Cyber Security PPP: Addressing Advanced Cyber Security Threats and Threat Actors

Cyber Security Threats and Threat Actors Training - Assurance Driven Multi- Layer, end-to-end Simulation and Training

D5.1: Real event logs statistical profiling module and synthetic event log generator v1[†]

Abstract: This deliverable provides a first report on designing and developing of two major components of the THREAT-ARREST training platform: (1) the generator of synthetic event logs, based on real event logs profiling and customisation of IBM's Data Fabrication Platform and (2) the module that will carry out the statistical profiling of real event logs, as they are received from the Assurance Platform in work package (WP) 3.

Contractual Date of Delivery	31/08/2019
Actual Date of Delivery	31/08/2019
Deliverable Security Class	Public
Editor	*Michael Vinov (IBM)*
Contributors	IBM, STS, TUBS
Quality Assurance	*George Hatzivasilis (FORTH),* *Martin Kunc (CZNIC)*

[†] *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 786890.*

The *THREAT-ARREST* Consortium

Foundation for Research and Technology – Hellas (FORTH)	Greece
SIMPLAN AG (SIMPLAN)	Germany
Sphynx Technology Solutions (STS)	Switzerland
Universita Degli Studi di Milano (UMIL)	Italy
ATOS Spain S.A. (ATOS)	Spain
IBM Israel – Science and Technology LTD (IBM)	Israel
Social Engineering Academy GMBH (SEA)	Germany
Information Technology for Market Leadership (ITML)	Greece
Bird & Bird LLP (B&B)	United Kingdom
Technische Universitaet Braunschweig (TUBS)	Germany
CZ.NIC, ZSPO (CZNIC)	Czech Republic
DANAOS Shipping Company LTD (DANAOS)	Cyprus
TUV HELLAS TUV NORD (TUV)	Greece
LIGHTSOURCE LAB LTD (LSE)	Ireland
Agenzia Regionale Strategica per la Salute ed il Sociale (ARESS)	Italy

Document Revisions & Quality Assurance

Internal Reviewers

1. *George Hatzivasilis (FORTH)*
2. *Martin Kunc (CZNIC)*

Revisions

Version	Date	By	Overview
1.0	27/08/2019	Editor	Final Version
0.6	5/08/2019	Marinos Tsantekidis	Addressed FORTH's review comments
0.5	19/07/2019	Marinos Tsantekidis	Completed Statistical Profiling module
0.4	23/07/2019	Editor	Added Introduction and Conclusions sections
0.3	21/06/2019	Marinos Tsantekidis	Added introduction of the Statistical Profiling module
0.2	19/06/2019	Editor	Initial description of the Data and Security logs fabrication technology
0.1	12/05/2019	Editor	First Draft

Executive Summary

This document provides a first report on designing and developing of two major components of the THREAT-ARREST training platform: (1) the generator of synthetic event logs, based on real event logs profiling and customisation of IBM's Data Fabrication Platform (DFP) and (2) the module that will carry out the statistical profiling of real event logs, as they are received from the Assurance Platform (WP3). D5.1 is the first deliverable of the task "T5.2 – Statistical profiling of real event logs and generation of synthetic events logs".

The deliverable includes an introduction to the DFP technology and describes its extension to support fabrication of cyber security log files.

To improve the accuracy of synthetic security event generation process, the project security assurance and monitoring platform was extended to enable statistical profiling of real event logs and identify the statistical profiles of different types of individual system events, as well as different combinations of such events.

Table of Contents

List of Abbreviations

DFP	Data Fabrication Platform
TDF	Test Data Fabrication
CSP	Constraints Satisfaction Problem
CTTP	Cyber Threat Training and Preparation
UML	Unified Modeling Language
WP	Work Package

List of Figures

1 Introduction

The objective of this document is to present and provide a first report on designing and developing of two major components of the THREAT-ARREST training platform: (1) the generator of synthetic event logs and (2) the module that will carry out the statistical profiling of real event logs, as they are received from the Assurance Platform (WP3). D5.1 is the first deliverable of the task "T5.2 – Statistical profiling of real event logs and generation of synthetic events logs".

The deliverable includes an introduction to IBM's Data Fabrication Platform (DFP) technology and a description of its Constraint Satisfaction Programming (CSP) solver to generate synthetic realistic data, followed by a description of the initial work done to enhance the tool with the ability to generate sequences of simulated cyber-events in general, and synthetic security events log files in particular to support the THREAT-ARREST project requirements.

A methodology and technical approach for statistical profiling of real event logs is also provided. Real event logs are captured from the operation of the cyber system through the assurance platform connected to the THREAT-ARREST framework. The profiling and analysis of these events also enables detection of attacks with different temporal characteristics, including instantaneous, long-lived and repeating attacks. The operation of this mechanism is driven by simulation-related specifications of the Cyber Threat and Training Preparation (CTTP) model.

The document is structured as: Section 2 outlines the operation of DFP and its extension for THREAT-ARREST, Section 3 presents the component for statistical profiling of real event logs, and Section 4 concludes and links overall contribution with other related tasks and deliverables.

2 Generator of Synthetic Security Event Logs

2.1 IBM’s Data Fabrication Platform (DFP)

2.1.1 Introduction to DFP

IBM’s Data Fabrication Platform (DFP) (IBM, 2017; IBM Research, 2011) is a web-based central platform for generating high-quality data for testing, development, and training. The platform provides a consistent and organizational wide methodology for creating test data. The methodology used is termed “rule-guided fabrication”.

