Security Analytics Framework Validation based on Threat Intelligence

Viktor Sowinski-Mydlarz, Vassil Vassilev, Karim Ouazzane and Anthony Phipps

London Metropolitan University Cyber Security Research Centre, Tower Building, 166-220 Holloway Road,
{w.sowinskimydlarz, v.vassilev, k.ouazzane, tsaphip1}@londonmet.ac.uk

Abstract – Logical analysis of the ontology of digital security in banking helps us to identify the possible entry points for illegal access. The threats described in the ontology are detected by Machine Learning engines. The theoretical analysis is validated by verifying the framework and Machine Learning algorithms. Intelligence Graphs (original term) which are adding the actions to knowledge graphs to form workflows, are a base for validation of the framework through simulated execution of the scenarios specified in them.

The output is a method for analysing live network traffic data (machine learning algorithm) combined with semantic model to give a hybrid framework for threat intelligence in digital banking, leading to a complete threat detection platform. The model is validated using operation workflows, namely 12 scenarios of banking "journeys" under the duress of various threats.

In this work we are presenting the validation of the framework by simulation of the banking operations and transactions stemming from the Ontology of Digital Banking used as a model of the banking infrastructure (assets, vulnerabilities and threats included).

For better understanding of the subject matter, the authors would like to refer the reader to a previous article on general framework we developed.

Keywords – Security Analytics, Threat Intelligence, Machine Learning, Digital Banking, Knowledge Graphs

1. Introduction.

Scenario-based Methodology for Validation of Frameworks combines the methodology for validating of the logical model through simulation, with the methodology for verifying the methods and algorithms for detection/classification/prediction through use of simulated, synthetic, historical, or real data. The simulation is of the framework in action, which requires only the analytical model and the emulation of events and actions without actually executing them. Core of this methodology is what we are calling "Intelligence Graphs". Here we describe validating the framework using scenarios for executing controlled transactions under threat. Scenario testing is conducted employing scenarios obtained from the use cases. Using scenario testing, complex logic can be tested applying easy to evaluate statements describing the functionality to be tested. The journey through the banking processes under the duress of cyber threats involves entities such as events (which are asynchronous) and rules affecting the flow of data. Actions (describing transitions between situations) are decided on by either events, rules, or user input. The test procedure is to start with initial situation (normal or abnormal, in case of vulnerability) and proceed through intermediate states where threats manifest, affecting the banking procedures leading either to a deadlock or normal situation after the threat has been mitigated. Running analytics and applying corrective (COR) actions helps avoiding deadlocks. The framework model which combines the ontology with the security policies for incorporating threat intelligence has been validated using a representative set of scenarios for executing transactions under threat. There have been 12 scenarios designed for the validation of the framework.

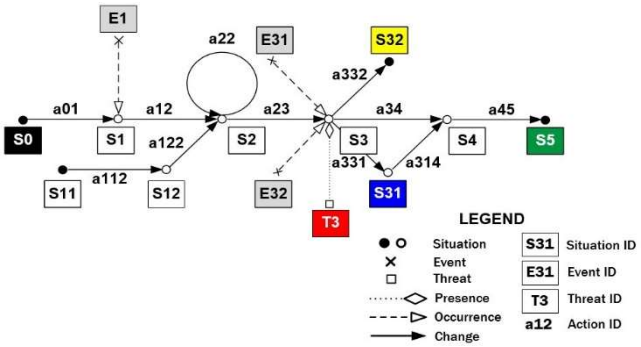

FIGURE 1. INTELLIGENCE GRAPHS FOR MODELLING TRANSACTIONS UNDER THREAT.

2. Literature Review.

2.1. Attack trees as foundation for knowledge graph in our approach.

Schneier's attack tree model paved the way for using graphs in threat intelligence. [2] Attack tree is a conceptual diagram describing how assets (targets) can be attacked. It is multi levelled with a root node, children nodes and leaves. Children nodes represent conditions to be satisfied for parent nodes to be true. If the root is satisfied, the attack is successful. Nodes are satisfied by their direct child nodes. For instance, if there is one grandchild node under the root, the steps need to be taken in the process of the attack – satisfying grandchild node conditions for the parent node to be true, then its parent for root node to be true. Alternatives are represented by AND / OR operators. Attack trees can give a false sense of security as overlooking an attack vector becomes easy. The results are reliant on the initial cost estimates, which are often inaccurate. Attack trees do not consider secondary factors. The dashboard developed for our framework serves not only as a graphical user interface but also constructs an interactive dynamic graph. This graph is not an attack

tree per se, it serves the same purpose of describing an attack though. The difference is that our graph is not hierarchical as an attack tree.

2.2. Elements of threat modelling relevant to our model structure.

Threat modelling allows to identify, categorize, convey, and analyse threats. It helps to mitigate risks and protect the infrastructure by improving security. It takes place in planning, design, and implementation phases. Modelling lets the developers better comprehend their software and its components. This knowledge quantifies positive and negative impact on the system. The number of resources (time constraints, financial) needs to be proportional to the possible risks. The model is the basis for the security decisions and policies used to safeguard the product. There are several benefits coming from keeping security at the vanguard of development through threat modelling. Continuous monitoring of threats and an up-to-date threat profile can mitigate risks. Threat intelligence is crucial for delivering considerable protection and secure coding. [3] The elements of threat modelling are:

- Assets: Data and hardware to be protected.
- Threats: The way hacker can affect the system.
- Vulnerabilities: The gaps in the security allowing the hacker in. The steps in the process can be listed as:

- Identifying the assets.
- Outlining architectural context of the asset being processed. It may include the software framework, version, and other architectural details.
- Breaking down application and its processes, including all the sub-processes that are running the application. Data flow diagram (DFD) is created.
- Identifying and listing threats in a descriptive way.
- Classifying the threats (parallel instances) to identify in the application in a structured and repeatable manner.
- Rating the severity of the threat.

Threat modelling should be performed in case of a change in the system's architecture and as soon as the architecture is ready. Also, it should be run after a security incident has occurred or new vulnerabilities are introduced. It is not required to carry out threat modelling at the early stages of the SDLC (Software Development Life Cycle).

TABLE 1. THREATS IN TYPICAL SCENARIOS OF ONLINE TRANSACTIONS.

Scenario	Threat(s)	Threat Present In	Potential Dead-end or Normal Situation
Update with Spyware	Spyware / Baiting	S2_Infected_Attachmnt S23_Infected_Software	S5_Normal_State
Login with Spyware	Spyware	S0_Browser_Started	S5_User_Logged_In
Transfer with Spyware	Spyware	S13_Infected_Malware	S10_User_Logged_Out
Balance with DdoS	DdoS Attack	S0_User_Logged_In	S4_User_Logged_Out
View Balance Quid Pro Quo	Quid ProQuo	S12_Support_Imitated	S8_User_Logged_Out
Vishing & Smishing	Vishing &Smishing	S0_Normal_State	S5_Payment_To_Crim
Session Hijacking	Session Hijacking	S2_User_Authentctd	S5_Card_Removed
Email Spam Received	Email Spam	S0_Spam_Received	S9_Machine_Overtaken
Cross channel with Pretext	Pretexting	S0_IT_Support_Imitd	S5_Payment_To_Crim
Scareware & Rogue	Scareware,Rogue	S1_Infection_Simulatd	S6_Machine_Overtaken
ATM Infected	ATM Infected	S0_ATM_Not_Updated	S10_Payment_To_Crim

2.3. Knowledge Graphs as combination of Semantic Web and Machine Learning.

A model of a knowledge domain designed by specialised experts using intelligent machine learning algorithms constitutes a knowledge graph. [4] The terms "knowledge graph" and "knowledge base" are often used interchangeably, leading to a wrong assumption that they are synonymous. They can also be used as synonyms of ontology. Google Knowledge Graph is known as a knowledge base, same as YAGO ("Yet Another Great Ontology"). [5] The creators rarely differentiate between ontology, knowledge graph and knowledge base. The latter is built by associating entities, relations, and data with the ontology. Incomplete, incoherent, and imprecise information transforms into a complete, integrated, and accurate knowledge graph. Ontology can be defined as a set of

axioms defining knowledge in a specific domain. It can be visualized as a graph and describes complex relationships between classes and individuals.

