Component Selection Process in Assembling Cyberattack Simulation Models

Katia P. Mayfield  
Dept. of Mathematical, Computing and Natural Sciences  
Athens State University  
Athens, USA  
katia.mayfield@athens.edu

Mikel D. Petty  
Dept. of Computer Science  
The University of Alabama in Huntsville  
Huntsville, AL  
pettym@uah.edu

Tymaine S. Whitaker  
Dept. of Computer Science  
The University of Alabama in Huntsville  
Huntsville, AL  
tsw0003@my.uah.edu

Walter A. Cantrell  
College of Computing and Technology  
Lipscomb University  
Nashville, TN  
alcanitrell@lipscomb.edu

Scott M. Hice  
Dept. of Mathematical, Computing and Natural Sciences  
Athens State University  
Athens, USA  
shice@my.athens.edu

Jeremiah McClendon  
Dept. of Mathematical, Computing and Natural Sciences  
Athens State University  
Athens, USA  
jmcclen3@my.athens.edu

Pedro J. Reyes  
Dept. of Mathematical, Computing and Natural Sciences  
Athens State University  
Athens, USA  
preyes@my.athens.edu

Abstract - Studies in cybersecurity are being conducted ranging from algorithms to the modeling and simulations of cyberattacks. With all of research and initiatives towards cyber security, the first step is being able to understand the vulnerabilities that make an infrastructure non-secure and the process that is followed by attackers to exploit those vulnerabilities. Using models to understand the steps that an attacker is required to take allows for the graphical representation of the steps taken by the attacker and leads to the ability to simulate these attacks where defense techniques may be studied. Cyberattacks may involve multiple attack techniques and be a combination of attack previously modeled. This study covers a selection process of basic components in which single models or other basic elements may be composed in a broader cyberattack model.

Keywords: Cyberattacks, cybersecurity, selection, components, models

I. INTRODUCTION

The study of cybersecurity must assume that an attacker may be able to combine different strategies to achieve its goal. The possibility of an attacker utilizing multiple attack techniques results in the protection of the target system to become more complex. This study focuses on the selection and composition of different attack patterns into a new model that represents the multiple techniques as a single cyberattack. The basic model under study, Petri nets with players, strategies and cost (PNPSC), is an enhanced view of the traditional Petri nets, including the analysis of actions taken by attackers and defenders and the consideration of costs that are involved as consequences of those actions.

Considering all the different cyberattacks that now exist, attack patterns have been researched and labeled as a knowledge resource instrumental on the design of secure software [1]. 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 attack process.

There have been several studies conducted on cyberattacks and proposing solution on how to model such attacks. Pan et al. 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 [13]. Yeole et al. 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 [16]. Navdeep Kaur and Parminder Kaur attempt to provide a way of integrating attack mitigation into the early stages of the system development life cycle. They use threat modeling to plan 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 where, within the system development lifecycle, the SQL Injection vulnerability would be introduced [6]. Most of these studies are addressing specific situations and do not apply to all reported attacks.

Petri nets have been used to model different systems; in particular, it has been applied to cyberattacks. Khan et al. 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]. The modeling of the attack behavior and consequent reaction of the security systems becomes an essential source of information for analysis and planning of defense mechanisms. He and Zhu describe an approach which establishes trust between strangers through the exchange for digital credentials needed in peer-to-peer networks and grid computing. They model
the policies in participating negotiation as a Petri net and propose a Strategy based on Negotiation Petri nets (SNPN) [4]. The Petri net models address specific situations and do not consider the possibility of composing attacks. This study utilizes the basic Petri net structure in a broader analysis of cyberattacks and focuses on selecting components when assembling new models. The next section presents a discussion of the Common Attack Pattern Enumeration and Classification (CAPEC) which documents an execution flow report demonstrating a pattern that an attacker takes to perform their attack. These patterns can then be modeled to analyze the evolution of the attack and the impact that it will have.

Some studies of modeling cybersecurity include projects in attacks on computer networks, malware propagation using agent-based modeling, and game theory [2, 5, 8]. In particular, MITRE Corporation developed an online database called the Common Attack Patterns Enumeration and Classification (CAPEC) which documents over 500 different attacks, how they are performed, their goals, and mitigations [12]. The CAPEC report does not specify if the attacks are performed by a single attacker or an organization, the assumption is made that the attacks can be completed by either. Even though there is no documentation on the number of attackers that are performing the attack, the patterns do describe different methods or techniques that can be used to achieve a specific attack goal. An attacker may prefer to run one attack over another due to their own knowledge of being able to control the process, however since he or she has options, these must be considered. The attacker may combine two or more techniques or sections of them, creating a new attack pattern, to improve the chances of success, requiring a specific study of such situations.

