University of Piraeus

School of Information and Communication Technologies

Department of Digital Systems

Postgraduate Program of Studies

MSc Digital Systems Security

Digital Systems Security

Unsupervised AI-based supply chain attack detection

Supervisor Professor: Christos Xenakis

Antonios Mavrelos antonis.mavrelos@ssl-unipi.gr Student ID:MTE2114

Piraeus

2023

1 Executive Summary

A supply chain is the combination of the ecosystem of resources needed to design, manufacture, and distribute a product. In cybersecurity, a supply chain includes hardware, software, cloud or local storage and distribution mechanisms.

The supply chain attack represents a potentially a hidden part of the attack surface and zero trust (100% effective zero trust would eliminate the threat). But the supply chain must be known and understood before it can be remediated. Why attack a single target when successful manipulation of the supply chain can get access to dozens or even hundreds of targets simultaneously? The danger and effectiveness of such attacks is amply illustrated by the SolarWinds, log4j, Spring4Shell, Kaseya, and OpenSSL incidents. At the same time, it is very easy to hide the true target by implementing a massive campaign.

Software supply-chain attacks are not a new development and security experts have been warning for many years that they are some of the hardest types of threats to prevent because they take advantage of trust relationships between vendors and customers and machine-to-machine communication channels, such as software update mechanisms that are inherently trusted by users.

The European Union Agency for Cybersecurity mapping on emerging supply chain attacks finds 66% of attacks focus on the supplier's code (Cordey, 2023) and for a good reason. The report reveals that **an organization could be vulnerable to a supply chain attack even when its own defenses are quite good.** The attackers explore new potential highways to infiltrate organizations by targeting their suppliers. Moreover, with the almost limitless potential of the impact of supply chain attacks on numerous customers, these types of attacks are becoming increasingly common.

AI is already a game-changer in Cybersecurity by advancing speed accuracy and precision in defenses. The reality is that a finite number of people in cybersecurity today take on infinite cyber threats. According to an IBM study (IBM, 2022), defenders are outnumbered. 68% of responders to cybersecurity incidents say it's common to respond to multiple incidents at the same time. There's also more data flowing through an enterprise than ever before—and that enterprise is increasingly complex. Edge computing, internet of things, and remote needs are transforming modern business architectures, creating mazes with significant blind spots for security teams. And if these teams can't "see," then they can't be precise in their security actions.

Table of Content

2 Acronyms & Abbreviations

Term	Description
SSCA	Software supply chain attacks
ICT	Information and Communications Technology
NHS	National Health System
OCI	Open Cloud Initiative
SBOM	Software Bill of Materials.
NPM	Node Package Manager

3 Introduction to supply chain attack

In the U.S. Department of Defense's Defense Federal Acquisition Regulation Supplement, a supply chain risk is defined as "the risk that an adversary may sabotage, maliciously introduce unwanted function, or otherwise subvert the design, integrity, manufacturing, production, distribution, installation, operation, or maintenance of a covered system so as to surveil, deny, disrupt, or otherwise degrade the function, use, or operation of such system."

The growing number of security incidents due to supply chain attacks, further increases the impact of such attacks to the smooth business operation and, most importantly, the trust in and, consequently, the viability of the vendors (incl. hardware, software, infrastructure) providing their ICT solutions as part of the supply chain.

In fact, supply chain attacks are now in the spotlight in the cyber security domain, as they are among the top causes of breaches. Gartner predicts that by 2025, almost half (45%) of the organizations worldwide will have faced attacks on their digital supply chains (Rimol, 2022). Furthermore, the average supply chain compromise lifecycle is higher than the global average, with the average time to react and contain a supply chain attack being 9% more than the overall average of security incidents.

In January 2020, the FBI launched an alert about a supply chain attack which involved the Kwampirs Remote Access Trojan (SANS edu, 2020). This attack targeted several industries, including healthcare, energy, and engineering worldwide. Notice was given by FBI on the healthcare sector, which was significantly affected, from local hospital organizations to major transnational healthcare companies. Their assessment concluded that it constituted a supply chain attack, with the perpetrators gaining access to numerous healthcare entities via vendor software supply chain and hardware products. The latter included products utilized for the management industrial control system (ICS) assets in hospitals.

Targeting an organization's supply chain can significantly amplify the impact of attacks such as ransomware. In August 2022, Advanced, a software and services provider being part of the NHS digital supply chain, was hit by a ransomware attack (Milmo, 2022). The attack affected many services relying on Advanced's system across the NHS, including patient referrals, ambulance dispatch, out-of-hours appointment bookings, mental health services and emergency prescriptions. Along with the fears for the patient data being leaked and/or modified, several services remained offline causing major disruptions in NHS operations, but also piling up huge amount of work for the following months due to the shifting to manual processes.

3.1 Software

Adversaries may manipulate products or product delivery mechanisms prior to receipt by a final consumer for the purpose of data or system compromise.
Supply chain compromise can take place at any stage of the supply chain including:

- Manipulation of development tools
- Manipulation of a development environment
- Manipulation of source code repositories (public or private)

- Manipulation of source code in open-source dependencies
- Manipulation of software update/distribution mechanisms
- Compromised/infected system images (multiple cases of removable media infected at the factory) (McGill, Hardware Trojans and Supply Lines, 2021)
- Replacement of legitimate software with modified versions
- Sales of modified/counterfeit products to legitimate distributors
- Shipment interdiction

While supply chain compromise can impact any component of hardware or software, adversaries looking to gain execution have often focused on malicious additions to legitimate software in software distribution or update channels (Support, 2020).

The most famous cases of supply chain attacks are the ones on Solarwinds' Orion platform and log4j a very common logging frameworks for Java.

Solarwind's case is an Advanced Persistent Threat (APT), attack campaigns that the intruder, or team of intruders establishes and illicit, long-term presence in the company network. For almost a year the attackers had access to Solarwind's systems, altering the code of Orion (Orion Platform, n.d.), an infrastructure monitoring and management platform a product that is used on almost all critical infrastructure sectors in US. Telcoms, accounting firms, all branches of US Military, the Pentagon, The Department of Homeland Security, the National Institute of Health, the State Department, hundreds of universities, and colleges, 425 of US Fortune 500 companies were compromised. By downloading the altered version of Orion from Solarwind's servers, a company/organization without knowing was installing a backdoor into their systems.

What is even more interesting is that due to the nature of supply chain attack the attackers had access to other core business components that are used even more widely than Orion. VMWare a company specialized in virtualization technologies was one of Orion clients that had its systems breached, along with Microsoft. On both cases Orion was weaponized, with backdoor activated. The products of both companies are used almost everywhere in the enterprise, fortunately VMWare or Microsoft didn't report any tampering on any of their products.

In business world it is almost no-brainer to trust this kind of 3rd party companies that are used on military sector, and the key word is trust. Companies trust, other companies, their products, and solutions because most of the complexity and the time that is needed for full regression tests beyond functionality and compatibility.

And this was proven with the log4j attack (Log4Shell), the most serious software vulnerability (CISA director says the LOG4J security flaw is the "most serious" she's seen in her career, n.d.). The vulnerability was introduced in 2013 but was noticed in 2021, after 8 years, and this is one of the most used components in Enterprise world. With a CVSS score of 10.0 (CVE-2021-44228, 2021), having a low attack complexity, with no privileges or user interaction required, it was a nightmare. (Yoran, 2021). Log4Shell was pervasive across all industries and geographies. One in 10 corporate servers being exposed, one in 10 web applications and so on. One in 10 of nearly every aspect of our digital infrastructure had the potential for malicious exploitation via Log4Shell and that is because of the sear number of Java applications that use log4j framework.

