A Framework for E2E Audit Trails in System Architectures of Different Enterprise Classes

Luca Patzelt[1], Georg Neugebauer[1], Meik Döll[2], Sacha Hack[1], Tim Höner[1] and Marko Schuba[1]
[1]Department of Electrical Engineering and Computer Science, FH Aachen University of Applied Sciences, Eupener Str. 70, 52066 Aachen, Germany
[2]SOPTIM AG, Im Süsterfeld 5-7, 52072 Aachen, Germany

Keywords: Auditing, Auditing Framework, Audit Trail, End-to-End Audit Trail, E2E Audit Trail, Pseudonymisation.

Abstract: In today's world, there are more and more IT systems that are interconnected to provide services to a wide variety of business classes. Since their services are usually inevitably linked to financial and political interests, the number of attacks aimed at disrupting or profiting from these and the associated systems in various ways is constantly increasing. In this paper we design and implement a framework for the comprehensive auditing of IT systems in system architectures of different enterprise classes. For our solution, we evaluate formal requirements regarding audit trails, provide concepts for the pseudonymisation of audit data, develop software components for E2E audit trails and finally present a secure system architecture based on Kubernetes and Istio in conjunction with the storage components ArangoDB and HashiCorp Vault to achieve an efficient framework for creating E2E audit trails.

1 INTRODUCTION

The constant desire of companies to automate and optimise work processes through the introduction of IT systems and software processes is creating an increasingly digital world. As a result of these digital and partly complex processes in companies, the risk of cyber-attacks on the IT infrastructure and systems continues to rise.

The complete logging and tracking of events in IT systems, also called auditing, is a related and recurring demand of companies. The aim of these audits is to make both malicious activities by hostile actors such as unintentional changes by legitimate and authorised actors transparent and to support subsequent actions.

In doing so, events should not only be considered on one system, but linked across many different systems. On the one hand, this would ease the tracing of events and their origins, and on the other hand, it would enable the possibility to detect attacks that might remain undetected by simple audit trails. The audit trail required for this is therefore defined as an "E2E audit trail".

Due to the many ways in which IT systems can be linked together, as well as the different requirements of companies, it makes sense to look at them in terms of auditing by company class. Here, the focus is primarily on the three company classes "universities", "KRITIS service providers" and "SaaS service providers".

The accumulation and processing of audit data across various systems as an E2E audit trail therefore has great potential to improve the security of an IT infrastructure and the systems it contains. In this paper we present an implementation of a framework for E2E audit trails that enables a company to establish such a process in its own system architecture using the developed framework.

For our solution, we evaluate formal requirements regarding audit trails (Section 2), provide concepts for the pseudonymisation of audit data (Section 3), discuss attack scenarios and suitable rule sets (Section 4), develop software components for E2E audit trails within a secure system architecture (Section 5), present an evaluation with an example of E2E violation graphs (Section 6) and finally conclude our paper in Section 7.

2 AUDIT TRAILS

The term "audit trail" is a combination of the two

Patzelt, L., Neugebauer, G., Döll, M., Hack, S., Höner, T. and Schuba, M.
A Framework for E2E Audit Trails in System Architectures of Different Enterprise Classes.
DOI: 10.5220/0012367000003648
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP 2024)*, pages 750-757
ISBN: 978-989-758-683-5; ISSN: 2184-4356
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

words audit (from the Latin audire: to hear - in this case an examination of processes, activities or results) and the English word trail, and thus describes a process which enables the tracking and named examination of "actual or attempted actions monitored and documented or recorded electronically or manually or on paper during a certain period of time" (Dr. Siller et. al, 2018).

Primarily, events and the associated meta-information are stored in the form of audit entries as a continuous record, often called a "log". These logs are then usually stored centrally in a system to allow easy tracking of results in this system. The focus is generally on documenting changes and deletions, as these are particularly critical events regarding IT security attacks.