The primary DFP use case for fabricating synthetic data contains two actors: a user (initiator) and Database/File (participators). This use case includes two sub-scenarios: data requirements modelling and data generation. The data requirements use case includes three sub-scenarios: resources and structure definitions, constraint rules definitions and fabrication configuration definitions. The data structure for databases (schema, tables, columns, etc.) is automatically imported, however structural hierarchy of data elements (structs, arrays, tables, fields, types) need to be manually defined by the user. The constraint rules are required to construct a model of the data and thus enable creation of meaningful realistic data values. Input and output resources include standard relational databases (e.g., DB2, Oracle, PostgreSQL, SQLite), standard file formats (e.g., Flat file, XLS, CSV, XML, JSON) and streaming via the MQTT protocol. The next figure depicts the related Unified Modeling Language (UML) use case diagram.

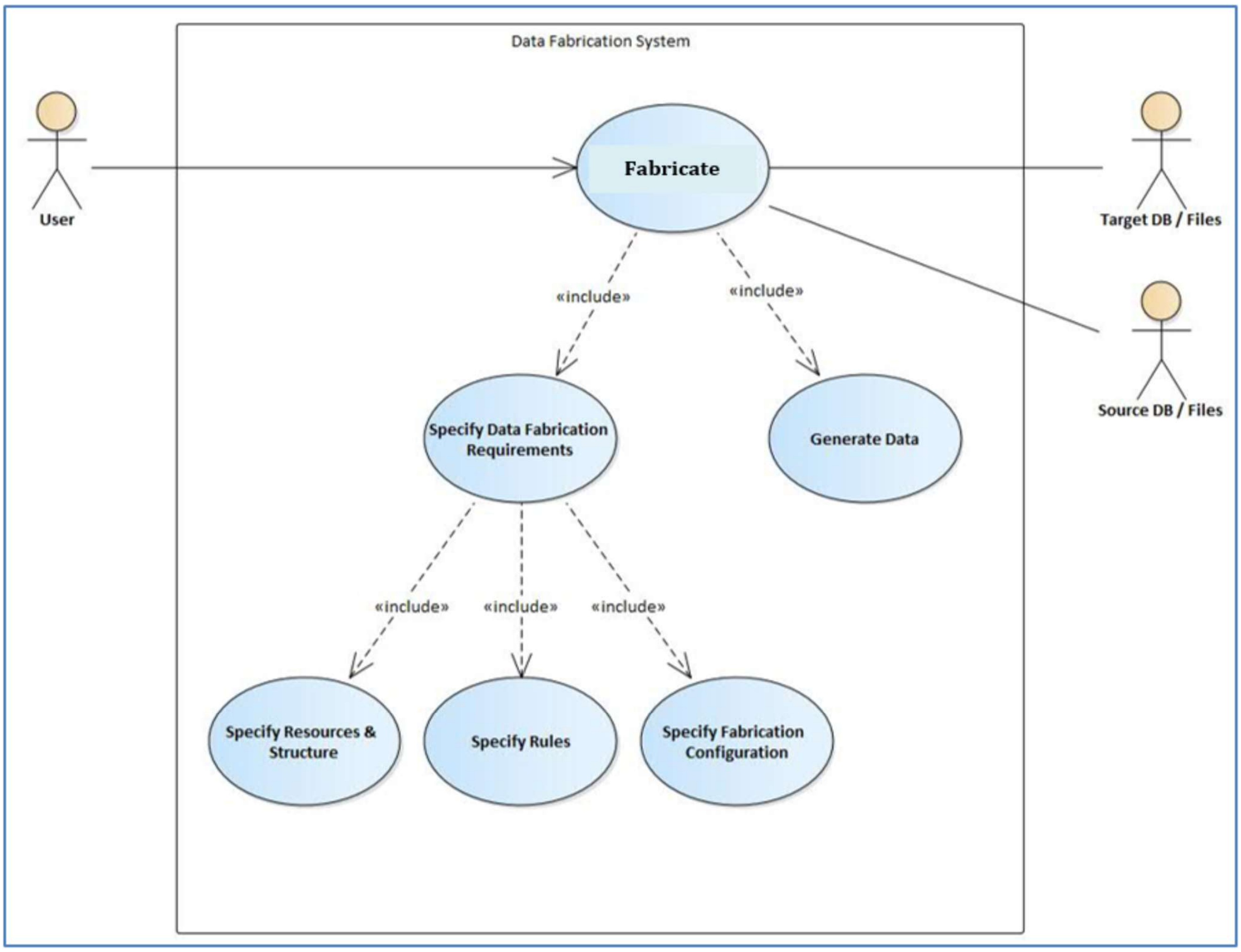

Figure 1 UML use case diagram for the DFP

In rule-guided fabrication, the database logic is extracted automatically and is augmented by application logic and testing logic modelled by the user.

The application logic and the testing logic can be modelled using rules that the platform provides. The platform is also extendible and new rule types can be added by users and automatically integrated into the platform in an organization-wide manner.

Once the user requests the generation of a certain amount of data into a set of test databases, the platform internally ensures that the generated data satisfies the modelled rules as well as the internal databases consistency requirements.

The platform can generate data from scratch, inflate existing databases, move existing data, and transform data from previously existing resources, such as old test databases or even production data. The platform provides a comprehensive and hybrid solution that can create a mixture of synthetic and real data according to the user requirements.

IBM Data Fabrication Platform uses a proprietary CSP solver (IBM, 2002; Richter et al., 2007; Naveh et al., 2006; Bin et al., 2002; Abaroni et al., 2015; Boni et al., 2012; Ben-Haim et al., 2012; Bin et al., 2011; Shin et al., 2011; Naveh, 2010; Asaf et al., 2010; Dubrov et al., 2009; Richter et al., 2007; Sabato and Naveb, 2007; Naveb et al., 2006; Richter et al. 2006; Naveb and Emek, 2006; Naveb 2005; Geller and Veksler, 2005; Dechter et al., 2002; Lewin et al., 1995) that has been used for verifying IBM hardware systems for over a decade. The solver finds a solution for all the requirements in a data fabrication task. The solver is capable of handling very complex problems in a timely manner. The following figure depicts the overall Test Data Fabrication (TDF) flow.

