AN OVERVIEW OF MODELING CYBERATTACKS WITH PETRI NETS

Katia P. Mayfield, John A. Blonsky, Tymaine S. Whitaker, and Mikal D. Petty

1Athens State University
300 N. Beaty Street, Waters Hall S103C, Athens AL 35611 USA
Katia.Mayfield@athens.edu

2University of Alabama in Huntsville,
301 Sparkman Drive, OKT N353, Huntsville AL 35899 USA
pettym@uah.edu, tsw0003@uah.edu, bjohn316@gmail.com

Keywords: Cybersecurity, Modeling, Petri nets

ABSTRACT: The University of Alabama in Huntsville is conducting four interrelated research projects that together form an integrated research program in cyberattack modeling. Cyber security is a large concern for many private and public organizations. Cyberattacks are very different from not long ago when the computer risks that people needed to be concerned with were those caused by programmers and human error. Different types of vulnerabilities, resulting of the application of new technology, have exposed many systems to malicious intentions, ensuing what we now know as cyberattacks. Most cyberattacks follow patterns that have already been identified and documented. One of the repositories of such documentation is MITRE Corp. through their Common Attack Pattern Enumeration and Classification (CAPEC) reports. Such reports include information such as attacks pre-requisites, steps followed by the attacker, expected outcomes, and security controls that may be in place. Knowledge of cyberattacks allows for developers to be aware of vulnerabilities that attackers may try to exploit. Being able to model attacks based on the attacker’s perspective provides system developers with information required to develop security aware systems. A common method to represent process patterns is through models that can be simulated by computers, providing basic information on how such patterns behave and may evolve, allowing the prediction of future events. A typical example of such approach is weather forecast based on knowledge of current data patterns. One of the ways to model cyberattacks is based on the utilization of Petri Nets, which are known to be used for mathematical and graphical modeling. In this study, CAPEC reports are used as a source of information to create a Petri net, modeling one of the top ten attacks reported by Open Web Application Security Project (OWASP), specifically SQL Injections. A complete description of the translation of the CAPEC report in a Petri net is given, with a detailed explanation of how each Petri net component was chosen and linked to other components, follows a brief introduction to the CAPEC reports, SQL injection and Petri nets.

1. INTRODUCTION

Since the late 90’, cyberattacks and how advances in technology can be used against us have been a main focus of concern, ranging from individuals all the way to government agencies. A significant example was reported on June 23, 2017, by Samantha Ehlinger in an article for the San Antonio Express-News, showing the FBI conclusion that Texans were the most vulnerable and largest victim of cyberattacks in 2016 [1]. Additionally, recent ransomware attacks have had a global impact on computer dependent organizations. In particular, in May 2017, a ransomware cryptoworm, targeting computers running Microsoft Windows operating systems, encrypting data and requiring ransom payment, was named the WannaCry ransomware cryptoworm. In June 27, 2017 Sheera Frenkel wrote an article in The New York Times titled “Global Ransomware Attack: What We Know and Don’t Know.” Frenkel showed an image of what the attack message displayed and listed some of the global private companies that confirmed being impacted amplifying the panic effect in the general population, unaware of such risks [2]. The unknown may leave people to wonder what could possibly happen next as nobody knew who was behind the attack, even considering that there were some suspicions, and whether or not there will be additional damage to the systems that were infected with the ransomware. Such examples are just a sample of the basis to justify studies that could allow an improvement on cyber defenses.

A simple observation/solution is to create systems that are secure enough to not be vulnerable to a cyberattack and/or know to react to an attack in progress. However, for system designers to be able
to create secure products there must be knowledge of how the attack actually works. This study describes an initial effort on how to represent the different paths an attacker may follow and use that information to respond to the attack.