2.1 E2E Audit Trails

An "E2E audit trail" can document or track an event across several systems and thus allows the detection of possible anomalies between individual audit trails. For this purpose, regular audit trails as well as other (independent) audit messages and master data from different systems are collected to create logical links between them and thus correlate previously independent results with each other.

The primary goal is to log an event from the first instance in which a suspected malicious actor triggers an action, e.g., an interaction with a client application on a local end device, to the last instance, e.g., the modification or deletion of a value or entry in a data record of a database. At best, the complete chain of events or all interactions between all IT systems involved (from the user device to the database) is recorded so that it can be analysed without gaps. This makes it possible to detect a malicious act, e.g., by a hostile actor, even if each individual event appears inconspicuous by itself. Figure 1 below shows an application example.

3 PSEUDONYMISATION

The core task of an E2E audit trail, or audit trails in general, deals centrally with the logging or, more generally, the processing of data. Since this often involves sensitive or personal data, this project is often subject to local and overarching regulations.

An essential factor in the processing of sensitive or personal data, specifically here in Europe, is compliance with the European Data Protection Regulation, also known as the "GDPR". In relation to audit trails, the anonymisation and pseudonymisation of the data to be processed, with the aim of protecting the personal rights of the persons concerned, is an important part of the initial data processing. The audit information is collected, processed and stored.

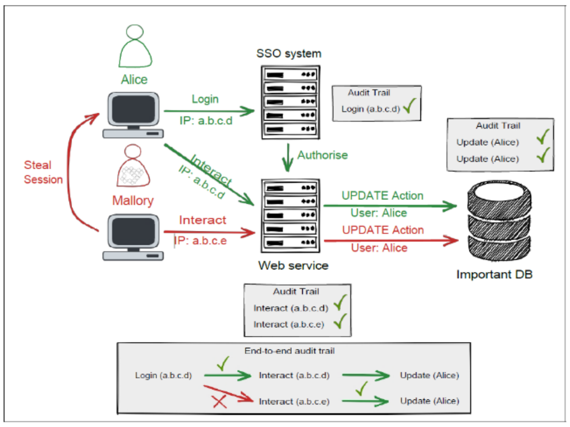

Figure 1: Difference E2E-/Audit Trail.

The fundamental difference between "anonymisation" and "pseudonymisation" is that personal data is irreversibly changed or removed in anonymisation so that a person can no longer be directly or indirectly identified. In contrast, in the case of pseudonymisation, data are changed in such a way that they can no longer be directly attributed to a person and additional information is required to assign them to a person again.

Since the possibility of linking independent events via common identifiers is indispensable for the evaluation of E2E audit trails, the following focuses primarily on the pseudonymisation of audit information, since anonymisation would make subsequent reidentification (e.g., in the case of a concrete security incident) impossible and thus defeat the purpose of an audit trail.

To support the implementation of GDPR-compliant pseudonymisation, the European Union Agency for Cyber Security (ENISA) publishes reports on, among others, "Techniques, Use Cases and Best Practices" (ENISA Best Practices, 2019 ENISA Advanced Techniques, 2022) in data pseudonymisation which we use as a guideline for data pseudonymisation within our framework.

3.1 Hierarchical Pseudonymisation Procedure

Regarding E2E audit trails, we have a challenging situation with respect to pseudonymisation, as on the one hand there is a great risk of re-identification due to speculation, but on the other hand the ability to link audit entries and identify real persons or entities in case of concrete suspicion must be maintained.

Since these requirements cannot be met by any of the three mutually exclusive pseudonymisation guidelines (see ENISA references) alone, a hierarchical or multi-level pseudonymisation procedure is recommended for the implementation of E2E audit trails, in which the identifiers are not purely "deterministic", "document-randomised" or "fully randomised", but with a combination of a "deterministic" and a "fully randomised" pseudonymisation. The concept of a multi-stage pseudonymisation procedure is not new and was described, among other things, in a white paper revised in 2019 by Rolf Schwartmann and Steffen Weiß, together with the Federal Ministry of the Interior, for Construction and Home Affairs, in 2018 (Schwartmann et al., 2019).