Because open-source software serves as the foundation of every software supply chain and most modern digital infrastructures, Log4Shell has quickly become a widespread and far-reaching issue that will impact businesses of all shapes and sizes for years to come.

Supply chain software is particularly fragile as vendors and partners become more interconnected. Attacks on open-source aspects of the supply chain skyrocketed by 650 percent in 2021 (State of the Software Supply Chain, 2021). More open-source-based vulnerabilities will likely emerge if businesses don't recognize their responsibility to help bolster protection. Perhaps most worryingly, four in 10 businesses (The 2022 State of Open Source Security Report, 2022) are not confident in their open-source software security, while the time taken to remediate open-source vulnerabilities has more than doubled from 49 days in 2018 to 110 days in 2021.

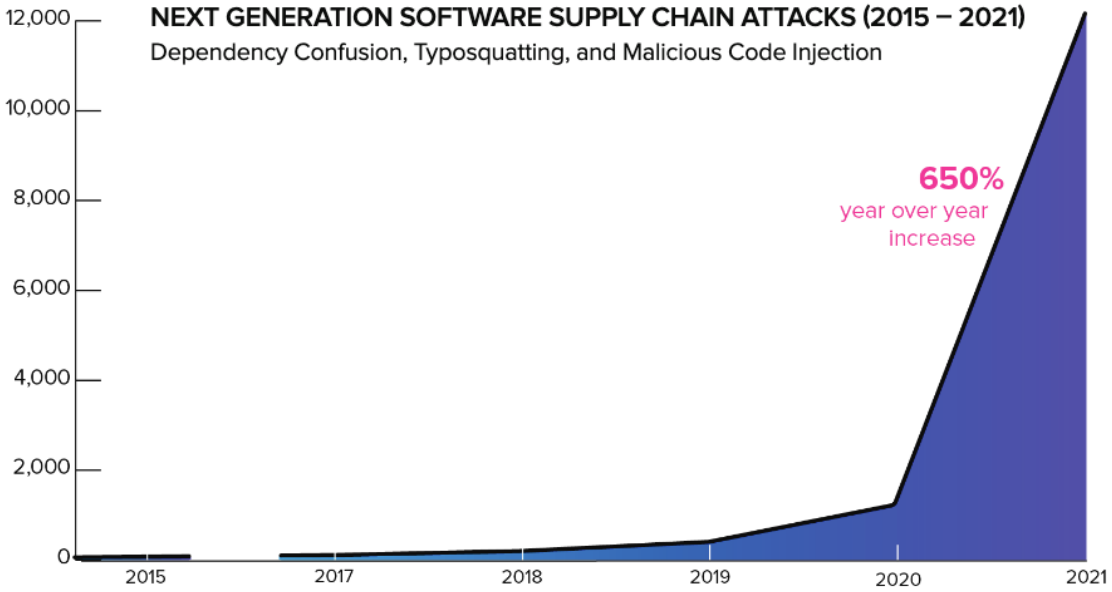


Figure 1 Next generation software supply chain attacks

3.2 Hardware

Software is hard, but not as hard as hardware. Supply chain attack in hardware is not an easy path for bad actors. They compromise hardware by inserting physical implants into a product component or by modifying firmware. Often these manipulations create a backdoor connection between the device and external computers that the attacker controls. Once the device reaches its destination, adversaries use the backdoor to gain further access or exfiltrate data. But first they must get their hands on the hardware. Unlike software attacks, tampering with hardware requires physical contact with the component or device and there are two known methods: interdiction and seeding.

In interdiction (McGill, Hardware Trojans and Supply Lines, 2021), saboteurs intercept the hardware while it's on route to the next factory in the production line. They unpackage and modify the hardware in a secure location. Then they repackage it and get it back in transit to the final location. They need to move quickly, as delays in shipping may trigger red flags.

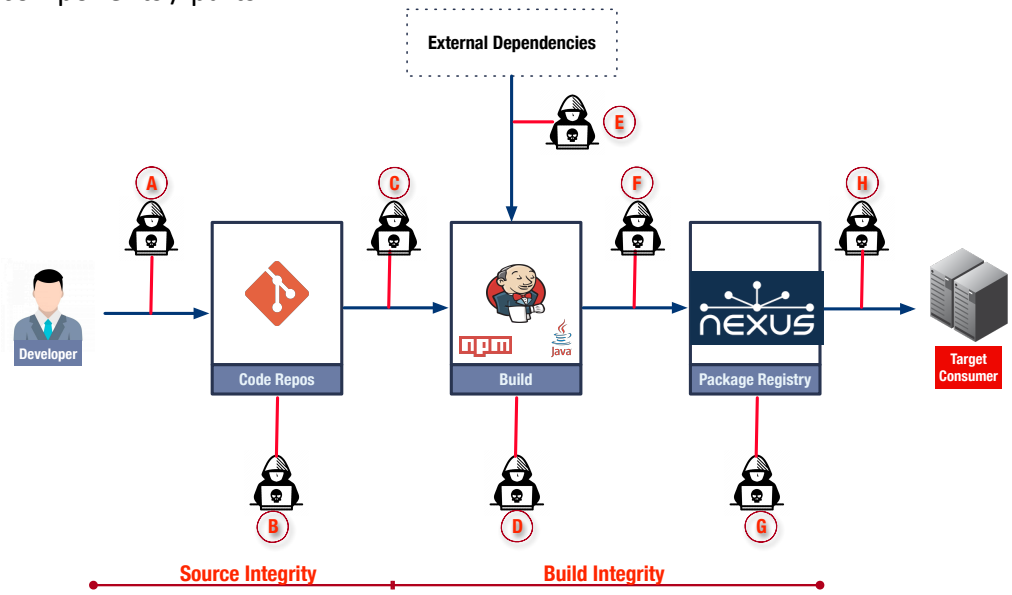
As hard as interdiction is, it's not nearly as challenging as seeding. Seeding attacks involve the manipulation of the hardware on the factory floor. To infiltrate a target factory, attackers may pose as government officials or resort to old fashioned bribery or threats to convince an insider to act, or to allow the attacker direct access to the hardware.

4 Software build pipeline Attack Surface

The attack surface in a software supply chain is vast. Today's cybercriminal strategies target every link in an attack chain, from gathering information and gaining access, to moving laterally across the network to discover resources to target, to evading detection while exfiltrating data. Traditional security strategies, however, tend to only focus on a handful of attack components, which gives criminals a significant advantage.

To address those challenges, security teams use a combination of tools, strategy, automation, and skilled professionals to monitor the entire attack chain and automate as much of the process as possible so that human resources can be focused on higher order analysis and response. Choosing such tools, however, requires understanding the entire length of the attack chain and how vulnerabilities in each of its links can compromise the security of your network.

A very common setup for software development consists of the following components / parts:



4.1 Developer

While cybersecurity advances have hardened IT infrastructure and made it increasingly difficult to hack systems remotely, criminals have a logical way around these measures: targeting the employees who are already inside the systems. Human factor it the first and by far the weakest link in the supply chain.