There have been several studies conducted on cyber-attacks and being able to model such attacks. Pan et. al. in “Causal Event Graphs Cyber-physical System Intrusion Detection System,” describe a methodology to model defense solutions for electric transmission system intrusions. In their study they model cyber physical systems behaviors using causal event graphs [3]. Yeole et. al. in “Analysis of Different Technique for Detection of SQL Injection” discuss mechanisms and techniques that are available to detect SQL injections. One of the main problems stated in their study is that the techniques they describe take the input and compare it to saved patterns of SQL injections, so if an attacker finds a different way to perform the attack their proposed solution will not be helpful [4]. Navdeep Kaur and Parminder Kaur in “Mitigation of SQL Injection Attacks using Threat Modeling,” attempt to provide a way of integrating attack mitigation into the early stages of the system development life cycle. They use threat modeling to perform a mitigation for an SQL Injection attack. Their model uses data flow architecture and deployment diagrams as also other unified modeling language concepts to demonstrate the stages of where, within the system development lifecycle, the SQL Injection attack would occur [5].

Considering all the different attacks that now exist, attack patterns have been researched and labeled as a knowledge resource instrumental on the design of secure software [6]. Most cyberattacks follow patterns that have already been identified and documented. A common process to represent any pattern is through models that can be simulated by computers, providing basic information on the evolution of the target pattern.

Petri nets have been used to model different systems; in particular, it has been applied to cyberattacks. Khan et. al. in “Cyber Resilience-by-Construction: Modeling, Measuring & Verifying,” define an Algebraic Petri net that is used as the basis to propose a formal framework to measure cyber resilience from attacks and failures [7]. Anita Zakrzewska and Erik Ferragut in “Modeling Cyber Conflicts Using an Extended Petri net Formalism,” have extended the design of a Petri net to support game-theoretic analysis of cyberattacks [8].

In this study cyberattacks are modeled through the use of Petri nets. The next sections present a brief introduction to Petri nets and some known cyberattacks. It is followed by a proposed definition of a formal representation of the Petri nets. Section 4 provides an initial walkthrough example followed by section 5 which describes how the initial Petri net design has been enhanced into the design that is being used in this research. Lastly there is a summary of the topics presented.

2. BACKGROUND

With attack patterns being recognized as a resource to design secure software, one should consider what process they will use to recreate such pattern [6]. In this study, attack patterns are recreated based on the Common Attack Pattern Enumeration and Classification (CAPEC) report published by MITRE Corp. and are modeled through the use of Petri nets [9].

Petri nets are models which are graphically and mathematically developed (compatible) [10]. They were first introduced in 1962 and have been used to model different applications such as parallel activities, dataflow computation, communication protocols, synchronization control, producer-consumer systems with priority and formal languages to name a few. Petri nets are graphs with nodes represented by circles and rectangles (solid or hollow), and directed edges as shown in Figure 1.

![Figure 1 A simple Petri net – Basic components](image)

The circles represent a “place”, which corresponds to a condition being true or false. As one can see from the partial design in Figure 1, a Petri net is a directed graph. For the purpose of establishing the modeling process in this study, the initial places that are found at the top of the graph are assigned one token, represented by the dots found within the places. Tokens are used to represent that either it is the resource required for a place to be true, or that the
Table 1 Some typical interpretation of Transitions and Places according to Murata

<table>
<thead>
<tr>
<th>Input Places</th>
<th>Transitions</th>
<th>Output Places</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>Event</td>
<td>Postconditions</td>
</tr>
<tr>
<td>Input data</td>
<td>Computational step</td>
<td>Output data</td>
</tr>
<tr>
<td>Input signals</td>
<td>Signal processor</td>
<td>Output signal</td>
</tr>
<tr>
<td>Resources needed</td>
<td>Task or job</td>
<td>Resources released</td>
</tr>
<tr>
<td>Conditions</td>
<td>Clause in logic</td>
<td>Conclusion(s)</td>
</tr>
<tr>
<td>Buffers</td>
<td>Processor</td>
<td>Buffers</td>
</tr>
</tbody>
</table>

place holds truth. The edges represent the transitions between the places. When modeling with Petri nets, emphasis should be placed on the rules associated to transition enabling and firing.