The aim of the implementation of such a multi-level pseudonymisation procedure developed here, is that the data records are both persisted logically separated from each other by many different pseudonyms per identifier and can be linked to each other via additional protected knowledge. When designing such a procedure, it should also be noted that none of the entities need to be completely depseudonymised during processing. This should only be necessary in cases of concrete suspicion, in which, for example, the interest of uncovering a concrete crime outweighs the personal rights of the person concerned.

Finally, it is advisable to implement the pseudonymisation procedure in such a way that it allows distributed calculation or pseudonymisation across several instances. This ensures that the required computational resources can be scaled across cluster-based environments used within our framework.

Regarding the tooling to be used, a multi-stage pseudonymisation procedure will be implemented using our storage concept defined in Section 5.2. Through its use, a complete logical separation between an identifier and a subpseudonym is carried out primarily with the help of the master pseudonym. In short, the subpseudonym is chosen with the help of a cryptographically secure pseudo-random generator, so that any kind of cryptanalysis with the aim of obtaining information about the identifier from the subpseudonym is useless. By choosing the smallest possible change interval of the subpseudonym and its resulting constant rotation, the amount of information per subpseudonym can be reduced to a minimum (Schwartmann et al.). This simultaneously reduces the ability of an attacker to carry out a speculative attack or the use of insider and background information to depseudonymise pseudonyms within our system.

As protection against brute force and dictionary attacks or the dictionary search, the use of secure key-dependent cryptographic hash functions or key derivation functions is recommended.

Finally, it is very important that secure data stores are used for the persistence of the ID master mapping, the master subsets and the audit record entries.

4 RULE SETS FOR ATTACK SCENARIOS

With the aim of an E2E audit trail to detect attacks as quickly as possible, or to reconstruct the sequence of events, the first step is to analyse which attack scenarios are common and where they are likely to leave traces. Rule modules can be created for the creation of rule sets, which can later be combined by an E2E audit trail.

```
def arbitrary Rule Module (** kwargs ) ->
    None : userhash = kwargs [' username ']
    timestamp = kwargs [' timestamp ']
    ...

    proof Obj = arbitrary_query ( userhash , timestamp )
    if proof Obj is not None and proof Obj [' status '] != " expected
        ": # Report a violation or odd behaviour
        report Violation ( proofObj , ' Something odd happened !', weigth =
        high )
    ...
# end arbitrary Rule Module
```

Figure 2: Arbitrary rule module.

A rule module, as shown in Figure 2, primarily consists of an entry method (here "*arbitraryRuleModule*") that takes as parameters any data needed for the decisions to be made in the module. Within this method, further required data may be retrieved with the help of data bank queries (here "*arbitrary_query*") to incorporate them into the decision-making process. If a rule violation is detected, this can be "reported" (here with "*reportViolation*"), whereby a relationship between the current entry and an evidence object, in combination with a remark and a risk assessment, is marked for persistence in the data storage. After all entries have been processed, all flagged relationships are saved as part of a transaction.

4.1 Attack Scenarios

In the following, a series of attack scenarios are listed, as they could be encountered in all three company classes. In addition, possible rules or pseudo code for their use are named, with which this attack can be recognised and, if necessary, further steps of an attacker can be identified and thus probably prevented.

Misuse of Authorisations by Internal Employees
According to KPMG, the role of employees in relation to IT security in a company should not be neglected: "The perception of misuse of a privileged account by an internal employee ranks fifth among cybersecurity vectors at 23% (*Mihaela, L. C., 2020*). With the help of suitable auditing procedures, it is possible to identify future security risks at an early stage and prevent further misuse. Although the use of an E2E audit trail does not bring any great added value compared to a regular audit trail, it does make it possible to cover the not directly recognisable case of several employees joining forces intentionally or unintentionally and carrying out undesirable actions in a system together with their authorisations.