Developers, DevOps, System administrators can be company/organization employees, or they can be of 3rd party. They may work from inside the premises or remote via VPN in any case there is an overhead to support them. VPN, Zoning, Identity Management, access rules for the firewalls, and source code repositories. Management of who can push code on each repository, who maintains each repository, who receives the push requests, who can view what, and who is the owner of each repository. Role Base Access Controls (RBAC) must be in place implemented with the least privileges principle (PoLP) (Principle of least privilege, n.d.) along with physical security. What equipment will they use (company, personal), what OS and

software is installed, what patch level, are there domain security policies in place or it is locally managed, what endpoint protection and safeguards are implemented to limit or contain the impact of a potential cybersecurity incident or attack. Each one of these questions are security projects that need to be carefully designed, implemented, and assessed.

But most important. Are they trained and assessed for the role they have in security fundamentals/awareness and how often. Social engineering attacks represent a continuing threat to employees of organizations. Stealing credentials, or access tokens from a user is one of the most common ways the attacker gets access into an organization. One high profile attack was the one on twitter (Support, 2020) where attackers used spear phishing (Phishing Scams, n.d.) on employees to get the needed credentials.

4.2 Source code

Developers use version control systems, to manage their code, track issues and changes, collaborate with other developers and store it. GIT is the de-facto standard for version control system and Microsoft's GitHub is the de-facto standard solution for version control with GUI, issue tracking, team collaboration, and CI/CD. Almost all opensource projects and most of commercials that do not use on-premises services are using GitHub.

Source code must not be decoupled from the final artifact. While most npm packages are open source, until now there was no guarantee that a package on npm is built from the same source code that's published. Linux Foundation (Linux Foundation, n.d.) created Sigstore (Sigstore, n.d.), a combination of technologies to handle signing, verification and provenance checks that enables developers to sign off on what they build, without needing to jump through hoops or know tricky security protocols, and it's a way for anyone using those releases to verify the signatures against a tamper-proof log. GitHub now supports Sigstore but it's up to the developers if they will use it.

Sigstore can also help in the case of indirect attacks like Octopus (Munoz, 2020). Back in 2018 Github Security Incident Response Team (SIRT) (Github Security, n.d.) received an initial notification about a set of repositories serving, a malware (Octopus) that is capable of identifying Netbeans project files and embed malicious payload both in projects and build JAR files. Anyone cloning the infected repo, building, or using the JAR files would be infected. In this case Sigstore can help traceback to the git hash that was used to build the code, it's much easier to find the malware if you know where and when (in time) to look.

4.3 Third-party libraries/code

Also known as "Compromised dependencies" is among the highest risk in the software supply chain. Modern software often relies on numerous third-party libraries and dependencies. If any of these dependencies have vulnerabilities or are compromised, they can introduce security risks into the software supply chain.

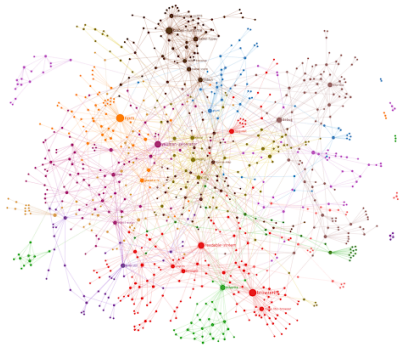
In June 2021, malicious crypto mining code was detected in multiple Python Package Index (PyPI) packages (Polkovnychenko, 2022).

In March 2021, at least 30 malicious docker images (with a collective 20 million download volume) in Docker Hub were used to spread crypto mining malware accounting for crypto jacking operations worth US$200,000 (Sasson, 2021)

4.4 Build System/Artifact Poisoning

Codecov (Codecov, n.d.) is a tool doing code coverage analysis. It lives in the build system and is running on testing stage making sure the software has maximum coverage. In 2021 an attacker managed to get access (Ilascu, 2021) to Codecov platform and using automation collected credentials of all of Codecov clients, who at that point had around 29000 accounts, customers like RAPID, Hashicorp and IBM. Hashicorp has one of the biggest SaaS solutions (Vault, n.d.) for secret, password, and certificate storage. The attacker managed to get Hashicorp private GPG signing key used to sign and verify software releases. They quickly remedy the situation by rotating the key and revoking the previous one but there was a window that the attacker could sign his code as Hashicorp. So, there is one vendor (Codecov) that was breeched, resulting in another vendor (Hashicorp) getting a backdoor which could result to any other company or user using their services to get a backdoor.

A new kind of attack is dependency confusion, Threat actors are uploading malware to open-source repositories such as PyPI, NPM, RubyGems and naming them such that they would be downloaded and used by the target company's application. Ethical hackers used this technique on PyTorch, one of the most significant Libraries for ML. It is used widly for research projects but also for production purposes. ChatCPT is using PyTorch and whole other libraries have dependencies on Pytorch.



Εικόνα 1 Dependency tree of angular framework

4.5 Deploy and Delivery

The final artifact(s) will be deployed on premises or on cloud native topology and can have many forms, a OCI Container (Open Containers, n.d.), a native executable for an OS, a war/jar that will be deployed inside an application server container, even a library that will be uploaded to a public repository and be included in package/library dependency system like maven (Maven, n.d.).

-A. Human factor (ex. Developer): Malicious actor using methods like spear phishing acquires a developer's credentials/access tokens to the code repo. Given the right permissions he can alter or introduce code into the system.

-B. Third Party Libraries: Developers are including vulnerable dependencies into their code.

-C. Malicious actor bypasses the official git repo and builds a different code that is then pushed into the artifact registry or production

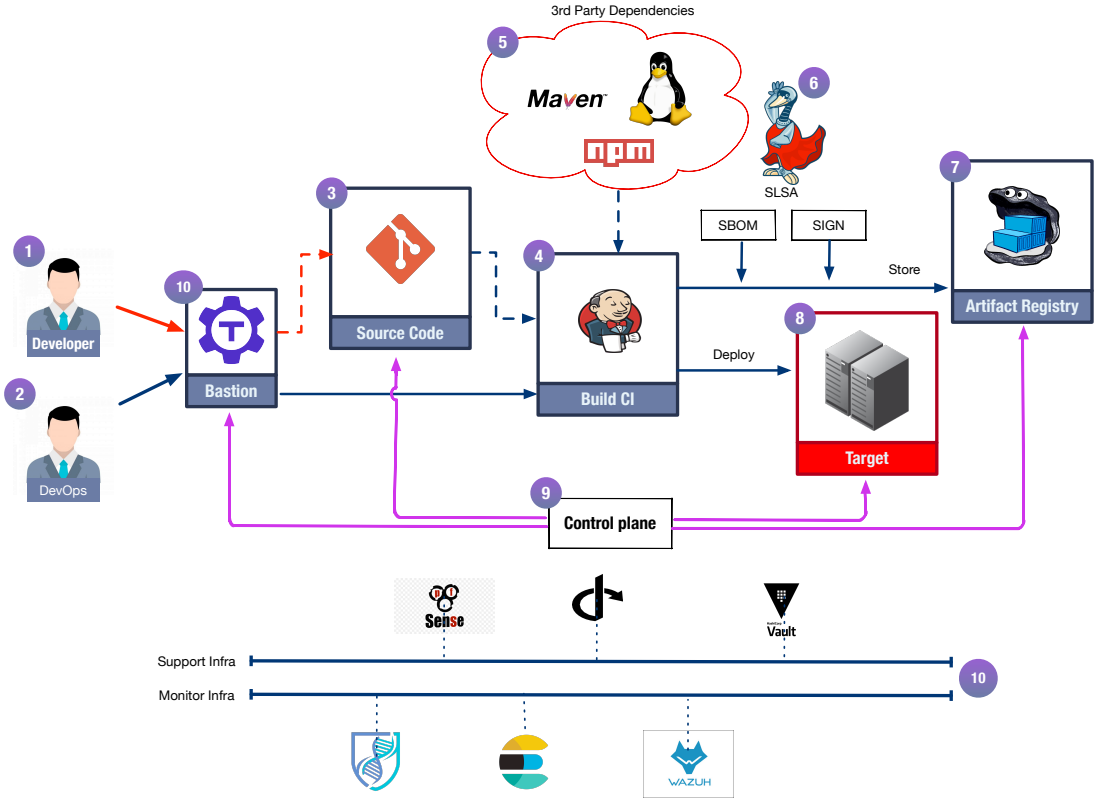
-D. Malicious actor alters or replaces CI/CD Actions/Pipelines.
-E. Malicious actor is attacking the system through external libraries that he knows the system is using (based on).
-F, G. Malicious actor is poisoning the artifact registry, replacing the original artifact with a compromised one, using CI/CD's access tokens or taking advantage misconfiguration of registry software.
H. Malicious actor is having direct access to production or delivery systems.