Table 1, proposed by Tadao Murata in “Petri nets: Properties, Analysis and Applications,” displays some of the basic interpretations of places, including transitions between them [10]. If a place does not have an incoming arc then that place is an initial node. Initial nodes imply where truth is established and therefore they have a token present. If there are no outgoing arcs from a place then the place is considered as an “end condition.”

More specific details of the design of Petri nets can be found in the companion paper by M. Petty, et. al. titled “Modeling Cyberattacks with Petri Nets: Research Program Overview and Status Report” [11].

3. WALKTHROUGH OF INITIAL MODEL DESIGN

To be able to determine which cyberattacks to focus on for this study, the research group utilized the Open Web Application Security Project (OWASP) and referenced Anita Zakrzewska and Erik Ferragut’s research, “Modeling Cyber conflicts Using an Extended Petri net Formalism” [8]. OWASP has the top ten common system vulnerabilities and SQL Injections has made the top ten for multiple years in a row [12]. Zakrewska and Ferragut conducted research where they generalized the graphical modeling of a Petri net to include player controlled components, this is outside of the scope of this paper, however, it will be addressed in future work.

To be able to properly model the SQL Injections, this study bases the Petri net model on the CAPEC-66 SQL Injection report [9]. CAPEC-66 contains a complete schema and classification taxonomy of the attack pattern. By following the CAPEC-66 report, a Petri net is developed that will allow software developers to fully understand the attackers perspective and the approaches use to exploit software systems, allowing such developers to enhance security during the system development lifecycle.

Through the process of dynamic validation, Nicholas Christiansen showed, in his thesis work, that the designed Petri net based on CAPEC-66, demonstrated in section 4, is a valid design which can be followed from the attackers perspective to be able to exploit a system, Mr. Christiansen’s test case was completed to exploit a Metasploitable system [13].

For the purpose of creating a model basis, there are three areas of the CAPEC report in which this study focuses on. First is the summary of the attack, second is the attack execution flow, and third is the attack prerequisites.

The summary of the report is used so that the model designer is able to fully understand the attackers’ perspective, how the attack works and the actions that must be taken to have a successful attack. For the purpose of modeling, the next item that is reviewed from the report is the attack prerequisites. Based on the required prerequisites to the attack, the designer of the model is able to determine the initial Petri net places. Figure 2(a) shows the attack prerequisites shown in the CAPEC-66 report, (b) shows the representation of the prerequisites as the initial places of the Petri net.

Once the initial places based on the prerequisites of the attacks have been modeled, the next step is to follow the attack execution flow to continue modeling the attack. Many of the CAPEC reports follow a structure where the first part of the attack execution flow starts with an explanation of the attacker’s exploratory phase. The action that is taken to complete the exploration, along with following the stated attack step techniques, provides the model designer with the information required to determine transitions.
The exploratory section also includes information on the possible outcomes based on the attack step techniques that are given. The outcomes documented in the CAPEC report display the places in which a transition can “fire” to.

Figure 3(a) shows the details for the SQL Injection, while (b) graphically represents the same information. At the top of Figure 3(b) is the transition in which the initial Petri net places lead to, therefore, making that transition the first action taken by the attacker.

Based on the main step of the explore phase of the CAPEC report, the action that is taken by the attacker is represented by the transition found at the top of the graph. There is one outcome based on that transition and it is that the attacker has reached a point where he has taken inventory of the functionality of the system that has been exposed. From this place, the attacker is able to take one of two actions, these are...
found under the attack step techniques. With whatever technique the attacker decides to attempt, the outcome will be the same, which is obtaining the results of the exploratory phase. There are two possibilities that can then occur from the exploration that the attacker has completed, one is of success which will then allow for the attack to continue, and the other is that of failure, represented by an end place of “No inputs to applications identified.”