Knowledge graphs are also known as semantic networks. They represent interconnected entities (such as situations, events, items, and actions in our ontology). In fact, knowledge graphs are similar to ontologies. The knowledge is visualized as a graph. The knowledge graphs are a different viewpoint at the same reality (the ontology) but it is convenient for two reasons - because it can be visualized, and because it can be used for contextual focusing on fragments of the ontology which combine only some classes (nodes of the graph) and relevant properties (edges of the graph). This is particularly important because we are going to model scenarios for validation of the framework in the form of intelligence graphs. [6]

3. Validation procedure.

3.1. The validation of the framework using scenarios.

The twelve scenarios correspond to twelve mini ontologies derived from the main ontological model, keeping the same format (OWL/RDF). They have been developed using a graph language (RDF), and they are related to the two levels of the framework (the ontological and logical level). It means that besides OWL logic, also SWRL rules apply to all scenarios. The context of the ontological classes was used for automatic generation of the security policies and control over the simulation. The OWL ontology axioms were interpreted into SWRL rules format. SWRL was embedded into the ontology and was extracted from relationships between classes. To recognize that the relationship between classes entails a rule, the class is checked if it belongs to situations and that the object property is an action. The head of the rule contains the resulting situation. The automated rule generation facility is still work in progress. The essence of the method is to combine the ontology with the policies in a purely relational model (the graph does not have taxonomic relations).

3.2. Methodology.

The testing methodology deviates from the software engineering methodology by three points:

- It requires building models of the transactions under threat using the modelling languages of the framework (the Intelligence Graphs).
- It requires validating the models by experts (i.e., the Lloyds bank professionals who approved the graphs).
- It is a part of a collaboration combining model validation and functional testing with integration testing using components belonging to colleagues (namely dashboard designed by Dr Paweł Gąsiorowski) and by simulating other components which are still to be implemented real-time event notification).

It is worth mentioning that the framework as it is now incorporates one manual component, namely decision making for executing particular operation in a given situation in the presence of alternatives. In the future this can be automated by adding an algorithm for decision making based on risk assessment, now the framework is interactive and requires user intervention though.

TABLE 2. SCENARIOS FOR VALIDATION.

Scenario	Starting Situation	Threat(s)	Threat First Present In	Final Situation	Potential Deadlock
Update Antivirus with Spyware	S0_Antivirus_Not_ Updated	Spyware Baiting	S2_Infected_Attachment S23_Infected_Software	S5_Normal_State	
Login with Spyware	S0_Browser_Started	Spyware	S0_Browser_Started	S5_User_Logged_In	S6_Login_Refused S45_Credentials_Stolen
Money Transfer with Spyware	S0_Browser_Started	Spyware	S13_Infected_With_ Malware	S10_User_Logged_Out	S17_Site_Maintenance S18_Disconnected …
Balance with DDoS	S0_User_Logged_In	DDoS Attack	S0_User_Logged_In	S4_User_Logged_Out	S10_Operation_Cancelled
View Balance with Quid Pro Quo	S0_Browser_Started	Quid Pro Quo	S12_IT_Support_Imitated _By_Hacker	S8_User_Logged_Out	S25_Site_Maintenance S16_Disconnected S22_Invalid_Credentials
Vishing & SMishing	S0_Normal_State	Vishing Smishing	S0_Normal_State	S5_Payment_To_Criminals	S8_Transaction_Cancelled
Withdrawal with Session Hijacking	S0_Card_Inserted	Session Hijacking	S2_User_Authenticated	S5_Card_Removed	
Sending Email Spam	S0_Machine_Overtaken _By_Hacker	Email Spam	S0_Machine_Overtaken _By_Hacker	S6_Spam_Received	S2_System_Monitored
Email Spam Received	S0_Spam_Received	Email Spam	S0_Spam_Received	S9_Machine_Overtaken _By_Hacker	S5_Vital_Data_Extorted
Cross Channel with Pretexting	S0_IT_Support_ Imitated_By_Hacker	Pretexting	S0_IT_Support_Imitated _By_Hacker	S5_Payment_To_Criminals	S8_Transaction_Cancelled S9_Account_Locked S10_Credentials_Stolen
Scareware & Rogue	S0_Normal_State	Scareware, Rogue, Spyware	S1_Infection_Simulated	S6_Machine_Overtaken _By_Hacker	
ATM Infected	S0_ATM_OS_Not_Updat	ATM Infected	S0_ATM_OS_Not_Updated	S10_Payment_To_Criminals	S9_Transaction_Cancelled

The twelve diagrams in Section 4.1. Validation Scenarios are based on extracts from the main ontology, using the same description logic as the main model does for representing real life situations. Each of these Intelligence Graphs represents a banking journey in the presence of one or more threats, starting with initial situation which can be “internet browser started” for example, with following situations of connecting to online banking, logging in, and selecting operation. Initial situation is represented in green for a typical starting state and red for the abnormal. Various events happen on the way, triggering actions directing the diagram flow. Events are shown in blue on the diagrams. The situations which are final and do not lead to other situations are so called “deadlocks” and are presented in red. Examples of such a situation are “disconnected” and “invalid credentials”. The journey happens under the duress of threat which is shown in black. Items in yellow are generic entities from the context of situations, they are simple classes connected to them in the ontology. Example would be antivirus software, session ID, amount of money extorted by criminals.

3.3. Source of scenarios.

The twelve scenarios correspond to twelve mini ontologies derived from our main ontological model, keeping the same format (OWL/RDF). They have been developed using a graph language (RDF), and they are related to the two levels of the framework (the ontological andlogical level). It means that besides OWL logic, also SWRL rules apply to all scenarios. The context of the ontological classes can be used for automatic generation of the security policies and control over the simulation. The OWL ontology axioms can be interpreted into SWRL rules format. SWRL is embedded into the ontology and can be extracted from relationships between classes. To recognize that the relationship between classes entails a rule, we need to check if the class belongs to situations and that the object property is an action. The head of the rule contains the resulting situation.

4. Experimental Framework Validation.

The two-dimensional nature of the framework requires a complex validation process, which cannot be finalized completely since it remains open for addition of more analytical engines and software components. Because of this complexity, we will demonstrate only a partial validation of the framework, selecting relevant components on the different levels of the framework. The scenarios of controlled execution of the transactions under security threat, which allow both analytical verification using dry run of the scenarios as well as actual testing in simulation mode, will play an integral role in this procedure.

4.1. Validation Scenarios.

The policy rules represent only a fragment of the actual policy of organisations and do not pretend to be complete. Their formulation has been dictated by the need to represent the working scenarios. In principle, it is possible to validate the rules formally by applying the procedure for checking the satisfiability of Horn clauses in clausal logic, which is the theoretical model of SWRL, but this would be a long, repetitive, and unnecessarily complex process without much practical value. Instead of this, the validation of the rules has been done by traversing the ontology for determining if there are applicable rules, which cannot be fired in any situation represented in the ontology due to impossibility to satisfy the restrictions in the conditional part, and rules, which do not lead to any situation in the ontology, prescribed in the consequent part of the rules. Additionally, they have been consulted with experts from the Cyber Security Division of Lloyds Banking Group in London, which have been advising the Cyber Security Research Centre during working on the project for Analysing the Logical Vulnerability.