The above components are described in a high-level abstraction of a software development setup. Depending on the type or significance of the product, the security awareness of the organization(s) or company(ies) responsible for the software, the acceptable risk, there can be many different setups supporting the development pipeline. Develop->Store Code->Build->Package->Deliver.

5 Hardening a fragile chain

Since the human factor can't be eliminated from the pipeline, it can be minimized by using already established technologies, as well as new ones. For example, the operational tasks like DevOps (Sign, Build, Deploy) could be automated and minimize the human intervention. Developers, SysOps, DevOps, SecOps could be authenticated using hardware keys, OIDC SSO (OpenID, n.d.), Zero trust (Zero Trust Security, n.d.) networking and many other technologies that prevent unauthorized intruders. A control plane can be introduced to control automations like:

- Block IP or Users in real time in bastion gateway or to all infrastructure after an event is elevated to incident.
- Prevent access to source code repositories, disable of build servers or shutdown production servers when SLSA reports that there is a vulnerable library included in a SBOM that is currently deployed to production, or when the signature of the running artifact is different than the one the build server reported on built.



Εικόνα 2 An example of a hardened software build topoloigy

New technologies like AI-based ones can be introduced to handle the hardening of every link of the software supply chain. For example:

- The traffic between each component can be feed to a system that can learn and extrapolate information that can control the access of a user, an IP, or a service token. This can be centralized with a syslog server and the inference system can signal the control plane of the Machine Learning findings.

6 Testbed / Scenario

For this thesis a simpler topology will be used with the basic components being:

1. A Developer, that pushes code into a central source code Repository.
2. A malicious actor, that interacts with the source code Repository using stolen developer credentials.
3. The Code Repository (Git)
4. A Build Server (mock)
5. An HAProxy that terminates the SSL communication between each component it's traffic logs will be send to an Elastic DB in JSON format
6. An Inference system.

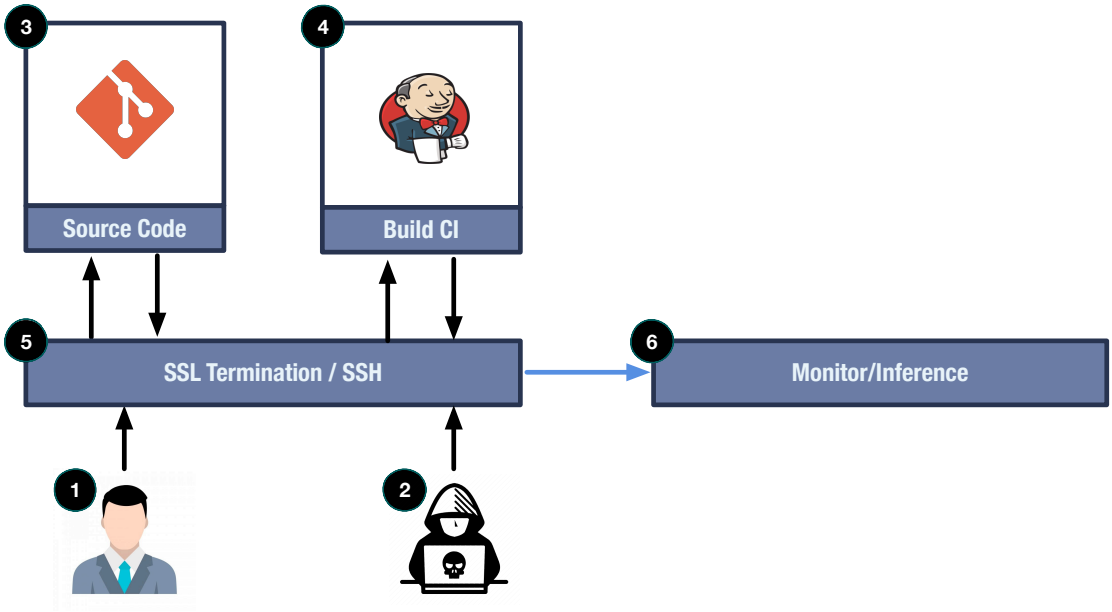


Figure 2 Testbed Components

The baseline scenario consists of a developer pushing code to the repo or triggering a manual build. The network flow (simplified) is:

1. The Developer is pushing to the git through the HAProxy
2. HAProxy is:
    a. Terminating the SSL connection.
    b. Storing the request in traffic log
    c. Sends the request to the Inference System.
    d. Forwards the request to the git server
3. Git Server is:
    a. Validating the developer's access token that is used as authentication mechanism.
    b. If its valid then and the usual git actions are executed
    c. Sending back the outcome of the push command to the developer
4. The AI Inference system is evaluating the data that HAProxy send stores them in the monitoring system for training of the model.