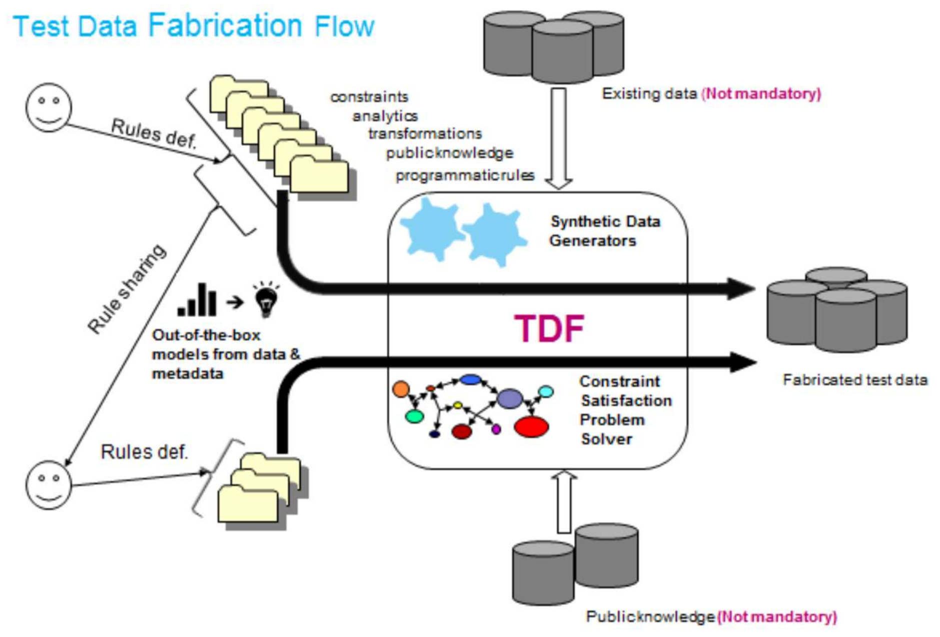
Test Data Fabrication Flow
Rules def.
constraints
analytics
transformations
publicknowledge
programmaticrules
Existing data (Not mandatory)
Synthetic Data Generators
TDF
Rule sharing
Out-of-the-box models from data & metadata
Constraint Satisfaction Problem Solver
Fabricated test data
Rules def.
Publicknowledge (Not mandatory)

Figure 2 Data Fabrication Platform flow

2.1.2 Synthetic Data Fabrication using CSP

DFP uses a solution that generates data using a CSP solver. This methodology is generic and flexible for various types of use cases yet also very safe, as all user constraints must be satisfied. The solution does not require access to real or masked data, or to historic actual queries, which all might involve some violation of privacy regulations. Data generation can be constrained directly by the users. These constraints can direct the generation towards desired testing objectives or realistic database statistics and query behaviour.

The platform uses the database schema or the file hierarchical structure, the user requirements via variables and constraints and a fabrication configuration specifying which rules to use and where to write the generated data into. The constraint-problem is solved, and the solution is used to construct the records needed to populate the database or file.

The fabrication process is described below. A user has provided a data project which contains the structure of the data, the constraint rules and the fabrication configuration. In order to construct a constraint satisfaction problem for the solver, the platform analyses the table metadata and gets table columns' data type and other properties, e.g., referential integrity constraints. The platform selects a sub-set of the relevant rules and tables using the fabrication configuration. In addition, relevant parent tables may be added due to referential integrity dependencies. Additional default rules are sometimes required as well (PK, Unique Column, string and binary column widths, etc.).

This information is used for the construction of a database table-dependency graph. For each table in that graph, starting at root nodes, structural record dependencies are built recursively.