In the CAPEC-66 report there is only one action to be taken to complete the exploratory phase. However, there is no guarantee that all CAPEC reports will only have one step for the exploration. The model must assume that it could have multiple actions, in which case the designer of the Petri net will have to determine if only one exploration phase is completed or if multiple could occur. This is important because the attack may occur either sequentially, only requiring one of the exploratory steps to be successful, or through multiple steps that need to succeed for the full attack to be successful.

In the scenario where the attacker has been able to identify at least one data input to the application, then the model continues where now the attacker enters the experimental portion of the attack. Just like with the explore phase, the experiment phase can have multiple steps to it. With the SQL Injection attack there are two experiment phases. These phases are sequential and will only occur when at least one of the techniques that is used in the previous step is successful (unless noted otherwise in the CAPEC report). Within the experiment phase information provided in the CAPEC report, there is not only the attack step techniques and outcomes as was already shown with the exploratory phase, but there may also be information on negative and positive indicators which are associated to the attackers success or lack thereof in the outcomes he/she is trying to achieve. There may also be security controls where detection and prevention could take place. For the purpose of this study the focus on creating the Petri nets is based on the attack step techniques, indicators and the outcomes of those techniques. Additional information from the CAPEC report can be used to later expand the Petri net and this is something that the authors of this study are working on, an example is mitigation. Figure 4 (a) displays the first step of the experimental phase with listed out attack step techniques, indicators, outcomes and security controls. Figure 4 (b) represents this information as part of the Petri net. As mentioned for the Petri net model, the focus is on the attack step technique representing the appropriate outcomes. As it is shown, there are four attack step techniques, each of these techniques will provide the attacker with a positive or negative indication of their attack. Therefore, all four action steps for the attack lead to a place where the indication is known as to whether there will be a positive or negative indication to the success of the attack. From the indicator place there are three actions that can be taken based on the two negative and the one positive indicator. The positive indicator will lead to a place where the attacker is aware that the SQL injection attack can continue, the two negative indicators would then lead the attacker to a place where the attack can no longer continue.

Once the attacker has reached the point where he/she is aware that a given input is vulnerable to SQL Injections then the second experimental step can take place which is to try to exploit the vulnerability. This step also has four attack step techniques, therefore there will be four transitions associated with each possible action of the attack. This section of the report also has indicators, however the indicators here differ a little from that of step one. In this step the indicators are only two, one positive and one negative which directly corresponds with the outcome of success or failure, therefore there is not an additional place to represent the knowledge of the indicators. The diagram flows directly to the actions that leads to success or failure of the attack.

After following the CAPEC-66 report, and creating the Petri net model from what is described in the report, the attacker will either end up being successful in their attempt or the attempt will fail.

After modeling the Petri net, following the verbatim process described in CAPEC 66, it was decided that further exploration of how the Petri net should be designed through multiple iterations based on different information found in the CAPEC 66 report was required. This led to the enhanced design of the initial Petri net which is described in Section 4.
1. Determine user-controllable input susceptible to injection:
Determine the user-controllable input susceptible to injection. For each user-controllable input that the attacker suspects is vulnerable to SQL injection, attempt to inject characters that have special meaning in SQL (such as a single quote character, a double quote character, two hyphens, a parenthesis, etc.). The goal is to create a SQL query with an invalid syntax.

**Attack Step Techniques**

<table>
<thead>
<tr>
<th>ID</th>
<th>Attack Step Technique Description</th>
<th>Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Use web browser to inject input through text fields or through HTTP GET parameters.</td>
<td>env-Web</td>
</tr>
<tr>
<td>2</td>
<td>Use a web application debugging tool such as Tamper Data, TamperIE, WebScarab, etc. to modify HTTP POST parameters, hidden fields, non-freeform fields, etc.</td>
<td>env-Web</td>
</tr>
<tr>
<td>3</td>
<td>Use network-level packet injection tools such as netcat to inject input.</td>
<td>env-web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
<tr>
<td>4</td>
<td>Use modified client (modified by reverse engineering) to inject input.</td>
<td>env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
</tbody>
</table>