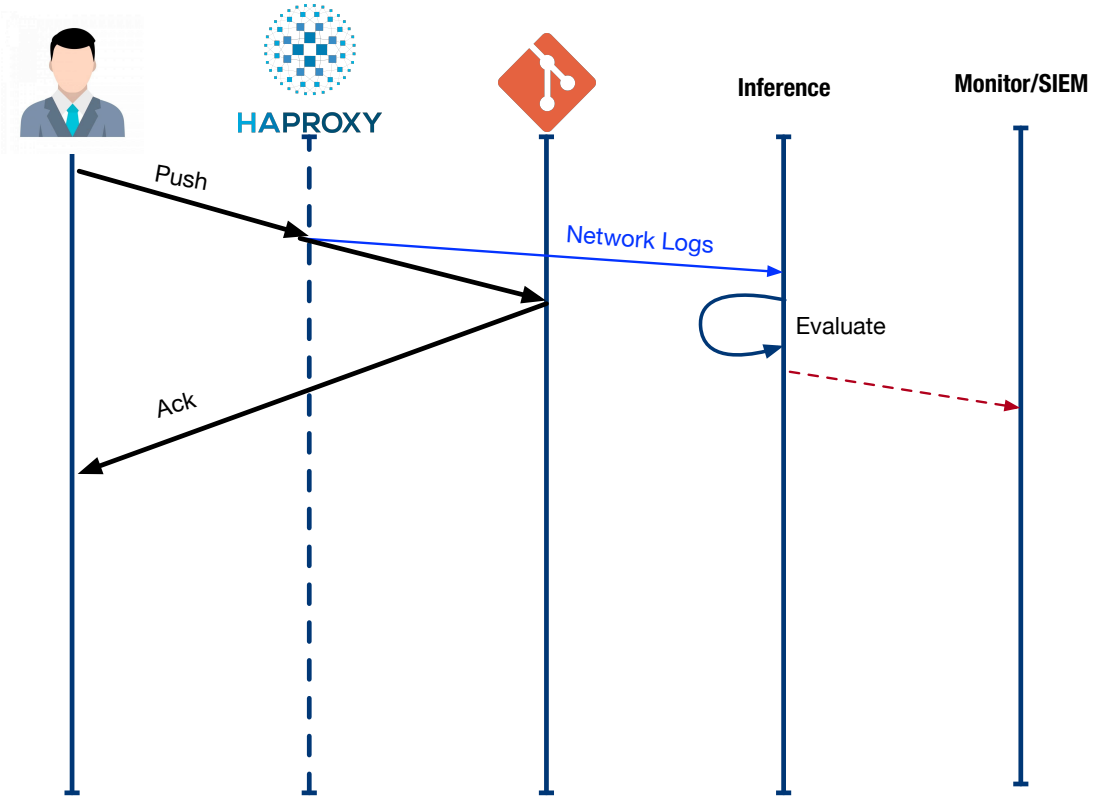


Figure 3 Baseline scenario flow

The attack scenario has the following changes:

1. The attacker is the one that is accessing the source code repository.
2. The AI Inference system is evaluating the data that HAProxy send and after saving them into the monitoring system it triggers the proper actions, using some kind of playbook system (OASIS Collaborative Automated Course of Action Operations (CACAO) for Cyber Security TC, n.d.).

Thus, the expected result is for AI to distinguish a malicious attacker from an authorized developer by "learning" from HAProxy traffic logs what is "normal" and what "is not". For this scenario the malicious actor will access the system from another IP. Other cases could be, another git client from the same IP, or the hour that the push is taking place.

7 AI/Machine Learning

7.1 Introduction

It all starts with AI witch simply means “Human Intelligence exhibit by machine”. An AI is a machine that acts like a human and currently the closest we have to AI is Narrow AI or Weak AI (NarrowAI, n.d.), that is machines can be as good or even better than human at specific tasks. For example, detecting heart disease from images or specific videogames that is trained. The weakness of Narrow AI is that each one is very good at doing one thing well, they can’t be like humans that they have multiple abilities, this is called General AI and it’s a milestone that we haven’t reached yet.

Machine Learning is a subset of AI, an approach to try and achieve AI to systems that can find patterns in a set of data. Stanford University describes Machine Learning as “the science of getting computers to act without being explicitly programmed”.

Deep Learning or Deep Neural Networks is one of the technics implementing machine learning.

In a single sentence Machine Learning is using an algorithm or computer program to learn about different patterns in data and then taking that algorithm and what is learned to make predictions using similar data. Machine Learning algorithms are also called Models.

7.2 Types of Machine Learning

Supervised Learning (Supervised learning, n.d.)is a subset of machine learning where input objects (for example, a vector of predictor variables) and a desired output value (also known as human-labeled supervisory signal) train a model. The training data is processed, building a function that maps new data on expected output values. An optimal scenario will allow for the algorithm to correctly determine output values for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Another “technique” in Supervised Learning is Regression (Seldon, 2021). It helps to investigate the relationship between independent variables of features and a dependent variable or outcome. Algorithms are trained to understand the relationship between independent variables and an outcome or dependent variable. The model can then be leveraged to predict the outcome of new and unseen input data, or to fill a gap in missing data.

Unsupervised Learning (Unsupervised learning, n.d.) is another subset of Machine Learning commonly used when groups or labels in the input data do not exist. The logic behind Unsupervised Learning is to let the machine create the categories and the clustering of the data.

Reinforcement Learning is a subset of Machine Learning that teaches the machine through trial and error. For example, a program learns a videogame by playing it millions of times until it gets the highest score.

7.3 AI in cybersecurity

AI can play a crucial role in enhancing the security of software supply chains by providing intelligent tools and techniques to identify, detect, and mitigate potential

risks and vulnerabilities. Here are several ways in which AI can help in securing software supply chains:

Anomaly detection: AI algorithms can analyze patterns and behaviors within the supply chain to identify anomalies that may indicate potential security threats. By monitoring various data sources, including code repositories, build systems, and deployment pipelines, AI can detect suspicious activities, such as unauthorized access or code modifications.

Code analysis: AI-powered static and dynamic code analysis tools can scan software components for vulnerabilities and identify potential security weaknesses. These tools can automatically review source code, libraries, and dependencies to detect common security flaws, such as injection attacks, buffer overflows, or insecure configurations.

Threat intelligence: AI can leverage vast amounts of data from security feeds, vulnerability databases, and threat intelligence platforms to proactively identify emerging threats and vulnerabilities. By continuously monitoring these sources, AI can provide real-time alerts and insights on potential risks that might affect the software supply chain.

Risk assessment: AI can assess the security posture of software components throughout the supply chain by analyzing factors such as code quality, dependency vulnerabilities, and licensing compliance. This enables organizations to evaluate the potential risks associated with specific software components and make informed decisions about their inclusion or exclusion from the supply chain.

Behavioral analysis: AI algorithms can learn and understand the typical behavior of the software supply chain, including the flow of code, dependencies, and interactions among different components. By monitoring deviations from normal behavior, AI can identify potential security breaches, such as unauthorized code modifications, suspicious build processes, or abnormal deployment patterns.

Automated testing: AI can automate the testing process to identify vulnerabilities and ensure the security of software components. It can generate and execute test cases, simulate attacks, and analyze the system's response to identify weaknesses that could be exploited. This reduces the manual effort required for security testing and improves the overall effectiveness of the process.

Threat hunting: AI can assist in proactive threat hunting by analyzing vast amounts of data and identifying potential indicators of compromise (IOCs) within the software supply chain. It can help security teams detect advanced threats and malware that may be hidden within the code or system components.

Overall, AI can provide organizations with powerful tools to enhance the security of their software supply chains. By leveraging AI techniques, organizations can identify and mitigate vulnerabilities, proactively detect threats, and ensure the integrity and trustworthiness of their software components.

7.4 Decomposing Machine Learning

Machine Learning comes in 3 parts:

7.4.1 Data Collection

The first part is collecting the data that will be used in the second part (Data Modeling) to train and evaluate the model.
In a software supply chain scenario, the data among others may be collected from:

- Network flow, traffic, VPN logs can be provided by Firewall(s)
- IDS/IPS or a superset of these technologies like Suricata can capture the traffic or recognize patterns acting like the first line of defense. The alert from these tools in conjunction with a SIEM can also be a source of data for the model.
- Load Balancers / SSL Terminators log the traffic from higher level, and this is the source of data that will be used in this thesis.
- Build Servers and Version Control Servers logs can also provide valuable information such as who pushed this patch, when, why on what branch/tag and was it deployed on production? For the last part data can be corelated with Web/Application Servers access/error logs.