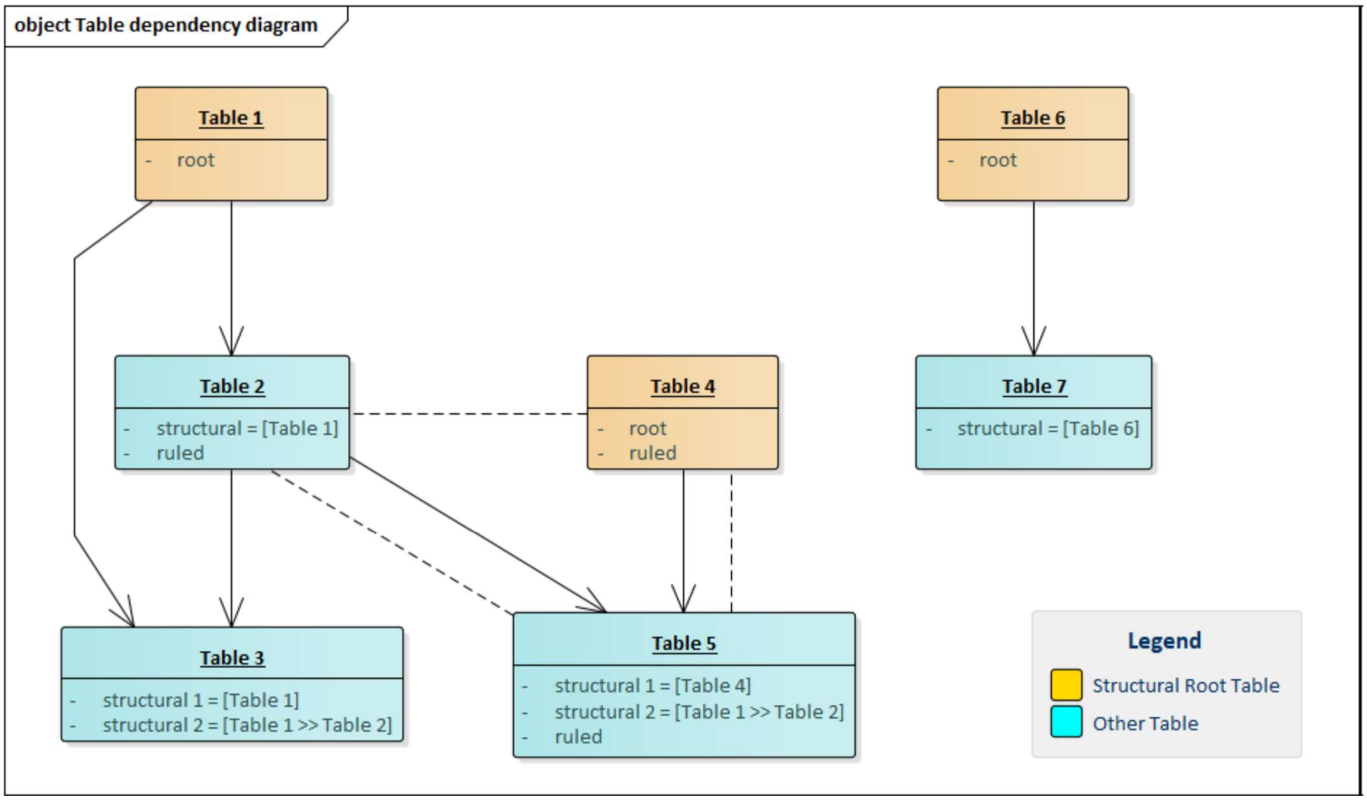

Figure 3 Table dependency diagram

Once the above graphs are built, the fabrication pattern is computed where each target table record is assigned to one of the following fabrication modes: ‘New’, ‘Reuse’, or ‘Other’.

Given the patterns, the graph and the rules, a CSP problem can be created. The CSP has a language used to model a real-world problem. A problem consists of variables and rules. The solution is an assignment of one value for each variable where all the rules are satisfied. The solver finds a random solution. Each time it solves a problem, a new solution is produced. The user does not specify how to solve the problem, but rather focuses on what to solve using the specified rules for a valid correct solution.

Each table is translated into a CSP variable struct or a CSP vector of variable structs, according to the record type, each table column is translated into a CSP variable. Each of the rules is added to the problem as a CSP constraint. A simple example of the CSP problem structure can be seen in the figure below. The CSP problem defined here is the famous eight queens puzzle (Bell and Brett, 2009).

```
variable integer arr[8];

constraint queens: forEach (q, 0, sizeOf(arr)-1, 0 <= arr[q] <= 7);
constraint sameRow: allDiff (q, 0, sizeOf(arr)-1, arr[q]);
constraint sameDiag: forEach (i, 0, sizeOf(arr)-1, \\
                        forEach (j, 0, sizeOf(arr)-1, ((i!=j) -> (abs(i-j) != abs(arr[i]-arr[j])) )) );
```

Figure 4 A CSP example

Finally, the problem is submitted to the solver. The solution is recursively parsed starting the iteration root following the topology of (vectored) record variables. In the case of a database project, an SQL insert statement for all table records is created and that SQL insert statement is submitted to the DB for execution. If streaming option is enabled and properly configured, the solution is converted to the relevant format and sent to the messaging broker specified in the configuration. In the case of file projects, the solver result is converted into the designated file format.

2.2 Extension of the DFP technology for Cyber Security domain

2.2.1 Concept

To support the THREAT-ARREST requirements, IBM Data Fabrication Platform is being enhanced with the ability to generate sequences of simulated cyber-events in general, and synthetic security events log files in particular. For such a case, the DFP needs to be properly initialized (presumably, based on definitions found in CTTP model) before a user or a client sub-system can get any synthetic log file.

First, a virtual Computer Network topology should be defined. This includes declaration of network-attached computers, switches and other relevant hardware. Each hardware node should be augmented with properties and "installed" software applications and services. User-provided rules and constraints should complement the network definition to guide the fabrication engine, how to choose values for hardware and software properties.

Then, an Event Scenario, built of connected Actions and Activities, should be defined. In case a cyber-attack log is required, an attack Scenario should be defined over the virtual Network.

After being properly configured with the Network topology and the Scenario definition, DFP creates a Constraint Satisfaction Problem based on those definitions, solves the Problem with the CSP Solver, producing pseudo-random property and function call parameter values, satisfying all the definitions, rules and constraints.

Finally, DFP simulates the Scenario, calling in application functions, declared by the Scenario actions, propagating events from one network node to another, and stores the resulting event messages down to some persistent storage, producing event log files.

2.2.2 First synthetic logs fabrication

Many operating systems, software frameworks and programs include a logging system. A widely used logging standard is syslog. Syslog originally functioned as a de facto standard, without any authoritative published specification, and many implementations existed, some of which were incompatible. The Internet Engineering Task Force documented the status quo in RFC 3164 (see (IBM, 2017)). It was later standardized by RFC 5424 (see (IBM Research, 2011)). The latter (RFC 5424) will be supported by the IBM Data Fabrication Platform and used as format for synthetic log fabrication through the THREAT-ARREST project.

Let's observe a toy-scenario of "sending a phishing email" when an attacker composes a phishing email, attaches a malicious file, and emails it over to a victim.

In such a case, a virtual Computer Network can be defined as shown in Figure 5.