Development Chain Attack
A development chain attack, or DCA, is like a supply chain attack which is characterised by the fact that an attacker does not attempt to directly attack or manipulate a resource, but instead directs his attack at the "supply chain" of the resource. Here, a sub-resource, such as an embedded dependency or a useful resource used in the development of the resource, such as a server, images, programmes, etc., are changed. This leads to the resource now using or interacting with the manipulated sub-resource and the attacker can thus change the behaviour of the actual resource without having to change it directly. An example of an attack on the development chain is shown in Figure 3.

Here Alice is developing a very important application that uses the hash function *h(x)*. To ensure that the functionality is preserved, she compares the source code of her development environment with that of the version management and checks the artefact against several known inputs and outputs after the build process.

Mallory then injects a malicious build image into the build process, altering the internal implementation of the hash function and allowing, for example, the easy creation of collisions. Mallory was, of course, clever enough to make her changes in such a way that they could not be detected by the tests Alice performed.

The delivery of the application to a customer, e.g., an operator of critical infrastructure, without a deep check at bytecode level, now leads to the fact that the process in which the application is used has been compromised without this being apparent.

To detect such an attack, an E2E audit trail can now audit every component involved in the development and build process and show whether and when it was changed by which person. If it is now determined that changes have been made that were not foreseen, a possible compromise can be uncovered and, if necessary, a new, this time clean, build process can be started. A set of rules for this would have to be adapted to the components used so that it includes and verifies all components.

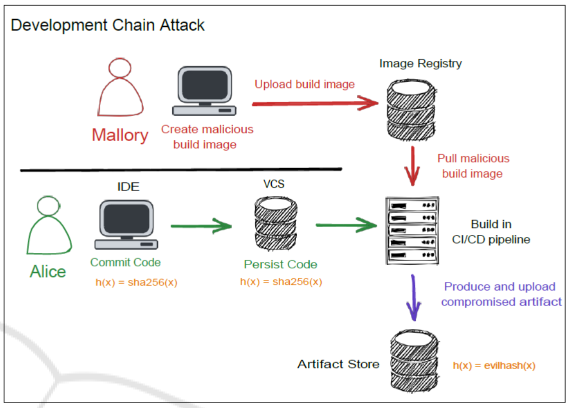

Figure 3: Development chain attack, "Trusting Trust".

5 SECURE SYSTEM ARCHITECTURE FOR E2E AUDIT TRAILS, DESIGN AND IMPLEMENTATION

For a secure implementation of a framework for E2E audit trails, the first step is the secure design of a system architecture for such a framework. Therefore, we highlight the required sub-aspects to design such a secure system architecture.

5.1 Requirements for a Secure System Architecture

We start with three major requirements based on the three enterprise classes described in the introduction.

Multi-client capability is one of the resulting requirements, which is particularly obligatory for KRITIS and SaaS service providers and is especially recommended for shared infrastructures, such as at universities, and any resulting shared auditing systems.

On-Premises is a requirement that arises from the need to run the software on one's own servers or the hardware used for it in the local infrastructure. One of the main reasons for this requirement is the processing and pseudonymised storage of sensitive and personal data that is required for later evaluation. Transparency does not stem directly from a requirement of a specific business clas, but from the general need to process sensitive and personal data within the European Union "lawfully, fairly and in a manner comprehensible to the data subject" (DSGVO, 2016).

5.2 Data Storage Concept

One of the main problems in implementing a framework for E2E audit trails is the choice of the correct or suitable storage infrastructure regarding the storage of the collected and pseudonymised audit entries as well as the associated pseudonym mappings.

When looking at the expected results of an E2E audit trail, it is noticeable that they can be represented as a graph. Each audit entry represents a node of a graph and the relationship between two audit entries can therefore be represented as an edge. With this knowledge it is obvious that a graph-oriented database, which is based on the functionality of a document-oriented database, has a very high potential for storing this kind of data. For this reason, we use the "ArangoDB" (ArangoDB, 2023) since it is well-suited for our data storage concept as a graph database, relatively widespread, and ships with a fully functional basic version which is available free of charge and enables cluster operation, which makes it promising for a scalable implementation.