The rule validation is founded on 1) analytical formulation, based on literature investigation 2) expert confirmation, based on Lloyds bank professional’s approval and 3) experimental verification by scenarios simulation. The last process can be treated as a dry run of the scenarios, mentioning which rules are fired (satisfy the conditions) and how the scenarios continue after they are fired (where do they go after the consequent part of the rule is executed). Description of each of twelve cases for simulation gives reasons why it has been chosen.

Situation in the diagram is denoted by an empty circle ○ for starting situation and by solid circle ● for final situation. Event is symbolized by a square □. The symbol for threat is a diamond ◊ and generic items are represented by a triangle △. Solid line arrow → signifies an action, dashed arrow ⇢ occurrence, dotted stem arrow ⇾ intervention. Open headed arrow ⟶ stands for having a subclass. Green rectangle means a starting situation ID whilst red rectangle is a final situation ID. Blue rectangle is an event ID, black is a threat ID and yellow is an item name. Situation IDs begin with prefix “S_” and a number, event IDs begin with prefix “E_” and a number. Actions’ prefix is an “a_” and a number constructed from index of domain situation and index of range situation. Situations, events, threats, items, and actions are all ontology concepts.

1. Update Antivirus with Spyware scenario shows the journey from initial situation of antivirus not being updated causing vulnerability leading to Spyware and Baiting threats initiating. This basic set-up relates to a common threat of spyware stemming from not using updated antivirus that can affect anyone, not necessarily in banking IT infrastructure. This is a starting point in many intrusions.

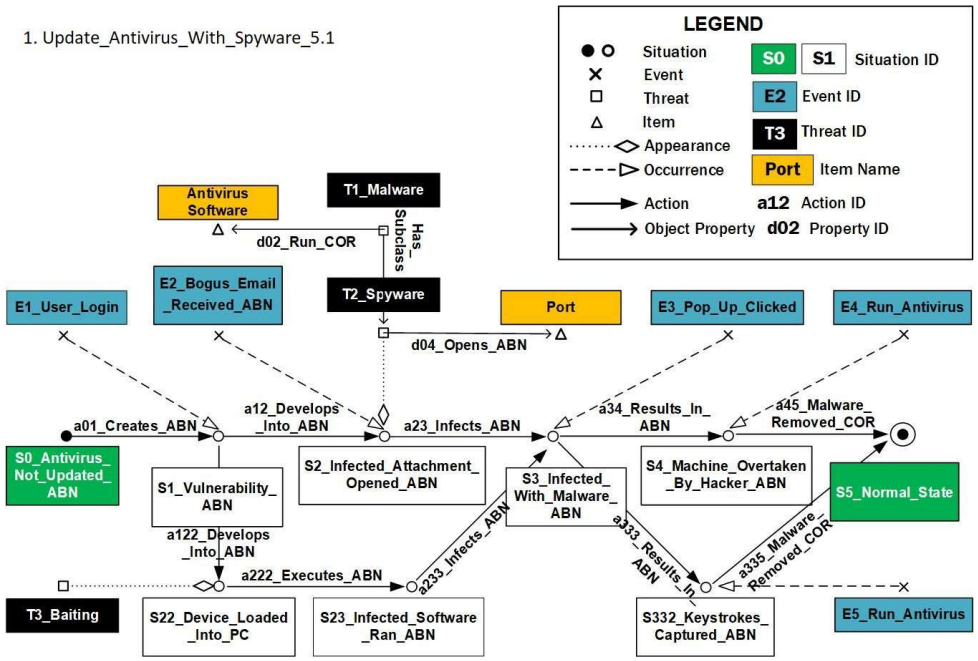

FIGURE 2. UPDATE ANTIVIRUS WITH SPYWARE.

2. Login with Spyware describes logging in in the presence of spyware threat, starting with opening the browser and ending with login either successful or refused. This is the next step in the intrusion process. Logging in is an everyday activity for any user and it can result in credentials being stolen.

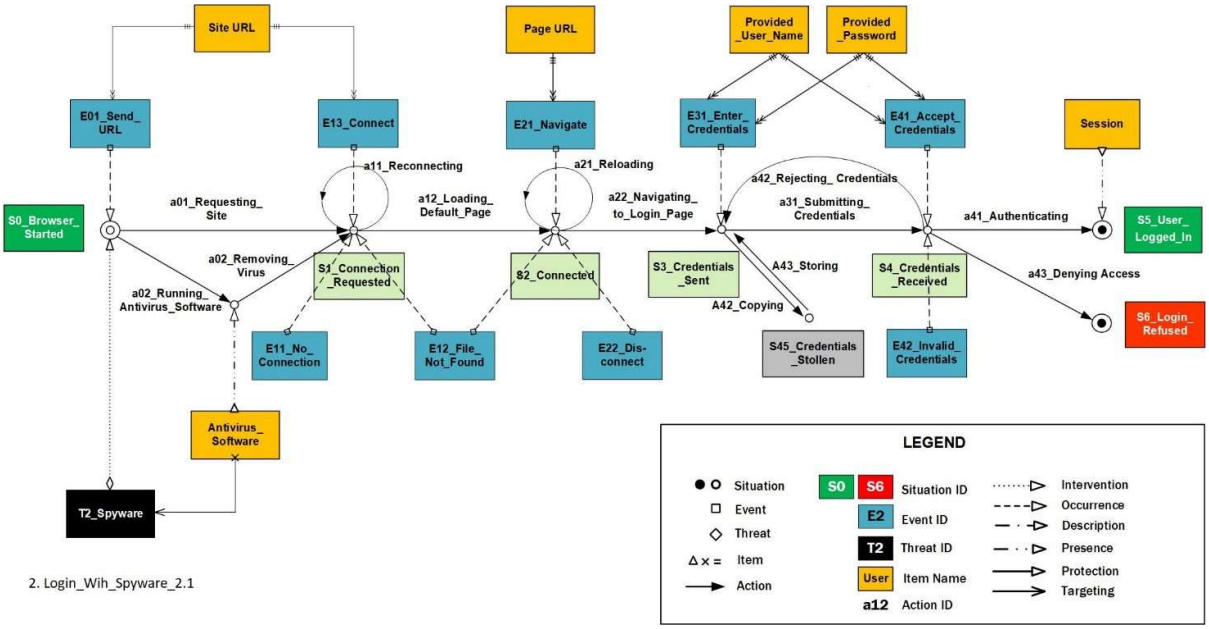

FIGURE 3. LOGIN WITH SPYWARE.

3. Money Transfer with Spyware shows extortion of money by the means of keylogging. After logging in, banking operation of money transfer is chosen, and this case shows how it can be affected by and misused by fraudsters.

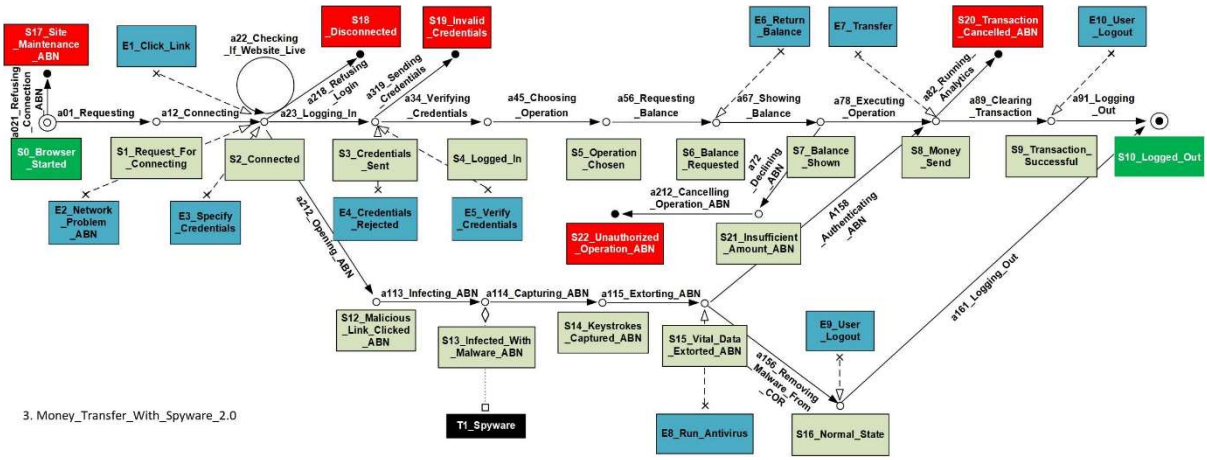

FIGURE 4. MONEY TRANSFER WITH SPYWARE.