**Indicators**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Indicator Description</th>
<th>Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Negative</td>
<td>Attacker receives normal response from server.</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
<tr>
<td>2</td>
<td>Positive</td>
<td>Attacker receives an error message from server indicating that there was a problem with the SQL query.</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
<tr>
<td>3</td>
<td>Negative</td>
<td>Server sends a specific error message that indicates programmatic parsing of the input data (e.g., NumberFormatException).</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
</tbody>
</table>

**Outcomes**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Outcome Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>At least one user-controllable input susceptible to injection found.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>No user-controllable input susceptible to injection found.</td>
</tr>
</tbody>
</table>

**Security Controls**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Security Control Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Preventative</td>
<td>Search for and alert on unexpected SQL keywords in application log (e.g., SELECT, DROP, etc.).</td>
</tr>
<tr>
<td>2</td>
<td>Preventative</td>
<td>Input validation of user-controlled data before including it in a SQL query.</td>
</tr>
<tr>
<td>2</td>
<td>Preventative</td>
<td>Use parameterized queries (e.g., PreparedStatement in Java, and Command.ParameterizeArgumentException to set query parameters in .NET).</td>
</tr>
</tbody>
</table>

Figure 4 (a) CAPEC Experiment Step 1  (b) Petri net showing the unsuccessful attack end place
2. Experiment and try to exploit SQL Injection vulnerability:

After determining that a given input is vulnerable to SQL Injection, hypothesize what the underlying query looks like, iteratively try to add logic to the query to extract information from the database, or to modify or delete information in the database.

**Attack Step Techniques**

<table>
<thead>
<tr>
<th>ID</th>
<th>Attack Technique Description</th>
<th>Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Use public resources such as &quot;SQL Injection Cheat Sheet&quot; at <a href="http://fernun.mavituna.com/makale/sql-injection-cheatsheet">http://fernun.mavituna.com/makale/sql-injection-cheatsheet</a>, and try different approaches for adding logic to SQL queries.</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
<tr>
<td>2</td>
<td>Add logic to query, and use detailed error messages from the server to debug the query. For example, if adding a single quote to a query causes an error message, try: &quot;’ OR 1=1’;--&quot;, or something else that would syntactically complete a hypothesized query. Iteratively refine the query.</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
<tr>
<td>3</td>
<td>Use 'Blind SQL Injection' techniques to extract information about the database schema.</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
<tr>
<td>4</td>
<td>If a denial of service attack is the goal, try stacking queries. This does not work on all platforms (most notably, it does not work on Oracle or MySQL). Examples of inputs to try include: &quot;; DROP TABLE SYSDICTION; ;&quot; and &quot;; DROP TABLE SYSDICTION; ;&quot;. These particular queries will likely not work because the SYSDICTION table is generally protected.</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
</tbody>
</table>

**Indicators**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Indicator Description</th>
<th>Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Positive</td>
<td>Success outcome in previous step.</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
<tr>
<td>2</td>
<td>Negative</td>
<td>Failure outcome in previous step.</td>
<td>env-Web env-ClientServer env-Peer2Peer env-CommProtocol</td>
</tr>
</tbody>
</table>

**Outcomes**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Outcome Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>Attacker achieves goal of unauthorized system access, denial of service, etc.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>Attacker unable to exploit SQL Injection vulnerability.</td>
</tr>
</tbody>
</table>

Figure 5 (a) Attack step techniques (b) Petri net representing the step attack techniques
4. ENHANCED CAPEC 66 PETRI NET DESIGN

One of the main objectives of using CAPEC as a resource to designing cyberattack models is due to the fact that CAPEC provides knowledge based on the attackers perspective. This is provided so that way designers and developers are able to understand the minimal skill sets and knowledge that an attacker must have. By having this knowledge, designers and developers can then focus on the portions of the CAPEC report that describe mitigation techniques.