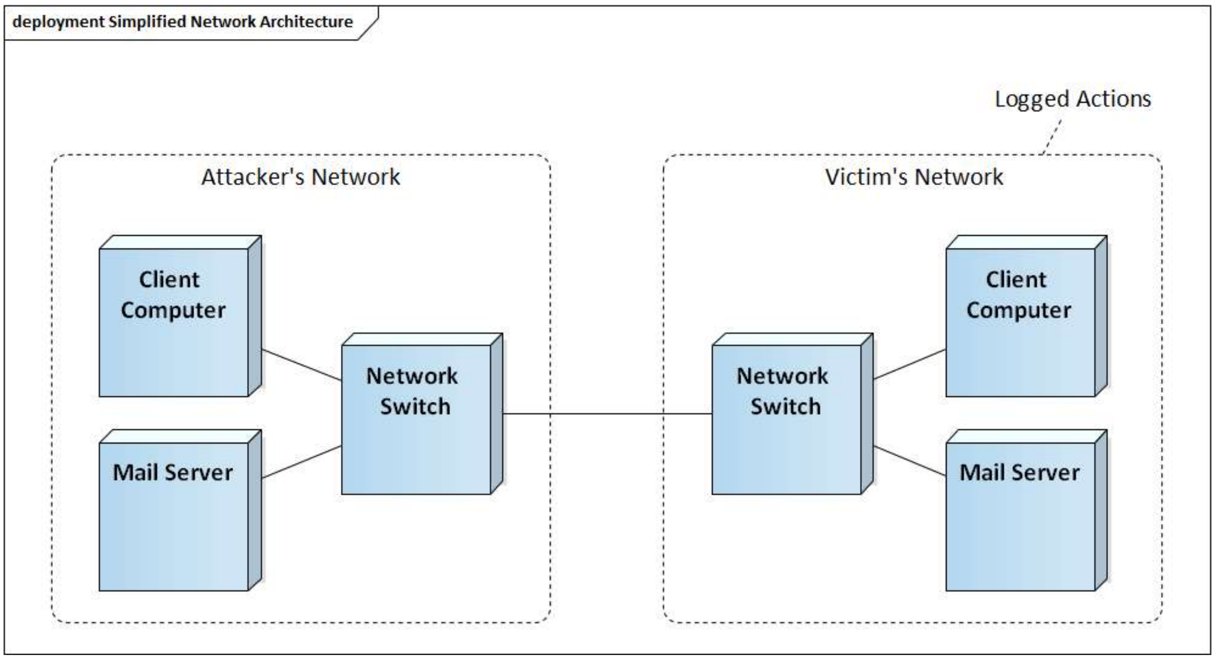

Figure 5 Network topology

Relevant properties like computer name, IP address, MAC address, email domain user name and others need to be defined for each network-attached computer. Then, these properties should be constrained to ensure that all the IPs and MACs, email domains and user names are unique, appropriate client computer and a server reside in the same sub-net cluster, and so on.

After the Computer Network topology, hardware properties and constraints are defined, one should define an attack scenario, like it is presented in Figure 6.

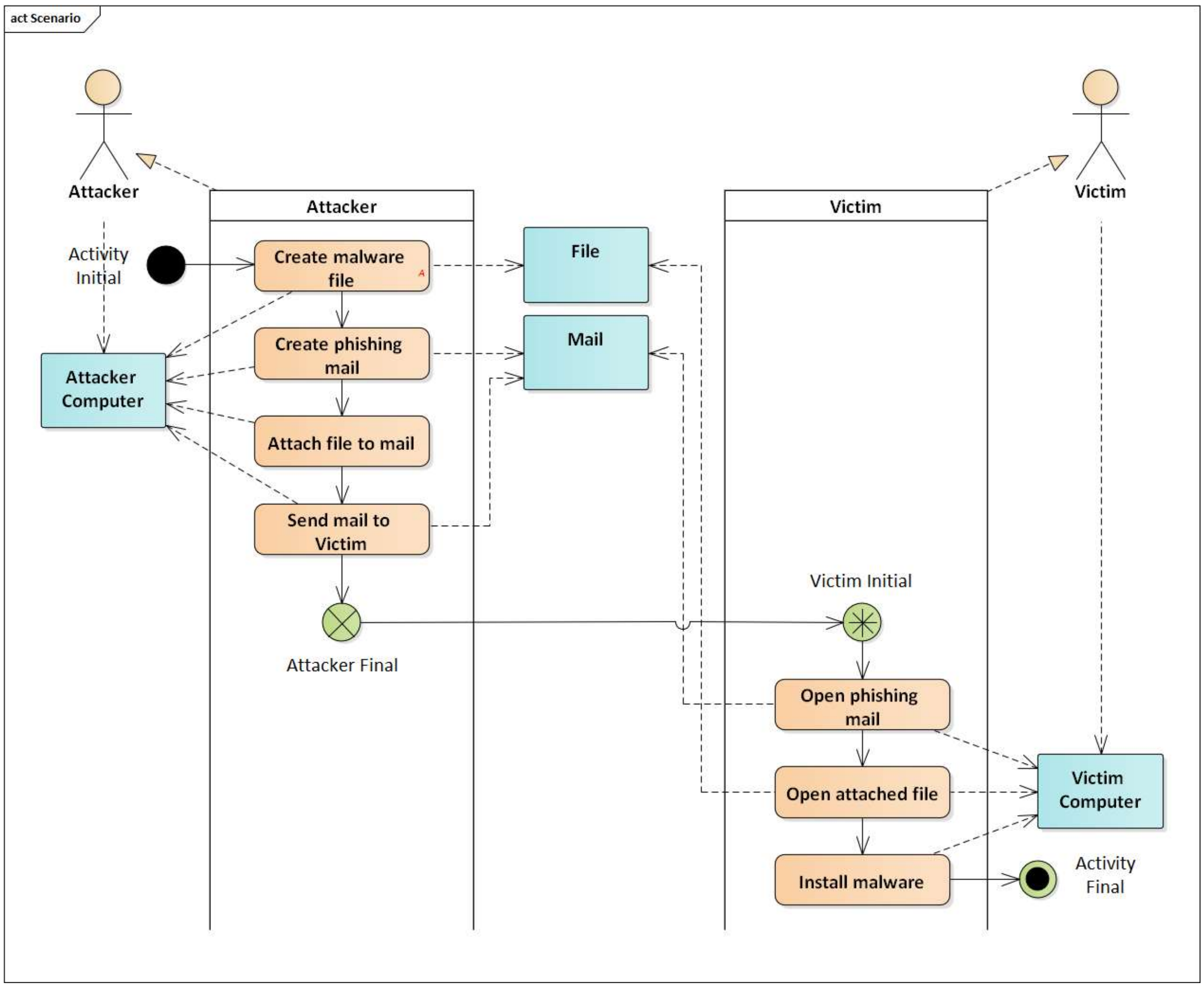

Figure 6 Attack scenario for the phishing email case

Then, after fabrication and simulation is done, one can get a desired outcome – syslog files, collected from computers on the victim's sub-network – as it is shown below.