The basic model under study, Petri nets with players, strategies and cost (PNPSC), is an enhanced view of the traditional Petri nets, including the analysis of actions taken by attackers and defenders as also costs involved [9]. As stated by Petty et. al, the formalism for PNPSC’s has been determined and a method to automatically generate PNPSC cyberattack models based on available information from an attack in the CAPEC vulnerability database has been implemented and associated to a pattern completion score. The formalism and additional details of PNPSC nets can be found in [9].

The goals of this study is to determine how to compose the cyberattacks based on the different components that are available. When assembling the cyberattack for the target computer systems, the components will be selected from a repository. The repository will be accessed through a web application and based on user specifications, the components will be selected and assembled in the requested cyberattack model. These same components may also be reused in multiple cyberattack models for different target computer systems.

In a study not related to cybersecurity, Hamadi and Benatallah proposed a Petri net based algebra for modeling the composition of Petri nets representing Web services control flow [3]. They established a grammar and used Backus Normal Form (BNF) notation to describe the semantics of algebra operators used to combine the Petri nets. Web services were their target for Petri Net designs, and the algebra grammar syntax was focused on its application to the Web. Even though their research was geared towards Web services, the authors provided a general theoretical framework for Petri net composition.

They defined Petri nets composition operations based on Web service models containing a single start state and one final state. Their definitions allowed the Petri nets to be composed by using the following operations, (1) sequential, (2) alternatively, (3) arbitrary sequence, (4) iteratively, (5) in parallel with communication taking place, (6) discriminatorily, (7) through dynamic selection, or (8) refinement [3]. Such operations can be briefly described as:

- sequential, where a Petri net follows another
- alternatively, when two Petri nets can be applied and one of them is activated
- arbitrary sequence, similar to sequential except that there is no definition of which Petri net is the first one to be executed
- iteratively, representing the repetition of a Web service multiple times
- in parallel with communication, concurrent execution of two Petri nets with possible information exchange
- discriminatorily, execution of the same task in different Web services providers (what should be equivalent to the concurrent execution of the same Petri net with different parameters)
- dynamic selection, multiple Petri nets are source to a selection process, which is similar to the alternative composition with the added selection process
- refinement, which allows a Petri net operation be replaced by a more in-depth representation of its behavior.

Similar study in the area of Web Services and Petri Nets was conducted by Yang, et al. [15]. The authors use colored Petri nets to analyze the performance of behavioral properties for web services composition. Their focus is on
the verification techniques which allow for designers to test and repair errors before running a service. The verification of their composition is done through traversing an XML document to show that the composition simulation is guaranteed to terminate, the composition which is based on model transformation yields a unique result, and it is found to be complete. The authors verify that the syntactic correctness can be presented by checking it against a generated colored Petri net metamodel. However, the authors do not have a unique way to verify the technique of semantic correctness or effectiveness of the transformation.

B. Completion Score

The CAPEC report has over 500 attack patterns documented. However, many attack patterns have incomplete or insufficient information to model PNPS nets. Finding complete attack patterns in the CAPEC report were initially a manual, and time-consuming process. A scoring assessment tool was created to serve as a preliminary decision tool, and to assess the completeness of a CAPEC entry attack pattern [14]. The tool essentially searches a single CAPEC pattern to find the major components such as prerequisite, explore, experiment, exploit, and goal phase. The tool calculates and assigns a score to indicate complete or missing components. An overall score of 20 is calculated and assigned to a CAPEC pattern when all major components are present, which is the basis for creating PNPS nets. Fig. 1 shows a partial report generated by the tool. The completion score is calculated by summing the score of each attack pattern component. The tool will assign an “x” if a component is present, a “0” if a component is missing, and a “d” if only the description was given and not the attack pattern technique. A “d” cannot be assigned to the prerequisite and goal components because they are not part of the execution flow of the attack pattern.

![Fig. 1. CAPEC Scoring Assessment Report](image)

C. Model Components

According to Mayfield et al. there are two different types of components to work with. The PNPS nets are themselves considered to be a component at a coarse grain level. These components represent single cyberattack patterns that are capable of being modeled based on an assigned completion score.