4. View Balance with dDOS describes attempts at checking the account balance in case where server and network are crashed by Denial-of-Service attack. dDOS attacks are most common threats which financial organisations must deal with. Hence, we show how it affects banking operations (viewing balance in this example).

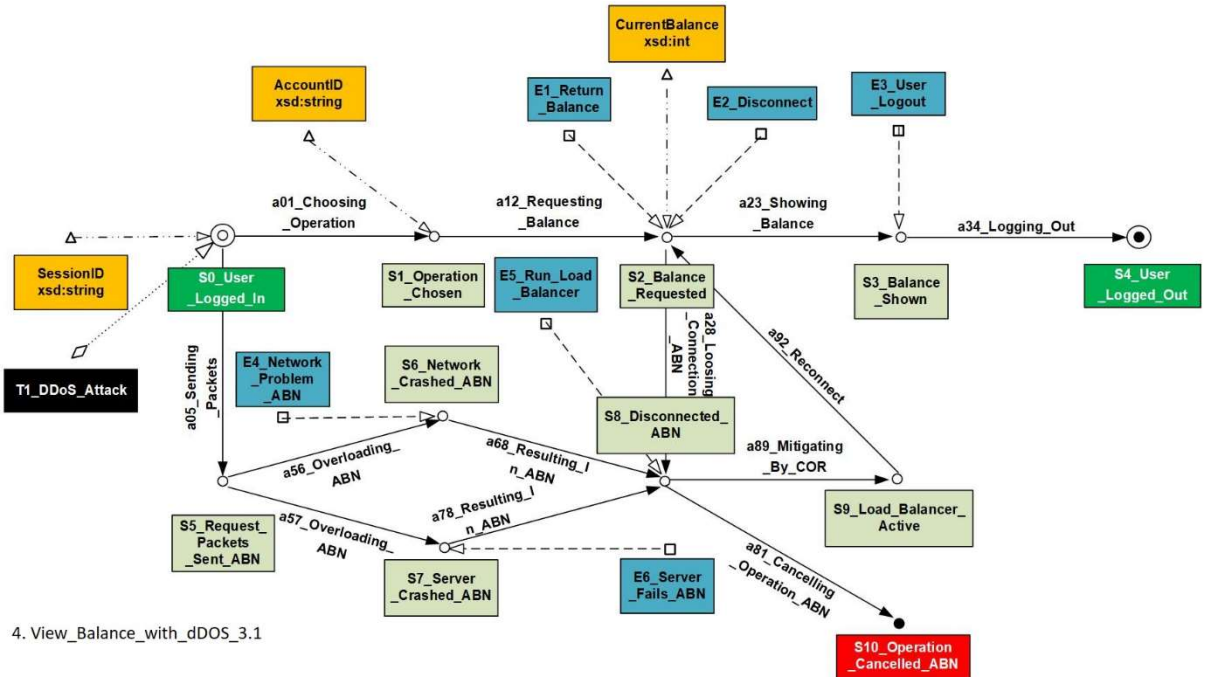

FIGURE 5. VIEW BALANCE WITH DDOS.

5. View Balance with Quid Pro Quo is like the previous one, this time threat is the hacker overtaking the PC by pretending to be customer service. Fraudsters often use this method to gain control over user's machine in which case viewing balance can lead to stealing credentials.

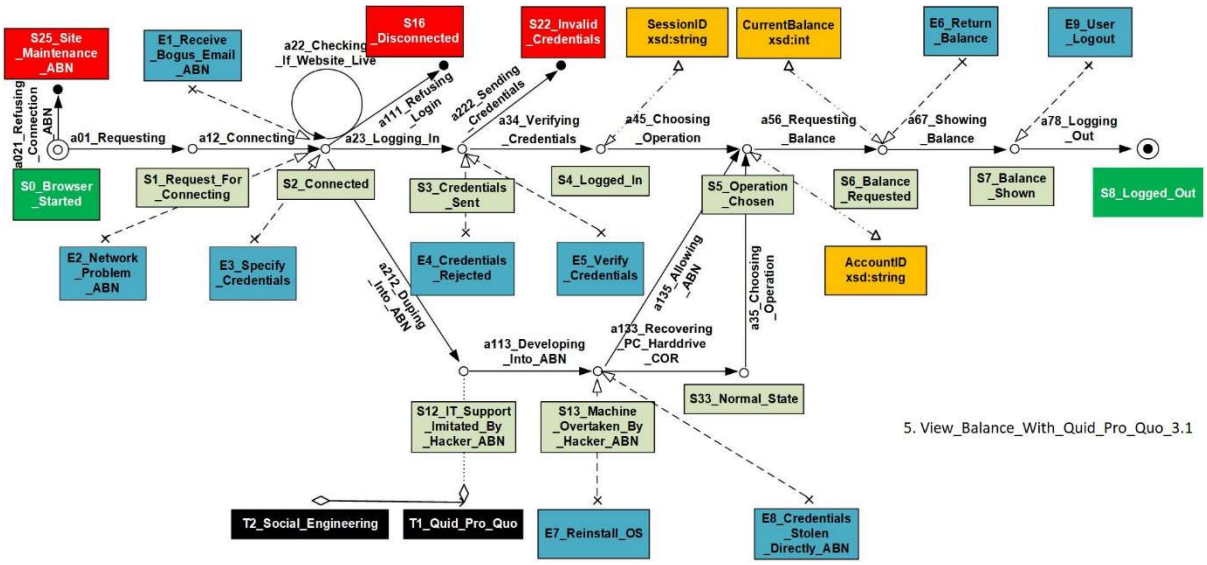

FIGURE 6. VIEW BALANCE WITH QUID PRO QUO.

6. Vishing & SMishing are voice phishing (over the telephone) or SMS phishing. Hoaxed phone number is called in case of vishing or malicious link is clicked in the SMS. These are quite common attacks, simple to execute and inconspicuous, the victim is easily deceived.

