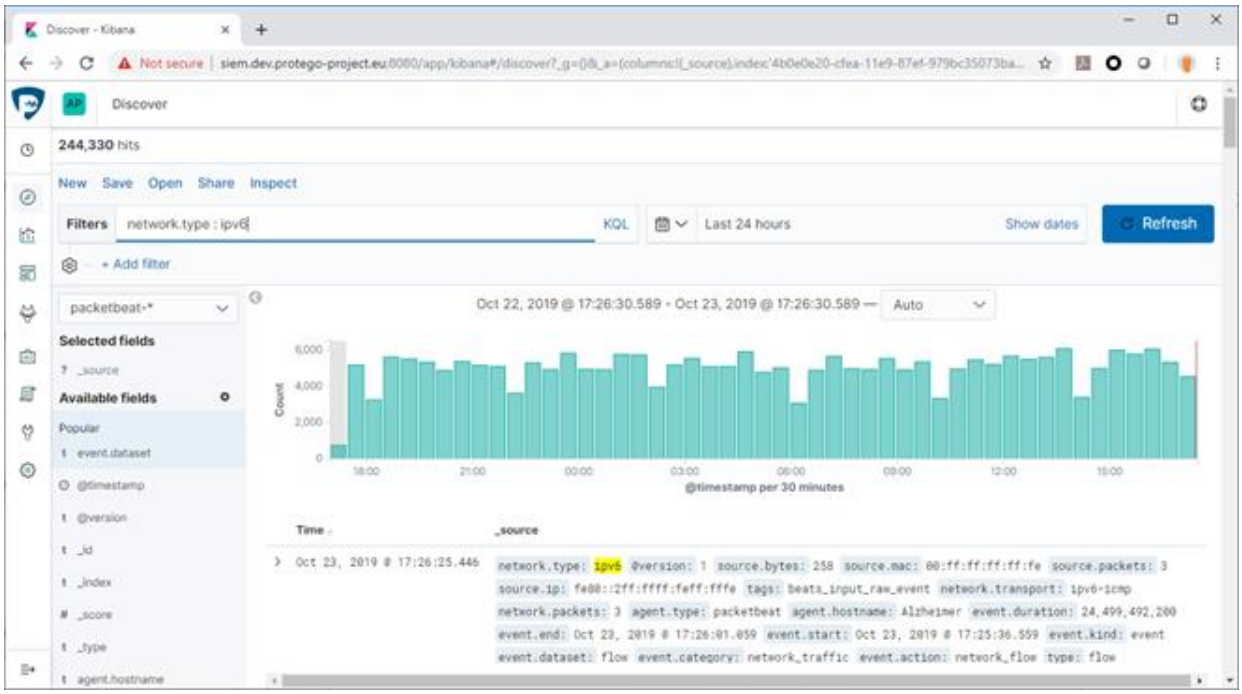
> Oct 22, 2019 @ 18:23:33.357 message: An account was successfully logged on. Subject: Security ID: S-1-5-18 Account Name: Alzheimer$ Account Domain: WORKGROUP Logon ID: 0x3E7 Logon Type: 10 Impersonation Level: Impersonation New Logon: Security ID: S-1-5-21-2349389593-285953731-3326121726-1001 Account Name: luis.carrascal Account Domain: Alzheimer Logon ID: 0x5256E63C Logon GUID: {00000000-0000-0000-0000-000000000000} Process Information: Process ID: 0xa0c Process Name: C:\Windows\System32\winlogon.exe Network Information: Workstation Name: Alzheimer

But in addition, at least other five events are generated for the same logging in that do not contain the account name of the user.

When analyzing the captured packets of network traffic for 24 hours in the same virtual machine the statistics showed a high number of packets for a testing environment (258,325 packets):

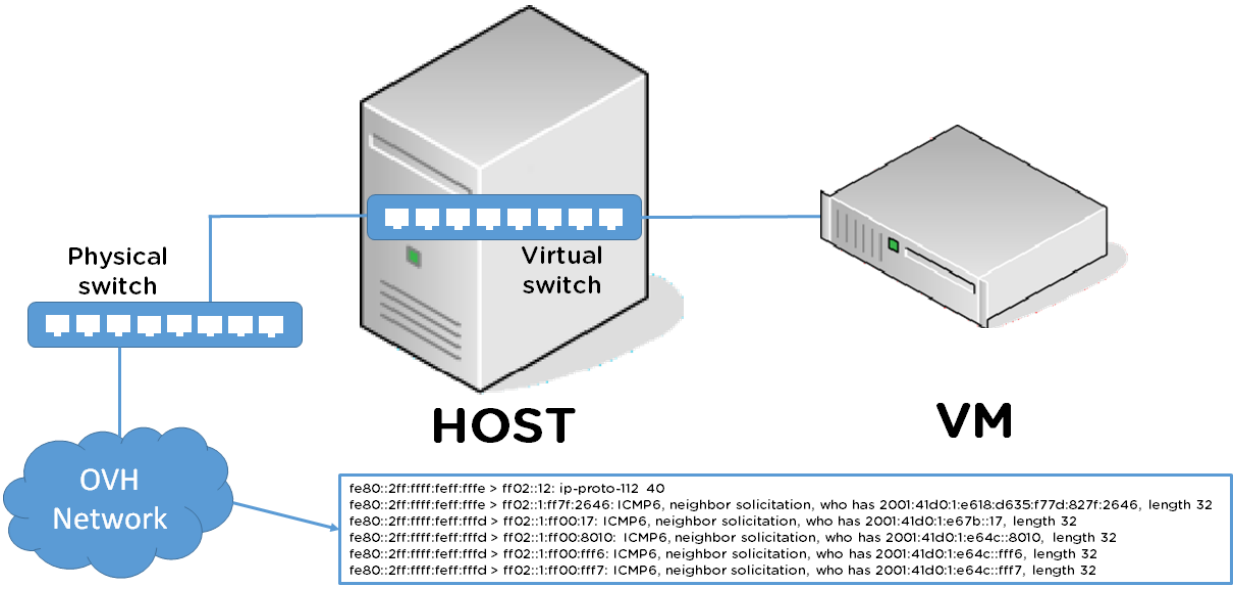


Once the packets were reviewed, it was easy to see high IPv6 traffic, 244,330 packets, nearly 95% of all the traffic:



IPv6 is not used in the whole testing environment and therefore it could be a misconfiguration of the virtual machine. But after checking the settings, they were correct. It could also be a misconfiguration in the virtualization environment that was hosting the virtual machine, but it was correct too. A deeper

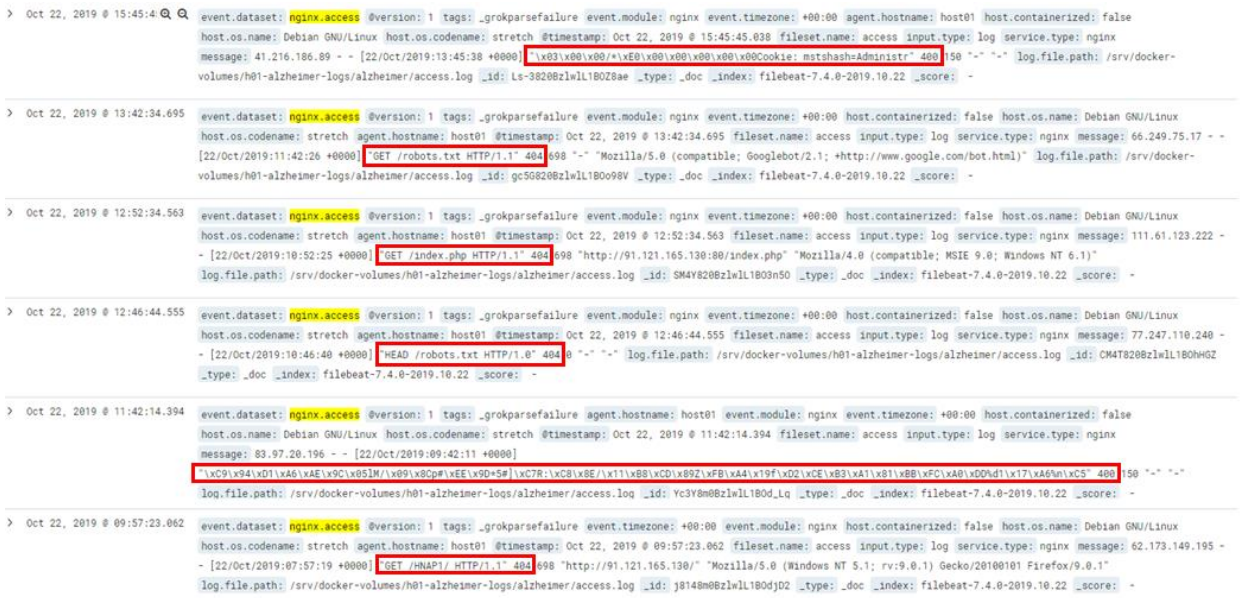
analysis showed that IPv6 traffic was not part of the testing environment itself, but it was broadcast traffic in OVH network, the Internet Service Provider where the physical servers were hosted:



These examples show that when deployed on a Production environment it is necessary to perform a data ingest for some time that helps to establish a baseline of events. With that baseline, all collected data needs to be carefully analyzed, and along with a knowledge of the infrastructures where the SIEM is deployed, it is essential to perform a fine tuning. Otherwise, there will be a flood of events that will most likely be ignored by a human analyst.

Another virtual machine that was part of the testing environment is a Linux system used as a reverse proxy with Nginx for an ASP.NET application that runs on the Windows virtual machine. This Linux system also runs a Bind DNS server. These two services in the Linux virtual machine are open to the Internet.

The access log generated by the Nginx reverse proxy was analyzed as well. This analysis showed the reality of what it involves exposing a service on the Internet nowadays, even without openly publishing it or registering it in public search engines:



Most of the events are illegitimate traffic. Some unauthorized users scan blocks of IP addresses to find open ports, and once they identify the service exposed, they try to hack into the system using different methods. The service exposed by Internet Information Server on the Windows virtual machine uses an ASP.NET web service. Therefore, the events that request the robots.txt file or any PHP file are illegitimate. The events where the requests are a sequence of binary codes expressed in hexadecimal notation are even a clearer example. Considering the service exposed, any request that returns a HTTP code 400 Bad Request, or 404 Not Found, are evidently illegitimate because the application in client devices uses a predefined API where those codes are not an option.

Something similar happens when syslog events for the Bind DNS service are analyzed:

Requests that are refused, denied, or fail because of an unexpected end of input clearly show illegitimate events.

Conclusion

What has been shown in this document is just scratching the surface, since the possibilities the Elastic Stack or the OpenSearch project give to create a Security Information and Event Management platform are immense. But these experiments demonstrate how this platform can collect multiple types of logs and other data, that can be visualized later to help investigate security incidents.

References

[1] Elastic Licensing changes https://www.elastic.co/blog/licensing-change.

[2] OpenSearch Project Introduction https://aws.amazon.com/es/blogs/opensource/introducing-opensearch/