Within the CAPEC database there are several attack patterns that serve as a parent of or as a child of other patterns. A parent pattern is one which will include one or more full child pattern as one of the ways in which an attacker is able to accomplish their attack. In this case a coarse-grained PNPS component is based on the full model design of composing several CAPEC attack patterns.

In a PNPS net, each transition has an associated rate which is interpreted as the rate parameter \( \lambda \) of an exponential distribution representing the transition’s stochastic inter-firing time. When multiple transitions are simultaneously enabled, an exponentially distributed random variate is generated for each enabled transition using the transition’s rate; then the transition with the smallest random variate, i.e., the smallest inter firing time, is fired. PNPS nets also include a set of players, each of whom may observe the markings of a portion of the net’s places. Based on the observed marking, and thus the state of the modeled system the marking represents, each player may act by changing (either increasing or decreasing, within limits) the rates of the subset of the transitions that the player controls. Changing transitions’ rates either increases or decreases the likelihood of those transitions firing. In that way the players seek to bring the marking of the PNPS net, i.e., the state of the modeled system, to be consistent with their goals. The firing of a transition can represent a state change caused by an action taken by one, or sometimes more than one, player, and so each transition firing may have a cost to one or more players. Players attempt to achieve their goal markings while simultaneously minimizing their total cost incurred.

A brief introduction to the model starts with a standard Petri net which is formally defined as a 6-tuple \( PN = (P, T, W, L, M_0, B) \), where

- \( P \) = \{p1, p2, ... \}; finite, non-empty set of places
- \( T \) = \{t1, t2, ... \}; finite, non-empty set of transitions
- \( W = W_i \cup W_o \), where \( W_i \subseteq (P \times T) \); set of arcs from places to transitions and \( W_o \subseteq (T \times P) \); set of arcs from transitions to places
- \( L \): \( W_i \rightarrow \mathbb{Z}^+ \), with \( 0 \leq M_0(p) \leq B(p) \) for every \( p \in P \)
- \( M_0 \): \( P \rightarrow \mathbb{Z}^+ \)
- \( B \): \( T \rightarrow \mathbb{Z}^+ \)
- \( d \): \( D \rightarrow \mathbb{Z}^+ \)

This basic definition of Petri nets was the basis for the design of PNPS nets, which are formally defined as an 8-tuple \( PNPS = (PN, G, \Theta, O, F, \Omega, \Gamma, X) \), where

- \( PN \); standard Petri net, as defined above
- \( G \); finite, non-empty set of players
- \( \Theta \); partition of transition set \( T \) into non-controlled and player-controlled transitions
- \( O \); partition of place set \( P \) into places observable by each player
- \( F \); firing rates for non-player-controlled transitions
- \( \Omega \); max firing rates for player-controlled transitions
- \( \Gamma \); mapping from observable markings to firing rates for player-controlled transitions
- \( X \); mapping from marked end places to firing rate

Finally, PNPS was later expanded to allow cost analysis of defending against cyberattacks, resulting in a Petri net with Players, Strategies, and Cost, which is defined as a 3-tuple \( PNPS = (PNPS, F, D) \)

**PNPS**: Petri net with Players, and Strategies as presented above.

\[ T: T \rightarrow Z^+ \cup \{0\} \]; cost for firing a transition
\[ D: T \rightarrow \mathbb{R}(G) \]; power set of players, identifying all possible subsets of players charged for fired transitions.
Fig. 2 shows the component representation of an attack qualified as parent in the CAPEC report. This example is one in which three different cyberattack patterns can be attempted. The result in the end will be the attacker being successful, blocked by a defender, or fail. In the figure, the attacker ready, attacker succeeds, attacker fails, and attacker blocked places are representative of “plug-ins” that are also found within the PNPSC of the components.