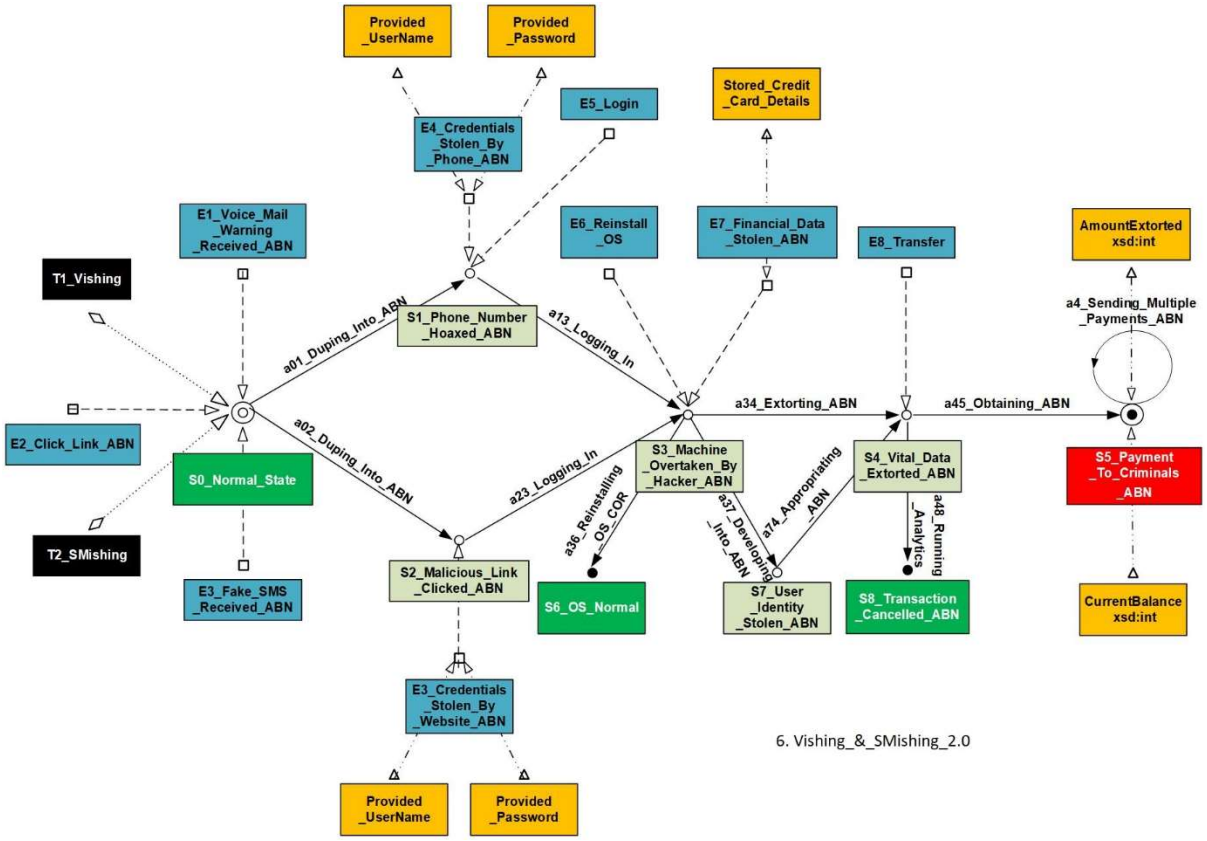

FIGURE 7. VISHING & SMISHING.

7. Withdrawal with Session Hijacking starts with card inserted and ends with card removed. In this case credentials are stolen by the means of Session Hijacking. This attack is more sophisticated and uses stolen identity to execute payments to criminals.

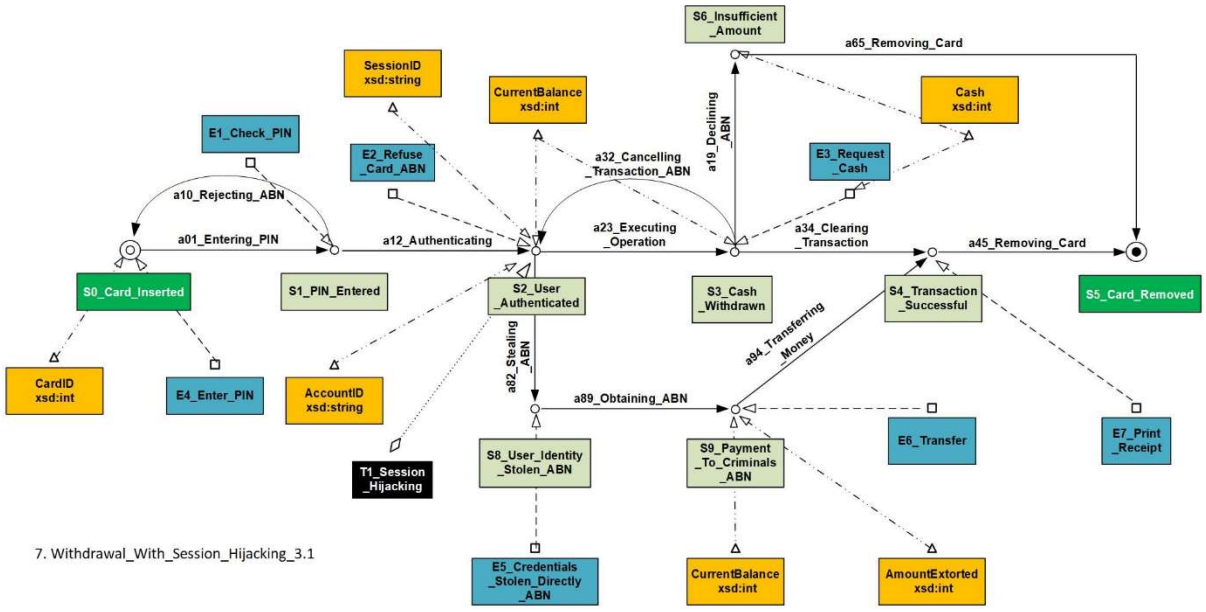

FIGURE 8. WITHDRAWAL WITH SESSION HIJACKING.

8. Sending Email Spam deals with hacker sending hoaxed emails from overtaken PC. The goal of sending spam is distributing malware which can be used in different types of intrusions. This is an entry point for many attack vectors and a common way to gain control over random PCs.

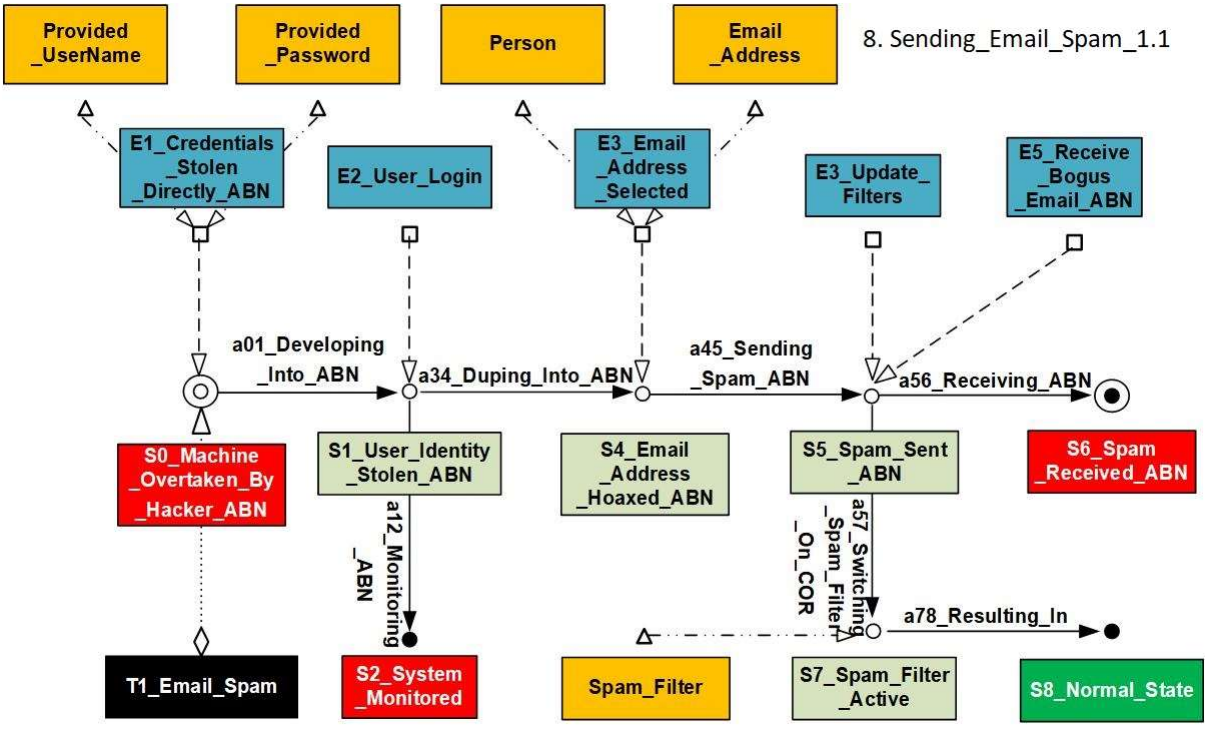

FIGURE 9. SENDING EMAIL SPAM.