Client Computer log

```
<22>1 2019-06-20T10:12:43.341Z client.victimnet.net IBM Mail Client App 1.0 - Login [UserInfo Username="victim"] User logged in

<22>1 2019-06-20T16:25:18.197Z client.victimnet.net IBM Mail Client App 1.0 - Receive mail [Mail Sender="attacker@attackerdomain.com" Recipient="victim@victimdomain.com" Subject="This is a phishing mail!!!" Attachment="trojan_horse.exe"] Mail received

<14>1 2019-06-20T16:28:47.813Z client.victimnet.net IBM Client Computer File System 1.0 - File save [File Filename="trojan_horse.exe"]
```

Mail Server log

```
<22>1 2019-06-20T10:12:43.683Z mailserver.victimnet.net IBM Mail Server 1.0 - Login [UserInfo Username="victim"] User logged in

<22>1 2019-06-20T16:24:53.345Z mailserver.victimnet.net IBM Mail Server 1.0 - Receive mail [Mail Sender="attacker@attackerdomain.com" Recipient="victim@victimdomain.com" Subject="This is a phishing mail!!!" Attachment="trojan_horse.exe"] Mail delivered to client
```

2.3 Integration into the THREAT-ARREST training platform

Two types of synthetic data, required for the THREAT-ARREST project, has been identified so far:

i. static general-purpose synthetic data, such as health records, for the needs of setting/performing a given training scenario;

ii. static or dynamic (interactive) security (event) logs for cybersecurity training in the context of a training scenario, such as security logs regarding malicious (anomalous) accesses to a server hosting DB of health records.

There is no need in any special DFP API to fabricate static data described in (i). Such data can be modelled in advance via DFP Web-based user interface and fabricated off-line (as described in the subsection 2.1), even before any training session starts. Fabricated data can be populated in well-known databases and/or predefined file locations to be consumed by other THREAT-ARREST components.

As for security event logs are concerned, it is suggested that IBM DFP will be enhanced (as it is described in the subsection 2.2), and the new functionality will be accessible via a log fabrication REST API.

Before one can get any security log, the DFP needs to be properly initialized and configured. Then a fabrication and simulation process should be controlled and monitored. Finally, after fabrication has been successfully completed, the fabricated log files need to be fetched out and consumed by a client sub-system. The related APIs are describing in the following subsections.

2.3.1 Cyber Network definition API

- *add/edit/delete sub-network* – functions enabling creation, modification, and removal of a sub-net (a folder-like entities, enabling grouping of network nodes and other sub-nets in a hierarchical manner)
- *add/edit/delete network node* – functions enabling creation, modification, and removal of a network node
- *add/edit/delete property* – functions enabling creation, modification, and removal of hardware or software properties (in addition to predefined ones)
- *add/edit/delete constraint* – functions enabling creation, modification, and removal of a constraint over hardware and/or software properties
- *connect/disconnect nodes* – functions enabling connection and disconnection of network nodes to create and modify a network graph of connected nodes

2.3.2 Scenario definition API

- *add/edit/delete activity* – functions enabling creation, modification, and removal of a scenario activity (a folder-like entities, enabling grouping of actions and other sub-activities in a hierarchical manner)
- *add/edit/delete action* – functions enabling creation, modification, and removal of a scenario action
- *add/edit/delete constraint* – functions enabling creation, modification, and removal of a constraint over the action function call parameters

- *link/unlink actions* – functions enabling connection and disconnection of scenario actions and activities to create and modify a scenario flow (a graph of connected scenario items)

2.3.3 Log fabrication API

- *fabricate* – a function starting the fabrication session
- *get fabrication status* – a function providing status of the current fabrication session
- *get fabrication data* – a function providing output data generated in the last session

3 Statistical Profiling of Real Event Logs

Simulation is one of the key enablers of the THREAT-ARREST framework and involves the generation of synthetic – and where necessary continuous – event streams leveraging the Data Fabrication Platform described in the previous section. The generation of such event streams is driven by the CTTP model specified for the cyber system of concern and the statistical profiling of real event logs captured from the operation of the cyber system through the assurance platform connected to the THREAT-ARREST framework. To drive the synthetic event generation process, we extend the security assurance and monitoring platform of STS to enable statistical profiling of real event logs and identify the statistical profiles of different types of individual system events, as well as different combinations of such events (e.g. (Hatzivasilis et al., 2019a; Hatzivasilis et al., 2019b)). These can be altered to reflect anomalous system conditions including, for example, atypical usage conditions, suspicious system or system component operation conditions, the launch of security attacks upon cyber systems and their components, or unexpected system operation effects (e.g. (Hatzivasilis et al., 2019c; Hatzivasilis et al., 2019d; Hatzivasilis et al., 2017)). The profiling and analysis of these events also enables the detection of (and provision of training against) attacks with different temporal characteristics, including instantaneous, long-lived and repeating attacks. The operation of this mechanism is driven by (and thus requires) simulation-related specifications in the CTTP model.