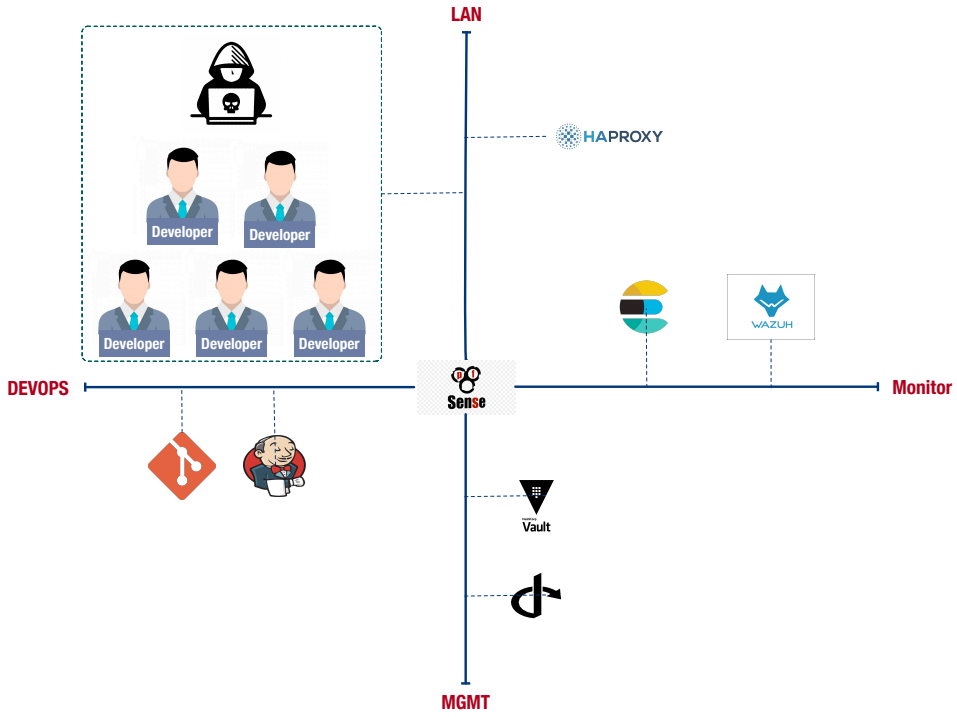


Figure 4 System/Network Topology for log harvesting

HAProxy is a very versatile software, and its logging capabilities can overwhelm the ML modeler in data collection stage. Running as Layer 7 proxy and using the default logging format the output format of an HTTP Request looks like this:

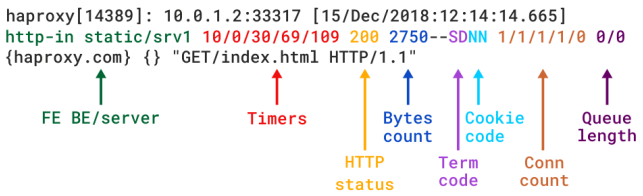


Figure 5 HAProxy log format

This is the format that will also be used in the Testbed but converted to JSON document using Elastic Agent HAProxy module:


```
{'log': {'file': {'path': '/var/log/haproxy/haproxy-traffic.log'},
   'offset': 90841863},
  'source': {'address': '172.16.211.54', 'port': 56456, 'ip': '172.16.211.54'},
  'fileset': {'name': 'log'},
  'url': {'path': '/repo-services.git/info/refs',
   'extension': 'git/info/refs',
   'original': '/repo-services.git/info/refs?service=git-upload-pack',
   'query': 'service=git-upload-pack'},
  'input': {'type': 'log'},
  '@timestamp': '2023-06-21T14:40:03.115+03:00',
  'ecs': {'version': '1.12.0'},
  'related': {'ip': ['172.16.211.54']},
  'service': {'type': 'haproxy'},
  'host': {'hostname': 'ha',
   'os': {'kernel': '5.15.0-73-generic',
    'codename': 'jammy',
    'name': 'Ubuntu',
    'type': 'linux',
    'family': 'debian',
    'version': '22.04.2 LTS (Jammy Jellyfish)',
    'platform': 'ubuntu'},
   'ip': ['172.16.211.2', 'fe80::546f:c6ff:fedd:18'],
   'containerized': False,
   'name': 'ha',
   'id': '842aef7663b249b39038430d6280f6f1',
   'mac': ['51-7F-C6-DA-00-18'],
   'architecture': 'x86_64'},
  'haproxy': {'server_name': 'git',
   'backend_queue': 0,
   'total_waiting_time_ms': 0,
   'termination_state': '----',
   'connection_wait_time_ms': 1,
   'backend_name': 'git_http',
   'http': {'request': {'raw_request_line': 'GET /repo.git/info/refs?service=git-upload-pack HTTP/1.1',
     'captured_cookie': '-',
     'time_wait_ms': 0,
     'time_wait_without_data_ms': 9},
    'response': {'captured_cookie': '-'}},
   'frontend_name': 'admin~',
   'server_queue': 0,
   'bytes_read': 569,
   'connections': {'server': 0,
    'retries': 0,
    'active': 3,
    'backend': 0,
    'frontend': 3}},
  'http': {'request': {'method': 'GET'},
   'response': {'status_code': 401, 'bytes': 569},
   'version': '1.1'},
  'event': {'duration': 10000000,
   'ingested': '2023-06-21T11:40:11.189207746Z',
   'timezone': '+03:00',
   'kind': 'event',
   'module': 'haproxy',
   'category': ['web'],
   'dataset': 'haproxy.log',
   'outcome': 'failure'}}}
```


Εικόνα 3 HAProxy Log entry

7.4.2 Data Modeling

After data has been collected, they must be manipulated and processed. Pandas (Pandas, n.d.) is a python library created for this purpose and is heavily used on most ML projects.

The first step is to distinguish the information that will be used and create an Excel like Table called DataFrame (Dataframe, n.d.):

	datetime	ip	request	method	status_code
0	2023-06-21 14:40:03	172.16.211.2	GET /repo.git/info/refs...	GET	401
1	2023-06-21 14:40:03	172.16.211.2	GET /repo.git/info/refs...	GET	401
2	2023-06-21 14:40:03	172.16.211.2	GET /repo.git/info/refs...	GET	401
3	2023-06-21 14:40:03	172.16.211.2	GET /repo.git/info/refs...	GET	200
4	2023-06-21 14:40:03	172.16.211.2	GET /repo.git/info/refs...	GET	200

Figure 6 Dataframe

The following code process a row and it's called for every HTTP request HAProxy send.

```
def get_important_elements(path):
    f = open(path, 'r')
    json_file = json.load(f)

    datetime_str = json_file['_source']['@timestamp'].split('.')[0]#'09/19/22 13:55:26'
    datetime_object = datetime.strptime(datetime_str, '%Y-%m-%dT%H:%M:%S')

    return pd.DataFrame({'datetime': datetime_object,
             'ip': json_file['_source']['host']['ip'][0],
             'request': json_file['_source']['haproxy']['http']['request']['raw_request_line'],
             'method': json_file['_source']['http']['request']['method'],
             'status_code': json_file['_source']['http']['response']['status_code']}, index=[0])
```