9. Email Spam Received complements the previous scenario closing the loop of sending spam, overtaking another PC, and sending spam again. The process repeats itself infecting even more machines.

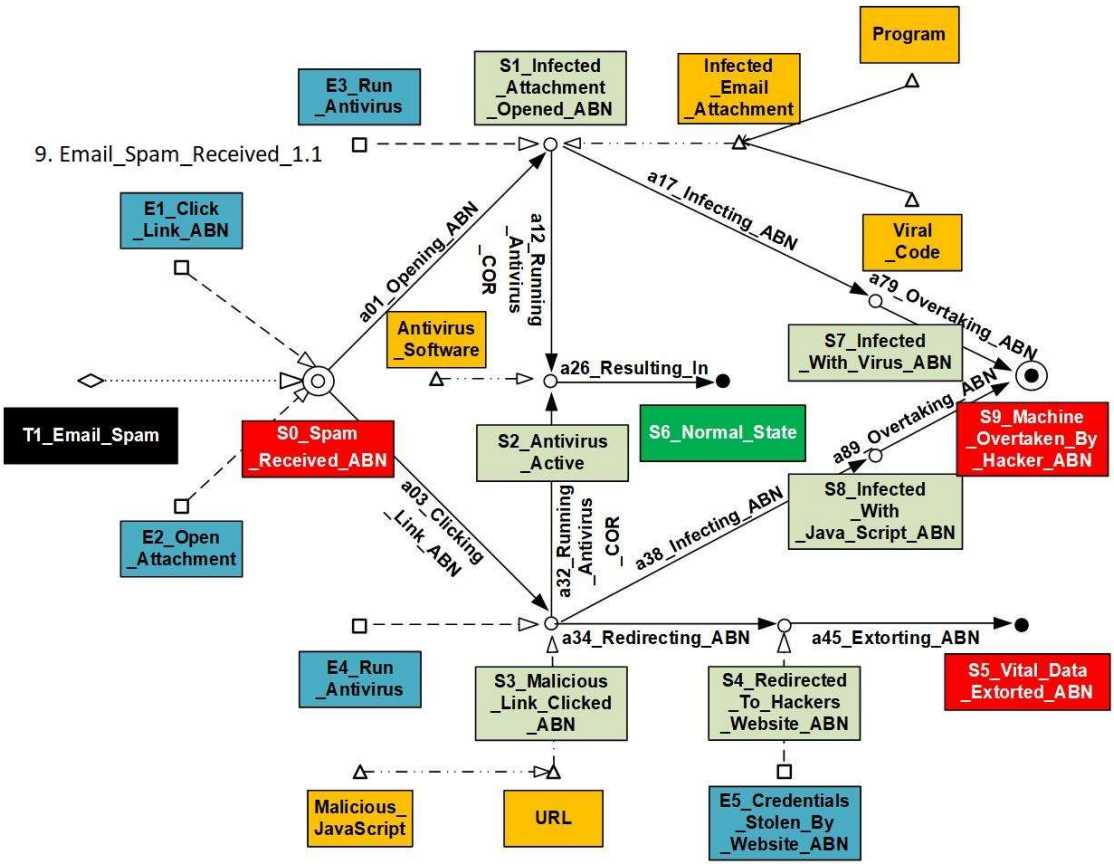

FIGURE 10. EMAIL SPAM RECEIVED.

10. Cross Channel with Pretexting deals with redirecting to hacker website by a phone call from hacker pretending to be IT support. The result is either money sent to criminals or transaction cancelled by running analytics. Pretexting is also a common type of attack, in this scenario happening over two channels, mobile phone and website. This case demonstrates a cross-channel threat.

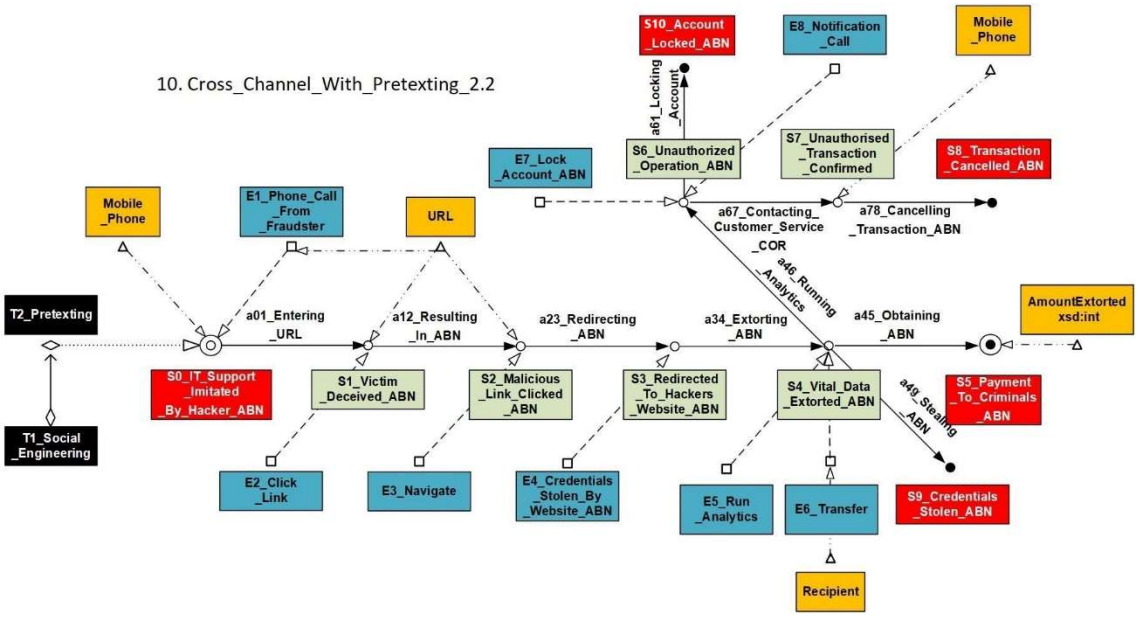

FIGURE 11. CROSS CHANNEL WITH PRETEXTING.

11. Scareware & Rogue are two similar threats that deceive the user leading to believe that their PC is infected already. Rogue requests ransom whilst scareware offers a fake antivirus program for removing a non-existent threat. From the user perspective these threats look like legitimate tools for removing malware from their PC.