Secure Key Value Store

In addition to the storage of pure audit record entries, the persistence of pseudonym mappings plays an extremely important role, as sensitive and personal data is processed or stored here in the case of the mapping between the original value and the master pseudonym. Therefore, secure storage and protection is of great importance here in particular.

One of the many software solutions for storing key-value pairs, which is a REST-based secret data store, is "Vault" by the company HashiCorp (HashiCorp). The decision in favour of a secret data store can be justified from the point of view of security. Due to its classification as a secret data store, Vault is designed for a very high security standard and, in addition to a range of different authentication methods, also offers the possibility of assigning clients their own key-value stores and creating dedicated access policies for them. In addition, Vault has so-called "audit devices", which can optionally save each request to the REST interface in a file or send it to a syslog server or a TCP, UDP or UNIX socket. This makes it possible to seamlessly trace every access to the data stored there and ensure that unauthorised access to sensitive and personal data cannot be completely ruled out but can at least be detected with a very high degree of certainty.

As a summary, with the decision to use ArangoDB as the storage solution for the large number of audit record entries, as well as the mappings between the master and subpseudonyms, and the choice of Vault as the key value storage for the sensitive mappings between the original values and the respective associated master pseudonym, a secure data storage concept is defined.

5.3 Cluster-Based System Architecture

Due to the great heterogeneity of the source systems and the data they contain, dynamically expandable and scalable infrastructures and processing units are needed that can be modularly adapted to the required systems. As already mentioned in the previous sections, a virtualised cluster environment is recommended for the implementation of a secure system architecture, which allows the most diverse components to be logically separated from each other with the help of ACL-like constructs.

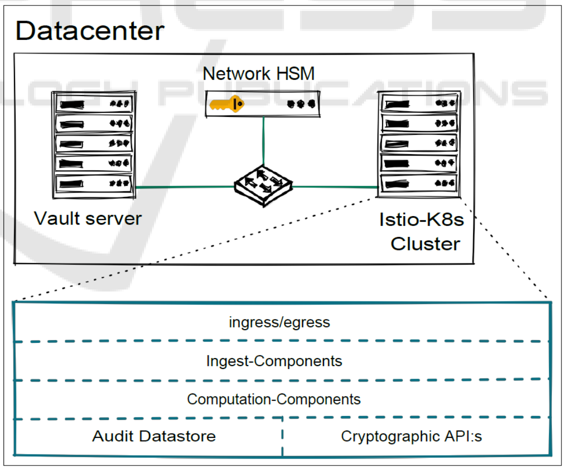

Figure 4: Structure of our E2E framework.

One software component that can meet these requirements is the free and world-leading container orchestration software "Kubernetes" ("K8s") (Kubernetes, 2023), which in combination with a so-called "service mesh", in this case "Istio" (Istio, 2023), enables the construction of a secure cluster infrastructure. This is specially designed for the provisioning and administration of containers, i.e., applications whose executable files, resources and dependencies are accumulated in a memory image and can be executed very efficiently on a compatible operating system.

The "service mesh", in this case various Istio components, establishes a policy-based management for all managed applications in the cluster and thus enables the monitoring of all network processes as well as the management of data traffic between the applications and their protection. Figure 4 shows a possible structure of such a data centre and cluster architecture.

Figure 5 shows how different components that are required for the processes of an E2E audit trail can be combined. With the help of Istio, full encryption via mTLS is possible within the cluster.

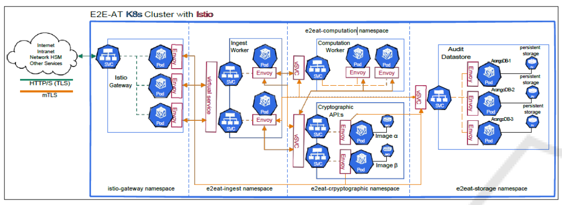

Figure 5: Kubernetes cluster with Istio.