Figure 7 Create a dataframe row

The IP octets are split into numbers. Same the timestamp. The day of the week and the hour can be extrapolated from the timestamp part. Also, the HTTP method is enumerated in number format.

```
ipparts = all_logs_df.ip.str.split(pat='.',expand=True)
ipparts.columns = ['ip1', 'ip2', 'ip3', 'ip4']
ll_logs_df['hour']=all_logs_df.datetime.dt.hour.values
all_logs_df['dayofweek']=all_logs_df.datetime.dt.dayofweek.values
```

Figure 8 IP split octets into numbers

The final DataFrame looks like the following table:

	status_code	hour	dayofweek	method_GET	method_POST	ip1	ip2	ip3	ip4
0	401	14	2	1	0	172	16	211	2
1	401	14	2	1	0	172	16	211	2
2	401	14	2	1	0	172	16	211	2
3	200	14	2	1	0	172	16	211	2
4	200	14	2	1	0	172	16	211	2
...	...	...	...	...	...	...	...	...	...
295	200	14	2	1	0	172	16	211	2
296	200	14	2	0	1	172	16	211	2
297	401	14	2	1	0	172	16	211	2
298	200	14	2	1	0	172	16	211	2
299	200	14	2	0	1	172	16	211	2

300 rows × 9 columns

Figure 9 DataFrame

Having the data in the “correct” format the next step is to select an algorithm to classify these data. Since this is unsupervised machine learning the KMeans clustering algorithm is chosen. K-means clustering is an unsupervised machine learning algorithm widely used for partitioning unlabeled datasets into distinct groups or clusters. The primary goal of this technique is to identify underlying patterns and structures within the data by minimizing the sum of squared distances between each data point and its corresponding cluster centroid.

The "K" in K-means signifies the number of clusters that will be formed during model training, highlighting one of its key strengths - users can predetermine the desired number of groups beforehand. For instance, if K=4 then 4 clusters would be created, and if K=7 then 7 clusters would be created. This functionality helps customizing the clustering models based on specific requirements or hypotheses about potential data patterns. However, determining the optimal value for "K" can sometimes present a challenge, particularly when dealing with complex datasets.

The algorithm setup is:

- Two clusters, one for the valid requests and the other for the malicious.
- 300 iterations (this is the default)
- Relative tolerance of 0.0001. this is regards to Frobenius norm of the difference in the cluster centers of two consecutive iterations to declare convergence.
- Random state 42. This is used for centroid initialization. This is just a random number

```
km = KMeans(n_clusters=2, max_iter=300, tol=0.0001, verbose=0, random_state=42, copy_x=True)


km.fit(X)
```

```
KMeans(n_clusters=2, random_state=42)
```

```
km.transform(X)
```

```
array([[  1.98      , 201.00124378],
       [  1.98      , 201.00124378],
       [  1.98      , 201.00124378],
       [201.009752  ,   0.70710678],
       [201.009752  ,   0.70710678],
       [  1.98      , 201.00124378],
       [201.009752  ,   0.70710678],
       [201.01472682,   0.70710678],
       [201.01472682,   0.70710678],
       [201.009752  ,   0.70710678],
       [201.01472682,   0.70710678],
       [201.01472682,   0.70710678],
       [  1.98      , 201.00124378],
       [201.009752  ,   0.70710678],
       [201.01472682,   0.70710678],
       [  1.98      , 201.00124378],
       [  1.98      , 201.00124378],
       [201.009752  ,   0.70710678],
       [  1.98      , 201.00124378],
```

Figure 10 KMeans setup

The distance to the cluster centers is calculated using the fit(X) or transform(X) functions.

```
distances_from_centroids = km.transform(X).min(1)
distances_from_centroids
```

```
array([  1.98      ,   1.98      ,   1.98      ,   0.70710678,
         0.70710678,   1.98      ,   0.70710678,   0.70710678,
         0.70710678,   0.70710678,   0.70710678,   0.70710678,
         1.98      ,   0.70710678,   0.70710678,   1.98      ,
         1.98      ,   0.70710678,   1.98      ,   1.98      ,
         0.70710678,   0.70710678,   1.98      ,   0.70710678,
         0.70710678,   0.70710678,   0.70710678,   0.70710678,
         0.70710678,   0.70710678,   1.98      ,   0.70710678,
         1.98      ,   1.98      ,   0.70710678,   0.70710678,
         0.70710678,   0.70710678,   0.70710678,   1.98      ,
         1.98      ,   0.70710678,   0.70710678,   0.70710678,
         0.70710678,   1.98      ,   0.70710678,   1.98      ,
         0.70710678,   0.70710678,   0.70710678,   1.98      ,
         0.70710678,   0.70710678,   1.98      ,   0.70710678,
         0.70710678,   1.98      ,   0.70710678,   0.70710678,
         1.98      ,   0.70710678,   0.70710678,   1.98      ,
         0.70710678,   0.70710678,   1.98      ,   1.98      ,
         0.70710678,   0.70710678,   0.70710678,   0.70710678,
```

```
# which record has the biggest distance from the closest centroid, is it an attack?
np.argmax(distances_from_centroids),np.max(distances_from_centroids)
```

```
(150, 196.01999999999995)
```

Figure 11 Outlier

Using the argmax on the distance data reveals the outlier. The row 150 has the maximum distance (196) from the center.

7.4.3 Deployment.

Deployment is the final step and the reason all of the above took place, to use the model in an application, or maybe an API. This is not covered in the Thesis.

8 Results/Performance

Obviously the functional requirement is the best possible accuracy of the model and the nonfunctional is the inference performance and scalability so that the solution is applicable.

8.1 Accuracy

The accuracy is improving with the inclusion of more parameters in the dataframe (columns) and more HTTP Requests (rows). Of course, not every added parameter will add value, this is a "try and fail" approach thus a better solution might be supervised learning with annotated data.

In this (unsupervised) scenario the percentage of false positives using K=2 was 0% and that's because it was an almost air-gapped environment that the malicious actor was attacking. (Principle of least privilege, n.d.) ensured that only the authorized developers could view/checkout code and only code-maintainers could modify code. An attacker who is using a stolen access token to access the source repo will probably use a different IP or choose off-work hours so there won't be any conflict with the actual owner of the access token, and these parameters can easily be modeled and trained.

In real world scenario the attacker may infiltrate the organization and using lateral movement access the code maintainer's PC and from there the source code. A row in the dataframe with a valid IP, a valid access token, from the same git client at working hours is almost impossible to be on the "malicious" cluster without improving the model with other parameters from the rest of the infrastructure, like VPN access logs, Suricata logs, Zeek, OSSEC agents from Developer's PC etc.

8.2 Performance/Scalability

To evaluate the performance of the given dataframe, a high-end GPU[1] was utilized, and 500.0000 requests were loaded (batches of 100.000) to the VRAM (24GBs) from a RAM Disk to minimize the I/O bottleneck. The process time was always below 1 second and the memory utilization was in VRAM limits even when all 500.000 requests loaded.

Of course, the dataset size is determinant factor. Each added parameter to the model increases the processing time and create bottleneck first to the CPU[2], then the PCI bus and last the GPU. The number of rows of the dataset, (requests) that can be loaded in the VRAM is not infinite. Solutions like the clustering algorithm (Aristides, Heikki, & Panayiotis, n.d.) must be evaluated for large datasets.