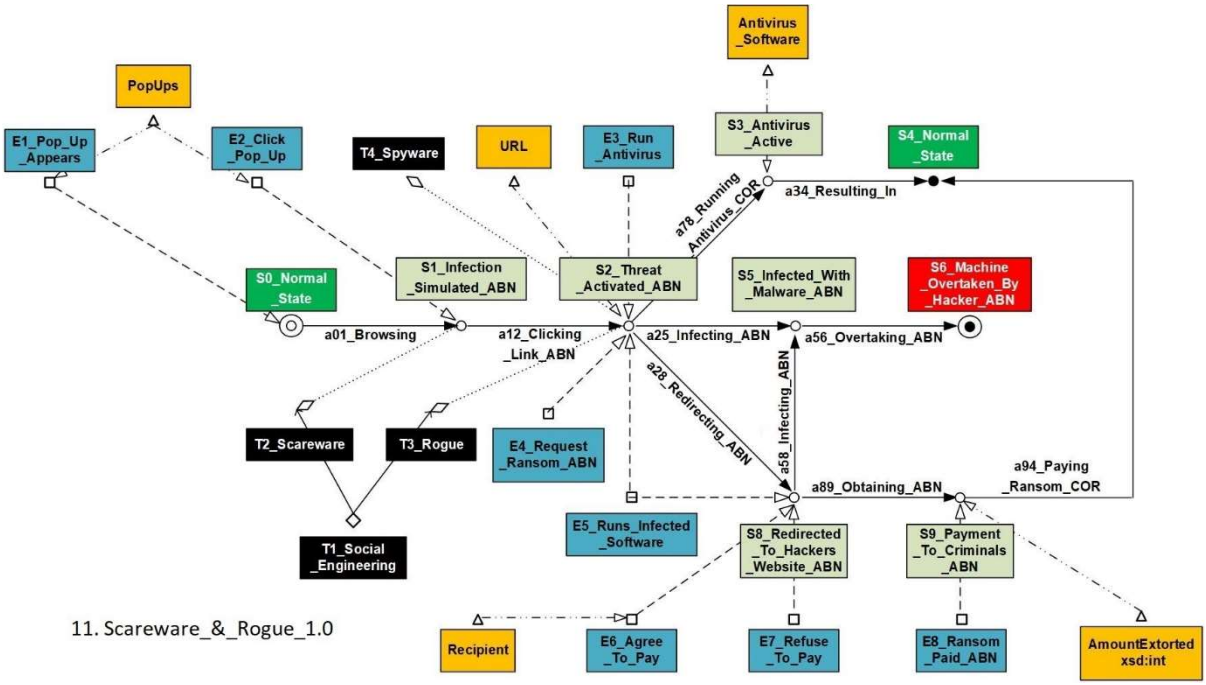

FIGURE 12. SCAREWARE & ROGUE.

12. ATM Infected is a real-life scenario in which the ATM operating system is infected and overtaken by hacker, allowing to swallow the card, steal the card details and extort the money. This scenario is based on real events.

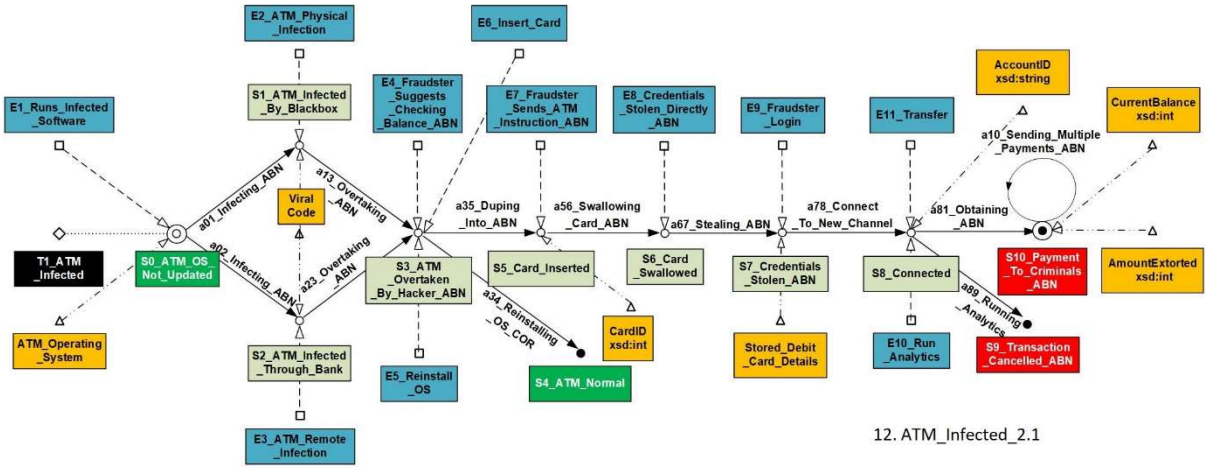

FIGURE 13. ATM INFECTED.

4.2. Directed graphs.

For representing the scenarios, we used directed graphs which are sets of objects called vertices or nodes that are connected by edges. These objects are Situations in our graphs. Edges are directed from one vertex to another and effectively are ordered pairs of vertices (Actions in our framework). Directed graphs are sometimes called digraphs or directed networks. Our graphs have stereotyped nodes and edges and are related to state chart diagrams - one of the five types of UML diagrams used to model the dynamic nature of a system. In these, various states of an object during its lifetime are defined and changed by events. They are useful for modelling reactive systems (responding to external or internal events). Graphs describe the flow of control from one state to another. States are defined as a condition of an existing object which changes when an event is triggered. In our terminology Action is the trigger for changing the system from one state (Situation) to another.

As can be seen in the diagrams, the vulnerabilities are identified in situations where the threats appear first. The potential deadlocks (situations without exit) are shown also. In the scenarios the corrective actions mitigating the threats lead to normal situations allowing the flow of the scenario to be resumed. The decision which path (arch) to take from the situation (node) is made based on the applicable rule or occurrence of an event. Events are asynchronous and can appear in the scenario at any point. The vulnerabilities or gaps in the security are entry points for threats and identifying them early allows to mitigate threats easily. Deadlocks can be caused by inconsistencies in the model or rules and identifying them helps to keep the infrastructure functioning properly. The scenarios can be used for SWRL rule extraction as they describe a part of the system, making them simpler to process.

4.3. Implementation.

The environment is based on 5 docker containers - Airflow to create model and process tasks, PostgreSQL for storing metadata, Kafka for data streaming, Zookeeper for managing sessions and topics and MLflow for showing models' statistics, comparison, and incremental learning. Docker Compose is a tool for defining and running multi-container Docker applications. The variables passed by AirFlow were created from the threat scenario ontologies using parts of original Sycamore project code. The Threat variable states the type of attack vector. Filename specifies the PCAP file used for initial model training. The three packet_types determine the TShark filters on packet and types of packets analysed. Detection is done using filters applied to packets, based on patterns of three elements (suspicious packets) which if correlated, signify a threat. For instance, in “Transfer With dDOS” scenario, suspicious packets are:

"packet_type_1": "GET.*HTTP.*1.1",

"packet_type_2": "POST.*HTTP.*1.1",

"packet_type_3": "HTTP\\/1.1\\s200\\sOK",

We are working on the implementation of other scenarios to be executed through ML algorithms and orchestrated by Airflow dashboard.

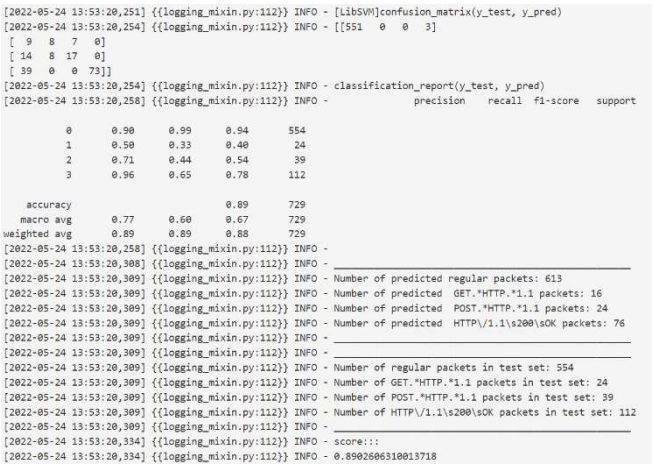

FIGURE 14. AIRFLOW DASHBOARD LOG FOR TRANSFER WITH DDOS SCENARIO

We will now discuss the workflow of Directed Acyclic Graphs inside the containers (Oak software). The Initial Model DAG loads and pre-processes the PCAP data converting it into CSV and parsing into a Pandas dataframe. The data is divided into training and testing sets, 10000 samples are set aside for streaming simulation. The model is constructed and fitted with class weights (as the dataset is imbalanced, there is a need for a mechanism to emphasise underrepresented labels). The Neural Network / SVM classifier is stored in current_model folder. The Updating DAG can be executed now. Normally it runs in 5 minutes intervals streaming data from Kafka producer through broker to the consumer. Broker acts as an intermediate between producer and consumer using a topic which is a hosting stream. Samples are converted into JSON, encoded, and pushed to the topic. From there they are retrieved, decoded, and converted back to NumPy array stored in a file in to_use_for_training folder. The current model is loaded, and its score is evaluated to adjust class weights. Scores between original and adjusted model are compared. If adjusted model outperforms the original one, it replaces the old one which is archived. The statistics (metrics) are sent to MLFlow container and shown in the dashboard. Alternatively, instead of streaming the data from the server the streaming process is simulated using samples set aside by the Initial DAG.