Istio also provides various types of guidelines that make it possible to isolate the components in the illustration, visible by the arrows, from each other and to control the data flow between them in a fine-grained way. The most important types of guidelines and configurations are:

Gateway Configuration
A gateway is a virtual definition of a load balancer, which acts at the network edge of the service mesh. The gateways describe how incoming and outgoing HTTP/TCP connections are to be handled.

Virtual Service Configuration
The configuration serves as a virtual switch component between gateways and services (SVC) and between services themselves and is applied to the proxies of the individual pods. It primarily specifies how to route between two components.

Destination Rules
Following the routing decision by the upstream "Virtual Service Configuration", this rule now determines how the traffic to the destination is to be handled in more detail. The topic of load distribution and its procedures, e.g., round robin, plays an important role here.

Authorisation Policies
The core competence of this guideline lies in making authorisation decisions. Primarily, the following W-questions are asked: "Who" may interact "when" with "whom" under "which" conditions "how"?

Peer Authentication Policies
Finally, it is essential to define this policy at least per namespace, a logical grouping of resources, if not for each group of pods to be defined, as it specifies how or whether which traffic from which port is to be secured.

Summary
Figure 5 therefore shows a secure system architecture that can be used in all three corporate classes presented in Section 2. The guidelines can be adapted exactly to the respective requirements.

Regarding the major requirements defined in this section, all three can be fulfilled. The requirement of multi-client capability results from the data storage and the stateless functioning of the other components, apart from the cache of the mappings between the original values and the respective associated master pseudonym in the cryptographic APIs. Due to the cluster-based structure of the system architecture, operation in the cloud as well as on-premises is possible, which also fulfils this requirement. Finally, there is the demand for transparency, which shows through clear guidelines based on Istio at which points data may flow from one point to another.

6 EVALUATIONS

In the following, we show the evaluation based on E2E result graphs computed by our framework using the computation workers described before. By combining several rule modules and their database queries a computation worker can relate audit data to each other (e.g., "A violates B" or "C uses D") via a processor. The set of these relationships (edges) together with the entries of the audit data (vertices)

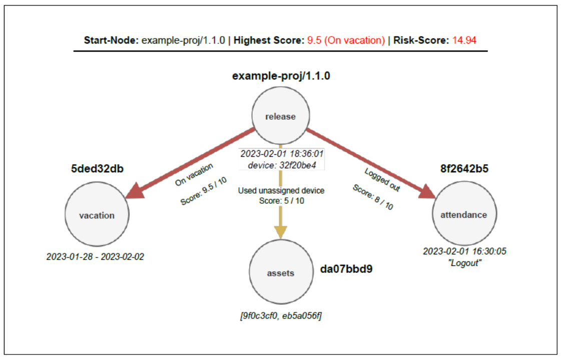

Figure 6: Violation graph for software delivery.

form a graph. Using attributes of the edges (e.g., identifier, colour, weighting, etc.), statements can now be made about a complex interrelated process.

The *ViolationProcessor* of our framework provides so-called "violation graphs", as shown in Figure 6, which have directed edges from a source entity to an evidence entity that proves the rule violation. An example of an evidence entity could be an entry in an attendance log showing that an employee was not logged in at the time of an event (source entity). However, it should always be noted that a rule violation does not directly equate to a danger, an attack or an intentional rule violation, since, for example, forgetting to log in can lead to false positives regarding an actual risk. Using key performance indicators, an aggregated risk assessment can additionally be given via this graph, which could be an indicator for the urgency of an investigation.

In the case of the graph in Figure 6, exemplary violations are shown using a release event in a source code administration. The first thing to notice is that the last event in the attendance record (CRC hash: 8f2642b5) is a logoff. This means that the person was not logged in at that time (01.02.2023 18:36:01). Secondly, a holiday entry is found that covers exactly this time period (28.01-02.02.2023). Finally, the device used (CRC hash: 32f20be4) is compared with the data of the inventory management (CRC hash: da07bbd9), whereby it can be determined that the person used a device that is known but not assigned to him (CRC hashes of his devices: (9f0c3cf0, eb5a056f)). The colours of the arrows, in combination with the "scores", indicate the risk assessment regarding this relationship.