In the enhanced Petri net design, the focus shifted in the content of the CAPEC report that was used to develop the model. In the initial design the focus was on the attack itself, here the focus is on the goals that the attacker has. This shift occurred because the design should not be specific to techniques, software, and hardware that are used for an attack. With the way the world is changing everyday along with technology, these techniques, software and hardware are one day going to be obsolete and the models that are created should be able to still be used without concern of having to require the specifications of new technology within the model. The enhanced model is also designed to be categorized as a Petri net with Players and Strategies (PNPS) which was introduced in the companion paper by M. Petty, et. al.

The design decisions that were made for the enhanced Petri net are as follows:

1. The transitions will now represent four different possible type of actions
   a. Non-controllable actions
   b. Attacker’s actions
   c. Active defender actions
   d. Passive defender actions

The one change with respect to the Petri net presented in section 3 that is found within this initial model of PNPS are the two different defender actions. The active defender that is modeled in the Petri net represents a person that is actually monitoring the system. The passive defender that is modeled is representative of areas where an Intrusion Detection System (IDS) may be found and it may block an attack from occurring on the system. Figure 6 shows a portion of the enhanced Petri net where the attacker’s actions are modeled as red transitions and the active defender’s actions are modeled in blue, while the non-controllable actions are found unchanged from their standard design.

Figure 6 Player controlled transitions defined

Another modification that was introduced in the Section 3 model implies that the model no longer represents the specifics of how the attack might occur but the focus has been changed to modeling the goals that the attacker is possibly trying to achieve. No matter which goal the attacker is trying to achieve there will be standard transitions that are a result from it.

1. The attacker’s goal can succeed or fail
2. The attacker’s goal can be blocked by a defender, either a passive or active defender.
3. The active defender can gain knowledge of the attack and actively monitor the system.

Figure 7 continues from the place “Controllable Inputs Found” from figure 6 and displays the Petri net with four goals that the attacker has according to CAPEC 66. The goals associated to SQL injections are (1) read application data, (2) modify application data, (3) gain privileges, or (4) execute malicious code. For each of these represented goals, the above mentioned transitions are found.
Figure 7 shows the addition of the attacker controllable transitions which are outlined in red, the active defender controllable transitions in blue, and the passive defender controllable transitions in green. It can be seen that each of the goals is following the same transition design, within each of these if the attacker fails at achieving the goal there are three outcomes from not being successful.

1. The attacker has been blocked and therefore has no opportunity to attempt the attack again.

2. The attacker has been detected by the defender however has not been blocked therefore allowing for the attack to continue.

3. The attacker has failed at achieving the goal without defender interaction and therefore the attacker has the opportunity to attempt the goal again or try to achieve one of the other goals associated to the attack.

Figure 8 takes a closer look at the transitions and places associated to each of the goals that the attacker can achieve.
With this enhanced Petri net design there is no need to stay focused on the detailed manner in which the attack takes place. The initial design of the Petri net is assumed to allow the possibility of having to be updated any time that a new technology was developed that could be used by the attacker. With the enhanced model the concern is just based on the success or failure of the attacker’s actions. By focusing on the success or failure of an attacker and not on the specific actions, the enhanced Petri net can also be validated, which is discussed in the companion paper by J.A. Bland, et al [14].

5. SUMMARY

Amid the architecture of a software security knowledge base and techniques, attack patterns play a unique role. By being able to model these patterns there is value added to the identification of security requirements, architectural risks, possible risk based and penetration testing and can go as far as allowing organizations to realize any vulnerabilities within their policies and standards [9].

This study provided the initial foundation to the design of Petri net models following a CAPEC report. A simple example showing an SQL injection attack modeling demonstrated the possibility of someone with knowledge of places already fired to predict the next action by the attacker. This initial simple example was then modified to take into consideration how someone could mitigate and prevent attacks by representing attackers and defenders within the Petri net. Additional work is still being conducted to add in probabilities (rates) for transitions to fire, automate the design of the Petri nets, and implement reinforcement learning.