Furthermore, the operational system evidence formed by the collected monitoring events and testing outcomes of the assurance tool, are passed over to the statistical profiling module and thereby the generation of realistic simulations is performed. Statistical profiling also covers event metadata (e.g., the timing of their occurrence and other characteristics such as their sender and receiver) and – where allowable by the applicable security policies – the actual event payload (e.g., data passed between components, parameter values of component operation calls, size of files read from or written to secondary storage and related directories, etc.).

The statistical profiling module is a tool written in the Python programming language. In version 1 (v1), the tool accepts as input a file that contains security event logs in XML format, provided by the assurance tool. It parses the file in order to identify the element tree and stores the found information in data structures (lists, arrays, etc.), along with extra information helpful for the implementation. Then, it iterates through said data structures and performs a statistical analysis for all data discovered.

```
================= Statistical analysis =================
Timestamp
2019-5-28 15:56:26 -> 1 occurrence(s) -> 25.00% of total
2019-5-29 01:06:26 -> 1 occurrence(s) -> 25.00% of total
2019-5-28 15:58:37 -> 1 occurrence(s) -> 25.00% of total
2019-5-29 01:08:06 -> 1 occurrence(s) -> 25.00% of total

Notifier
195.41.582.14 -> 4 occurrence(s) -> 100.00% of total

Sender
127.0.0.1 -> 4 occurrence(s) -> 100.00% of total

Receiver
192.168.43.26 -> 4 occurrence(s) -> 100.00% of total

Event payload
readOp -> 2 occurrence(s) -> 50.00% of total
writeOp -> 2 occurrence(s) -> 50.00% of total

Event parameters
ser-0177 -> 2 occurrence(s) -> 50.00% of total
ser-2142 -> 2 occurrence(s) -> 50.00% of total
DownloadRecord -> 2 occurrence(s) -> 50.00% of total
UploadRecord -> 2 occurrence(s) -> 50.00% of total
========================================================
```

Figure 7 Output of the statistical analysis module

The analysis reveals a series of events along with their metadata including the time they occurred, who notified about the event, who was the sender/receiver, what was the operation being performed (payload) and the parameters of this payload. Additionally, the tool displays the number of occurrences of each piece of data, along with the percentage of each one in relation to the total amount of events. Figure 7 Output of the statistical analysis moduleFigure 7 Output of the statistical analysis module

shows the output of the module. This information can be utilized by the aforementioned phishing scenario (Section 2) in order to generate more realistic data for normal or malicious interaction with the trainee's/victim's network (Ferrera et al., 2018; Cesena et al., 2017).

4 Conclusions

This document provides a first report on designing and developing of two major components of the THREAT-ARREST training platform: (1) the generator of synthetic event logs and (2) the module that will carry out the statistical profiling of real event logs.

To support the THREAT-ARREST requirements, IBM Data Fabrication Platform is being enhanced with the ability to generate sequences of simulated cyber-events in general, and synthetic security events log files in particular. The tool will be used to fabricate two types of synthetic data:

i. static general-purpose synthetic data, such as health records, for the needs of setting/performing a given training scenario;

ii. security event logs for cybersecurity training in the context of a training scenario, such as security logs regarding malicious (anomalous) accesses to a server hosting DB of health records.

The new functionality will be accessible via a log fabrication REST API.

To improve the accuracy of synthetic event generation process, the project security assurance and monitoring platform is extended to enable statistical profiling of real event logs and identify the statistical profiles of different types of individual system events, as well as different combinations of such events.

The profiling and analysis of these events also enables the detection of attacks with different temporal characteristics. The operation of this mechanism is driven by simulation-related specifications in the CTTP model. The statistical profiling module is a tool written in the Python programming language.

This deliverable, along with the deliverables D2.1-D2.4, D3.1, D4.1-D4.4, and D5.2-D5.3, is part of the MS2 regarding the 1st version of the various THREAT-ARREST components. These tools will be further extended and integrated during the 2nd year of the project.

5 References

[1] "Create high-quality test data while minimizing the risks of using sensitive production data." *IBM InfoSphere Optim Test Data Fabrication*, IBM, 2017, https://www.ibm.com/il-en/marketplace/infosphere-optim-test-data-fabrication.

[2] "Test Data Fabrication." *Security and Data Fabrication*, IBM Research, 2011, https://www.research.ibm.com/haifa/dept/vst/eqt_tdf.shtml.

[3] "Constraint Satisfaction." IBM Haifa Research, IBM, 2002, https://www.research.ibm.com/haifa/dept/vst/csp.shtml.

[4] Cesena, M., et al. 2017. SHIELD Technology Demonstrators. CRC Press, Book for Measurable and Composable Security, Privacy, and Dependability for Cyberphysical Systems, pp. 381-434.

[5] Ferrera, E., et al., 2018. IoT European Security and Privacy Projects: Integration, Architectures and Interoperability. CRIStin – SINTEF, Next Generation Internet of Things. Distributed Intelligence at the Edge and Human Machine-to-Machine Cooperation. Book Chapter 7, pp. 207-292.

[6] Hatzivasilis, G., et al., 2019a. Towards the Insurance of Healthcare Systems. 1st Model-driven Simulation and Training Environments for Cybersecurity (MSTEC), ESORICS, Springer, LNCS, vol. 11981, Luxembourg, 27 September 2019, pp. 1-14.

[7] Hatzivasilis, G., et al., 2019b. Cyber Insurance of Information Systems. 24th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2019), IEEE, Limassol, Cyprus, 11-13 September 2019, pp. 1-7.

[8] Hatzivasilis, G., et al., 2019c. WARDOG: Awareness detection watchdog for Botnet infection on the host device. IEEE Transactions on Sustainable Computing – Special Issue on Sustainable Information and Forensic Computing, IEEE, vol. 4, pp. 1-15.