Risk Score: For our framework, we used a weighted sum function because it is cubed for large values and increases faster than for small ones, but the sum remains comparable for the set of values in the sample data. Based on the set of relationships and corresponding values in the resulting graphs, it may make sense to evaluate the use of another function.

As a second type of result graph, the E2E processor of our framework provides "E2EGraphs", as can be seen in Figure 7, which show all relationships of all entities, not only those that violate rules. This type of graph is therefore particularly suitable for the end-to-end tracking of events and makes it possible to quickly and intuitively get an overview of all the components involved in an event and their relationships to each other. Through different colours and weights of the edges of the graphs, the observer can now directly recognise which relationships in this graph are positive, neutral or negative. At the same time, the thickness of an edge, i.e., it's the weighting or the risk value, the user can get a feeling for how critical a certain relationship is. This can be seen particularly well in Figure 7, which shows three large red arrows with a high risk value, two medium-thick yellow arrows with a medium risk value, and three thin green arrows with a non-existent risk value.

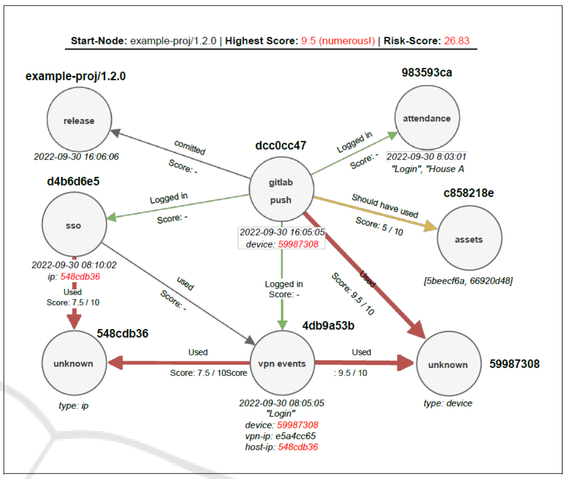

Figure 7: E2E graph for software delivery.

Based on key indicators, or threshold values of the key indicators, for the graph types, an evaluation process can now be carried out, which can constantly trigger notifications and external actions. The key figures can also be generated directly from the data storage by database queries, so that this should also be possible for many graphs in an acceptable time.

Graphs without weights, i.e., events without relationships, are filtered out, as no key performance indicators need to be calculated for them. Finally, the risk score is calculated by first cubing and dividing all weights. This query then returns the event identifier, the highest risk value and the risk score by summing up the pre-calculated individual values of the corresponding graph. Downstream actions could include the creation of a support ticket, the temporary blocking of a user or their traffic, and the complete isolation of a system.

7 CONCLUSIONS

In this paper, we presented the design and implementation of our framework for E2E audit trails in system architectures of different enterprise classes. Using our framework as an example, it was possible to show how audit log data from different systems can be imported, processed and prepared in the form of directed and weighted graphs by combining different components. These graphs now form the basis for detecting attacks on IT systems with the help of end-to-end tracking. Due to the very modular structure of the framework and the associated plug-ins and rule modules, it can be adapted to the different requirements of the enterprise classes and associated system architectures (see Section 2). By classifying all source systems to be connected, requirement profiles can be defined for a group of systems, which then enable the systems to be connected via a common or unified ingest process. User-defined rule modules for different types and patterns of attacks, as described in Section 3, then enable a meaningful linking of the data and a fine-grained tracking of rule violations. The use of intelligent systems or knowledge databases can have a supporting effect here.

The entire process uses a multi-level or hierarchical pseudonymisation procedure (see Section 3) that protects the personal data of clients, employees, students or other persons from whom data is collected. Based on the policies of the enterprise classes, the data store allows, for example, the simple deletion of data of a client that is older than x days. As the graphs are based directly on this data, they can be deleted together with the data without the risk of a hanging reference.