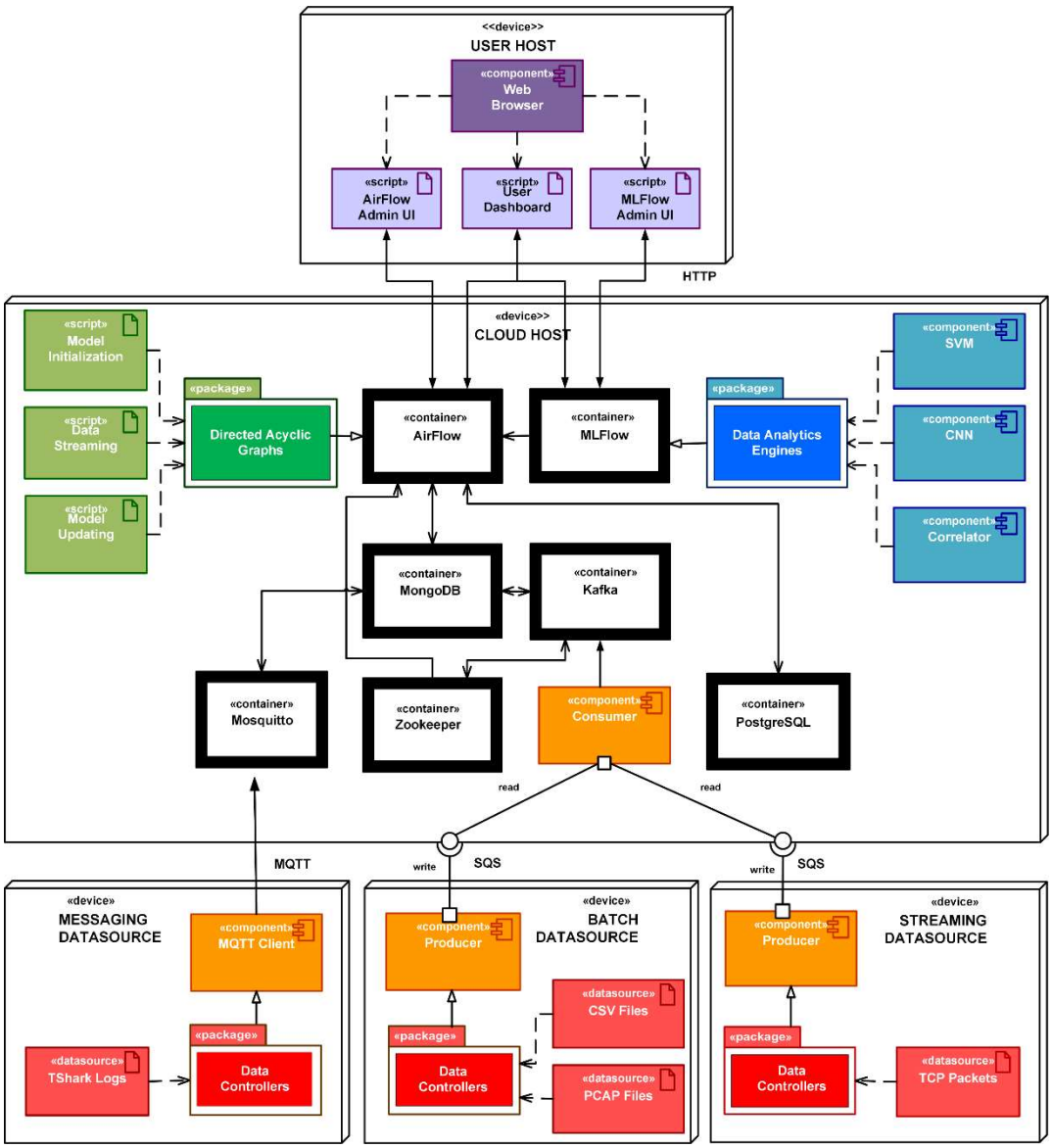

FIGURE 15. IMPLEMENTATION COMPONENTS.

5. Summary.

First, we formulated some explicit requirements for the scenarios. Completeness means that the scenario set was consulted with Lloyds Bank. Maximal determinism entails that the scenarios always continue in the lack of threats (if there is no threat present the journey persists until committing transaction). Minimal non-determinism requires that there is no more than one choice of path in the case of present threats. Isolation signifies that there are no overlapping paths in the scenario graphs. Consistency implies that the journey always leads to transaction commit.

The asynchronous events in the scenarios trigger the execution of actions. The actions themselves generate events which may trigger further actions. Therefore, we verified the event capturing mechanism of the framework. It is based on data analysis for detection of potential intrusions, identifying known threats or classifying unknown threats. Validation of the framework is based on the use of the intelligence graphs to demonstrate the capability of the framework to deal with typical scenarios by detecting the threats through ML, identifying them by checking the situations in which they appear in the ontology of threats, selecting appropriate counteraction using the security rules and planning their execution as containerized services. The validation of the hybrid framework is achieved through running 12 simulation scenarios describing the flow of control from one state to another.

6. Future work.

New algorithms stemming from this work comprise of rule indexing and index-based inference. Rule indexing produces contextual information about the ontological classes that can be used for automatic generation of the security policies and control over the simulation. The semantic context of situations, events and actions is found using the indexing scheme. For indexing, automatic rule generation and simulating the bank “journey” across the scenarios Java programming language is used in conjunction with Jena API. There are multiple suggestions for future conceptual development within the same framework. Automatic

generation of the intelligence graphs from OWL+SWRL is possible. Automated rule generation can be explored. Rules can be extracted from ontology axioms using starting situation, its declared properties, and ranges of properties. Generating sequential plans and execution workflows is worth examining also.

References:

1. Vassilev, V., Sowinski-Mydlarz, V., Gasiorowski, P. et al., "Intelligence Graphs for Threat Intelligence and Security Policy Validation of Cyber Systems". In: P. Bansal, M. Tushir, V. Balas and R. Srivastava (eds.), Advances in Intelligent Systems and Computing, Vol. 1164, pp. 125-140. Springer (2020); ISSN 978-981-15-4991-5.

2. Czagan D. (2014) “Qualitative Risk Analysis with the DREAD Model”. Infosec [online] Available at: https://resources.infosecinstitute.com/topic/qualitative-risk-analysis-dread-model/ [Accessed 27 Mar. 2022].

3. RashiGarg (2018) “Computer Network | Threat Modelling”. GeeksforGeeks [online] Available at: https://www.geeksforgeeks.org/computer-network-threat-modelling/ [Accessed 27 Mar. 2022].

4. Ehrlinger, L. and Wöß, W. (2016) “Towards a Definition of Knowledge Graphs”. Institute for Application Oriented Knowledge Processing, Johannes Kepler University. [online] Available at: https://jena.apache.org/ [Accessed 27 Mar. 2022].

5. Max Planck Institute for Informatics (2019) “YAGO: A High-Quality Knowledge Base”. Max Planck Institute for Informatics. [online] Available at: https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/ [Accessed 27 Mar. 2022].

6. Chowdhury, S. (2018) “Knowledge Graph: The Perfect Complement to Machine Learning”. Towards Data Science. [online] Available at: https://towardsdatascience.com/knowledge-graph-bb78055a7884 [Accessed 7 Nov. 2019].