[9] Hatzivasilis, G., et al., 2019d. MobileTrust: Secure Knowledge Integration in VANETs. ACM Transactions on Cyber-Physical Systems – Special Issue on User-Centric Security and Safety for Cyber-Physical Systems, ACM, vol. 4, issue 3, Article no. 33, pp. 1-15.

[10] Hatzivasilis, G., et al., 2017. SCOTRES: Secure Routing for IoT and CPS. IEEE Internet of Things Journal (IoT), IEEE, vol. 4, issue 6, pp. 2129-2141.

[11] Y. Richter, Y. Naveh, D. L. Gresh, and D. P. Connors (2007), "Optimatch: Applying Constraint Programming to Workforce Management of Highly-skilled Employees", International Journal of Services Operations and Informatics (IJSOI), Vol 3, No. 3/4, pp. 258 - 270.

[12] Y. Naveh, Y. Richter, Y. Altshuler, D. Gresh, and D. Connors (2007), "Workforce Optimization: Identification and Assignment of Professional Workers Using Constraint Programming", IBM J. R&D.

[13] Y. Naveh, M. Rimon, I. Jaeger, Y. Katz, M. Vinov, E. Marcus, and G. Shurek (2006), "Constraint-Based Random Stimuli Generation for Hardware Verification", AI magazine Vol 28 Number 3.

[14] E. Bin, R. Emek, G. Shurek, and A. Ziv (2002). "Using a constraint satisfaction formulation and solution techniques for random test program generation", IBM Systems Journal, 2002.

[15] Merav Aharoni, Odellia Boni, Ari Freund, Lidor Goren, Wesam Ibraheem, Tamir Segev (2015), "Rectangle Placement for VLSI Testing", CPAIOR 2015: 18-30

[16] O. Boni, F. Fournier, N. Mashkif, Y. Naveh, A. Sela, U. Shani, Z. Lando, A. Modai (2012) "Applying Constraint Programming to Incorporate Engineering Methodologies into the Design Process of Complex Systems" Proceedings of the Twenty-Fourth Conference on Innovative Applications of Artificial Intelligence, Toronto, Ontario, Canada. AAAI 2012.

[17] Y. Ben-Haim, A. Ivrii, O. Margalit and A. Matsliah (2012) "Perfect Hashing and CNF Encodings of Cardinality Constraints", SAT 2012, Trento, Italy.

[18] E. Bin, O. Biran, O. Boni, E. Hadad, E. K. Kolodner, Y. Moatti, D. H. Lorenz (2011), "Guaranteeing High Availability Goals for Virtual Machine Placement", ICDCS 2011.

[19] J. Shin, J. A. Darringer, G. Luo, M. Aharoni, A. Y. Lvov, G. Nam, M. B. Healy (2011), "Floorplanning challenges in early chip planning", SOCC Conference, 2011 IEEE International, pp. 388—393

[20] Y. Naveh (2010). "The Big Deal, Applying Constraint Satisfaction Technologies Where it Makes the Difference". Proceedings of the Thirteenth International Conference on Theory and Applications of Satisfiability Testing (SAT'10).

[21] S. Asaf, H. Eran, Y. Richter, D. P Connors, D. L. Gresh, J. Ortega, M. J. Mcinnis (2010). "Applying Constraint Programming to Identification and Assignment of Service Professionals". Accepted for presentation in The 16th International Conference on Principles and Practice of Constraint Programming (CP2010). The paper received the Best Application Paper Award.

[22] B. Dubrov, H. Eran, A. Freund, E. F. Mark, S. Ramji, and T. A. Schell, (2009). "Pin Assignment Using Stochastic Local Search Constraint Programming" in Proceedings of the 15th International Conference on Priniciples and Practice of Constraint Programming (CP'09), Edited by Ian P. Gent, pp 35-49.

[23] Y. Richter, Y. Naveh, D. L. Gresh, and D. P. Connors (2007), "Optimatch: Applying Constraint Programming to Workforce Management of Highly-skilled Employees", IEEE/INFORMS International Conference on Service Operations and Logistics, and Informatics (SOLI), Philadelphia, pp. 173-178.

[24] S. Sabato and Y. Naveh (2007), "Preprocessing Expression-based Constraint Satisfaction Problems for Stochastic Local Search", Proceedings of The Fourth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR).

[25] Y. Naveh, M. Rimon, I. Jaeger, Y. Katz, M. Vinov, E. Marcus, and G. Shurek (2006), "Constraint-Based Random Stimuli Generation for Hardware Verification", IAAI 2006.

[26] Y. Richter, A. Freund, and Y. Naveh (2006), "Generalizing AllDifferent: The SomeDifferent constraint", Proceedings of the 12 International Conference on Principles and Practice of Constraint Programming - CP 2006, Lecture Notes in Computer Science, Volume 4204, pages 468-483.

[27] Y. Naveh and R. Emek (2006). "Random stimuli generation for functional hardware verification as a CP application - a demo", IAAI 2006.

[28] Y. Naveh (2005). "Stochastic solver for constraint satisfaction problems with learning of high-level characteristics of the problem topography" CP 2005

[29] F. Geller and M. Veksler (2005), "Assumption-based pruning in conditional CSP", in van Beek, P., ed., CP, "Principles and Practice of Constraint Programming - CP 2005" of Lecture Notes in Computer Science (3709), 241-255 Springer.

[30] R. Dechter, K. Kask, E. Bin, and R. Emek (2002). "Generating random solutions for constraint satisfaction problems", AAAI 2002.

[31] D. Lewin, L. Fournier, M. Levinger, E. Roytman, G. Shurek (1995). "Constraint Satisfaction for Test Program Generation", Internat. Phoenix Conf. on Computers and Communications, March 1995.

[32] J. Bell, S. Brett (2009). "A survey of known results and research areas for n-queens", Discrete Mathematics, vol. 309, issue 1, pp. 1-31.