[1] Nvidia 4090

[2] AMD Ryzen 7800X3D

9 Conclusions

There is no doubt that ML is here to stay, especially in SSCA's case. Every tool that is able to corelate, ingest a large amount of "sensor" info and identify outliers is a huge asset to whom who knows how to use it to the maximum of it's potentials. The parameters, their sources (WebServer, Git logs, SSL termination, OSSEC, etc) the size of the datasets must all be evaluated and reevaluated.

Concerting the benefits of the unsupervised machine learning over supervised is the total absence of the need to annotate the datasets.

An interesting aspect of ML is the automation that can provide joining SecOps. Early evaluation means early notification and time to respond, maybe using automated actions and CACAO playbooks.

10 Bibliography

(χ.χ.). Ανάκτηση από NPM: https://www.npmjs.com

(χ.χ.). Ανάκτηση από Open Containers: https://opencontainers.org/

(χ.χ.). Ανάκτηση από Codecov: https://about.codecov.io

(χ.χ.). Ανάκτηση από OpenID: https://openid.net/developers/how-connect-works/

(2020, Mar 30). Ανάκτηση από SANS edu: https://isc.sans.edu/diaryimages/Kwampirs_PIN_20200330-001.pdf

Aristides, G., Heikki, M., & Panayiotis, T. (χ.χ.). *Clustering Aggregation*. Ανάκτηση από Boston University: https://cs-people.bu.edu/evimaria/cs565/aggregated-journal.pdf

Artificial intelligence for cybersecurity. (χ.χ.). Ανάκτηση από IBM: https://www.ibm.com/security/artificial-intelligence

CISA director says the LOG4J security flaw is the "most serious" she's seen in her career. (χ.χ.). Ανάκτηση από CNBC: https://www.cnbc.com/video/2021/12/16/cisa-director-says-the-log4j-security-flaw-is-the-most-serious-shes-seen-in-her-career.html

Cordey, S. (2023, January). *Software Supply Chain Attacks An Illustrated Typological Review*. Ανάκτηση από ethz.ch: https://www.enisa.europa.eu/news/enisa-news/understanding-the-increase-in-supply-chain-security-attacks

CVE-2021-44228. (2021, 10 12). Ανάκτηση από NIST: https://nvd.nist.gov/vuln/detail/CVE-2021-44228

Dataframe. (χ.χ.). Ανάκτηση από Pandas: https://pandas.pydata.org/docs/user_guide/dsintro.html#dataframe

Github Security. (χ.χ.). Ανάκτηση από Github: https://github.com/security/incident-response

IBM. (2022, 10 3). *New IBM Study Finds Cybersecurity Incident Responders Have Strong Sense of Service as Threats Cross Over to Physical World*. Ανάκτηση από ibm.com: https://newsroom.ibm.com/2022-10-03-New-IBM-Study-Finds-Cybersecurity-Incident-Responders-Have-Strong-Sense-of-Service-as-Threats-Cross-Over-to-Physical-World

Ilascu, I. (2021, Apr 16). *Popular Codecov code coverage tool hacked to steal dev credentials*. Ανάκτηση από Bleeping Computer: https://www.bleepingcomputer.com/news/security/popular-codecov-code-coverage-tool-hacked-to-steal-dev-credentials/

Linux Foundation. (χ.χ.). Ανάκτηση από https://www.linuxfoundation.org

Maven. (χ.χ.). Ανάκτηση από Apache: https://maven.apache.org/

McGill, T. (2021, Apr). *Hardware Trojans and Supply Lines*. Ανάκτηση από US. Naval Institute: https://www.usni.org/magazines/proceedings/2021/april/hardware-trojans-and-supply-lines

McGill, T. (2021, Apr). *Hardware Trojans and Supply Lines*. Ανάκτηση από US Naval Institute: https://www.usni.org/magazines/proceedings/2021/april/hardware-trojans-and-supply-lines

Milmo, D. (2022, Aug 11). *NHS ransomware attack*. Ανάκτηση από The Guardian: https://www.theguardian.com/technology/2022/aug/11/nhs-ransomware-attack-what-happened-and-how-bad-is-it

Munoz, A. (2020, May 28). *The Octopus Scanner Malware*. Ανάκτηση από Github: https://securitylab.github.com/research/octopus-scanner-malware-open-source-supply-chain/

NarrowAI. (χ.χ.). Ανάκτηση από DeepAI: https://deepai.org/machine-learning-glossary-and-terms/narrow-ai

OASIS Collaborative Automated Course of Action Operations (CACAO) for Cyber Security TC. (χ.χ.). Ανάκτηση από OASIS OPEN: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cacao

Orion Platform. (χ.χ.). Ανάκτηση από Solarwinds: https://www.solarwinds.com/orion-platform

Pandas. (χ.χ.). Ανάκτηση από Pydata: https://pandas.pydata.org/

Phishing Scams. (χ.χ.). Ανάκτηση από Federal Trade Commission: https://www.ftc.gov/news-events/topics/identity-theft/phishing-scams

Polkovnychenko, A. (2022, Dec 13). *PyPI malware creators are starting to employ Anti-Debug techniques*. Ανάκτηση από JFrog: https://jfrog.com/blog/pypi-malware-creators-are-starting-to-employ-anti-debug-techniques/

Principle of least privilege. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Principle_of_least_privilege

Principle of least privilege. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Principle_of_least_privilege

Rimol, M. (2022, Mar 7). *Top Security and Risk Management Trends for 2022*. Ανάκτηση από Gartner: https://www.gartner.com/en/newsroom/press-releases/2022-03-07-gartner-identifies-top-security-and-risk-management-trends-for-2022

Sasson, A. (2021, Mar 26). *20 Million Miners: Finding Malicious Cryptojacking Images in Docker Hub*. Ανάκτηση από Palo Alto Networks: https://unit42.paloaltonetworks.com/malicious-cryptojacking-images/

Seldon. (2021, Oct 29). *Machine Learning Regression Explained*. Ανάκτηση από seldon.io: https://www.seldon.io/machine-learning-regression-explained

Sigstore. (χ.χ.). Ανάκτηση από https://www.sigstore.dev

State of the Software Supply Chain. (2021). Ανάκτηση από Sonatype: https://www.sonatype.com/hubfs/Q3%202021-State%20of%20the%20Software%20Supply%20Chain-Report/SSSC-Report-2021_0913_PM_2.pdf

Supervised learning. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Supervised_learning

Support. (2020, Jul 16). Ανάκτηση από Twitter: https://twitter.com/TwitterSupport/status/1283591846464233474

The 2022 State of Open Source Security Report. (2022, May). Ανάκτηση από Snyk: https://go.snyk.io/state-of-open-source-security-report-2022.html

Unsupervised learning. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Unsupervised_learning

Vault. (χ.χ.). Ανάκτηση από Hashicorp: https://www.hashicorp.com/products/vault

Yoran, A. (2021, Dec 22). *One in 10 Assets Assessed Are Vulnerable to Log4Shell*. Ανάκτηση από Tenable: https://www.tenable.com/blog/one-in-10-assets-assessed-are-vulnerable-to-log4shell

Zero Trust Security. (χ.χ.). Ανάκτηση από Hashicorp: https://www.hashicorp.com/solutions/zero-trust-security