6. REFERENCES


7. Authors’ Biographies

KATIA P. MAYFIELD is currently an Assistant Professor of Computer Science at Athens State University. She is currently a Ph.D. student in Computer Science at the University of Alabama in Huntsville. She previously completed coursework towards a Ph.D. in Scientific Computing at University of North Dakota and she received an M.S. degree in Computer Science from Midwestern State University. She is a member of Upsilon Pi Epsilon (UPE), the honor society in Computing Sciences. Prior to teaching at Athens State University, she worked for two years as a Computer Science instructor at Cameron University and three years as a Graduate Teaching Assistant at the University of North Dakota. Her research interests include computer security, digital humanities, software engineering, and parallel processing. She has co-authored twelve published papers and previously worked in a project sponsored by the Texas Higher Education Coordinating Board.

JOHN A. BLAND is currently employed as a computer engineer and cybersecurity researcher at an army research laboratory. He received a B.S.E. degree in computer engineering and a masters in software engineering from the University of Alabama in Huntsville. He is currently working towards a Ph.D. in computer engineering with a focus on modeling & simulation and cybersecurity at the University of Alabama in Huntsville. Prior to working in the field of cybersecurity, he worked in multiple hardware-in-the-loop facilities and supported multiple live test events. His experience includes developing high fidelity simulations, embedded system specification, test planning, and system analysis. His research interest include applied cybersecurity, machine learning, and application of simulations.

TYMAINE S. WHITAKER is currently a Senior Software Engineer at Parsons and a Computer Information Systems adjunct instructor at Calhoun Community College. He received a B.A. degree in Computer Science from Talladega College and the M.S. in Computer Science from Alabama A&M University. His Master’s thesis, “Evaluation of a Face Recognition Software”, evaluated security and biometric components of a face recognition software. He is a member of Alpha Phi Alpha Fraternity, Inc. which develops leaders, services communities and promotes academic excellence. As an intern, Mr. Whitaker participated in the Undergraduate Student Research Program (USRP) where he performed modeling and simulation work in Advanced Concepts for Marshall Space Flight Center. After completing his undergraduate studies, Mr. Whitaker accepted a software engineer position at Science Application International Corporation (SAIC) to continue his modeling and simulation work at NASA. After working with NASA, Mr. Whitaker provided modeling, simulation and training for the U.S. Army Aviation and Missile Research, Development, and Engineering Center (AMRDEC). Mr. Whitaker is a Ph.D. student in Modeling and Simulation at the University of Alabama in Huntsville. His dissertation entitled “Generating Cyberattack Component Models from Vulnerability Databases” describes a new method for generating a repository of cyberattack component models from the Common Attack Pattern Enumeration and Classification (CAPEC) provided by MITRE.

MIKEL D. PETTY is currently a Senior Scientist for Modeling and Simulation at the University of Alabama in Huntsville’s Information Technology and Systems Center and an Associate Professor of Computer Science. Prior to joining UAH, he was Chief Scientist at Old Dominion University’s Virginia Modeling, Analysis, and Simulation Center and Assistant Director at the University of Central Florida’s Institute for Simulation and Training. He received a Ph.D. in Computer Science from the University of Central Florida in 1997. Dr. Petty has worked in modeling and simulation research and education since 1990 in areas that include verification and validation methods, simulation interoperability and composability, and human behavior modeling. He has published over 200 research papers and has been awarded over $16 million in research funding. He has served on both National Research Council and National Science Foundation committees on modeling and simulation, is a Certified Modeling and Simulation Professional, and is Editor-in-Chief of the journal SIMULATION: Transactions of the Society for Modeling and Simulation International. He has been dissertation advisor to seven graduated Ph.D. students in three different academic disciplines (Computer Science, Modeling and Simulation, and Industrial and Systems Engineering).