By using realistic data formats in the design of the sample data used, related to the expected formats of real data and log data of industry-standard Software based on information and requirements from industry cooperation’s, it has already been shown that the result graphs have a high potential to detect real and sophisticated attacks within industry.

REFERENCES

ArangoDB (2023), ArangoDB Inc. Graphenda tenbanksoftware. Version 3.9.7. URL: https:// github.com/arangodb/arangodb/tree/3.9.7, visited on 23/10/2023

DSGVO (2016) - VERORDNUNG (EU) 2016/679 DES EUROPÄISCHEN PARLAMENTS UND DES RATES (2016) - vom 27.4.2016 (Amtsblatt L 119 vom 4.5.2016, S. 1, ber. Amtsblatt L 314 vom 22.11.2016, S. 72, Amtsblatt L 127 vom 23.5.2018, S. 2). URL: https://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32016R0679

ENISA Best Practices (2019), European Union Agency for Cybersecurity u. a. Pseudonymisation techniques and best practices: recommendations on shaping technology according to data protection and privacy provisions. Hrsg. von I Agrafiotis, A Bourka und P Drogkaris. Publications Office, 2019. DOI: 10.2824/247711.

ENISA Advanced Techniques (2022), European Union Agency for Cybersecurity u. a. Data pseudonymisation : advanced techniques and use cases : technical analysis of cybersecurity measures in data protection and privacy. Hrsg. von P Drogkaris und A Bourka. European Union Agency for Cybersecurity, 2022. DOI: 10.2824/860099.

HashiCorp (2023), HashiCorp. Vault. Geheimnisda tenspeicher. Version 1.12.2. URL: https://github.com/ hashicorp/vault/tree/v1.12.2, visited on 23/10/2023

Istio (2023), Cloud Native Computing Foundation. Istio. Kubernetes Service-Mesh. Version 1.16.1. URL: https://istio.io/latest/news/releases/1.16.x/announcing-1.16.1/, visited on 23/10/2023

Kubernetes (2023), Cloud Native Computing Foundation. Container-Orchestrierungssoftware. Version 1.24.6. URL: https://github.com/kubernetes/kubernetes/tree/ v1.24.6 visited on 23/10/2023

Mihaela, L. C. (2020) „Current security threats in the national and international context“. In: Journal of Accounting and Management Information Systems 19.4 (2 2020), S. 351–378. ISSN: 1583-4387. DOI: 10.24818/jamis.2020.02007.

Ping C., Lieven D. and Christophe H. (2014). „A Study on Advanced Persistent Threats“. In: Communications and Multimedia Security. Hrsg. von Bart De Decker und André Zúquete. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, S. 63–72. ISBN: 978-3-662-44885-4. DOI: 10.1007/978-3-662-44885-4_5.

Schwartmann R. and Weiß S. (2019), Anforderungen an den datenschutzkonformen Einsatz von Pseudonymisierungslösungen. Version 1.01. 2019. URL: https://www.gdd.de/downloads/anforderungen-an-datenschutzkonforme-pseudonymisierung visited on 23/10/2023

Dr. Siller Betriebsberatung and Training Prof. (FH) Mag. Dr. Helmut Siller MSc. (2018), „Audit Trail“. In: Gabler Wirtschaftslexikon 53401.276494 (2018). URL: https://wirtschaftslexikon.gabler.de/definition/audit-trail-53401/version-276494, visited on 23/10/2023

Weir G. et. al (2017) „Cloud accounting systems, the audit trail, forensics and the EU GDPR: how hard can it be?“ In: British Accounting & Finance Association (BAFA) Annual Conference 2017.63134/

Wheeler D. A. (2005), „Countering trusting trust through diverse double-compiling“. In: 21st Annual Computer Security Applications Conference (ACSAC’05). 2005, 13 pp.–48. DOI: 10.1109/CSAC.2005.17.