Based on this formalism, compositions have been defined in a previous publication as a result of a parallel combination [10] of attack patterns represented by two PNPSC nets $N_1$ and $N_2$ as:

$N_1 = (N_1, N_2)$

$N_2 = (N_1, N_2)$

$P = P_1 \cup P_2$

$N_1 = (P_1, T_1, W_1, L_1, M_{01}, B_1, ES_1, EF_1)$

$N_2 = (P_2, T_2, W_2, L_2, M_{02}, B_2, ES_2, EF_2)$

The resulting PNPSC $N$ of a parallel composition of $N_1$ and $N_2$ is:

$N = (P, T, W, L, M_0, B, ES, EF)$ such that:

$P = P_1 \cup P_2$

$T = T_1 \cup T_2$

$W = W_1 \cup W_2$

$L = L_1 \cup L_2$

$M_0 = M_{01} \cup M_{02}$

$B = B_1 \cup B_2$

$ES = ES_1 \cup ES_2$

$EF = EF_1 \cup EF_2$

With $F, \Gamma, X$ defined as:

$F = F_1 \cup F_1 \Gamma$

$X : M(ES_1 \cup ES_2) \rightarrow \mathbb{R}$

A second type of component exists at a fine grain level. Each modeled attack pattern can be made of four phases, (1) explore, (2) experiment, (3) exploit, and (4) goals. Within the first three phases, the attacker is shown to be able to take actions associated to using specific techniques to fulfill that phase, on the fourth phase it is not techniques that are displayed but the goal that the attacker is trying to achieve. With the concept of fine grain component, each technique or goal that is modeled is represented as its own fine grain component.

Each phase has a technique or method that the attacker will follow to complete their attempt of that phase of the attack and move through the attack until their goal is achieved. A fine-grained PNPSC component is a subnet that is specifically designed to represent a method or technique that can be modeled in each of the four phases. Therefore, given an existent PNPSC attack pattern, a component subnet is formally defined as:

$\text{subPNPSC} = (\text{entryPlaces}, \text{exitPlaces}, \text{subPNPS}, subF, subD)=$ where:

$\text{subPNPS} = (subP, subT, subW, subL, subM_0, subB)$

- $\text{entryPlaces} = \{ep_1, ep_2, \ldots \}$
- $\exists t_i \in (T-\text{subT});$
- $\text{exitPlaces} = \{xp_1, xp_2, \ldots \}$
- $\exists (xp_i, t_i) \in (T-\text{subT});$
- $\text{subP} \subseteq P$ such that $v \in entryPlaces \ \forall v \in exitPlaces \ V(\mathcal{A}(v, l) \cap \mathcal{A}(u, y)) | t_i \in (T-\text{subT});$ no outgoing/incoming arcs from places inside the component. The count (or weight) for each arc representing the number of tokens that can be transmitted, the value -1 implies a blocked arc from a place to transition.
- $\text{subM_0}; \text{subP} \rightarrow \mathcal{Z} \cdot \mathcal{Y} \cdot \{1\};$ initial marking of tokens in places, with $0 \leq \text{subM_0}(p) \leq \text{subB}(p)$ for every $p \in P$
- $\text{subB}; \text{subP} \rightarrow \mathcal{Z} \cdot \mathcal{Y}; \text{upper bound on tokens per place}$
- $\text{subG}; \text{partition of transition set subT into non-controlled and player-controlled transitions}$
- $\text{subO}; \text{partition of place set subP into places observable by each player}$
- $\text{subQ}; \text{max firing rates for player-controlled transitions}$
- $\text{subF}; \text{mapping from observable markings to firing rates for player-controlled transitions}$
- $\text{subF}; \text{subT} \rightarrow \mathcal{Z} \cdot \mathcal{Y} \cdot \{1\}; \text{cost for firing a transition}$
- $\text{subD}; \text{subT} \rightarrow \emptyset \mathcal{G}; \text{power set of players, identifying all possible subsets of players charged for fired transitions.}$

Fig. 3 shows the representation of a PNPSC method that can be applied during the Explore phase of a cyberattack. Within this diagram the places that are double circled are considered to be “plug-ins” found within the current modeling process. The places that are double circled and bold are also “plug-ins,” these take into consideration the
possibility of having to track if an attacker fails during their exploration methods, which at the moment is not tracked within a regular PNPSC. Based on the example shown in Fig. 3, there are two techniques that the attacker can follow to achieve the explore phase of the attack, one technique follows the transition labeled A1 while the second is A6. The dashed lines that lead to a “Bypass” are also not found within all of the PNPSC pattern models. Some will have techniques within a phase to be of an optional nature, and therefore the ability to bypass that entire phase needs to be available.

Fig. 4 shows the representation of a PNPSC where the Explore phase would be designed as a “placeholder” (the cloud in the diagram) for the ability to “plug-in” components. By following composition rules described by Mayfield et al. additional components may be added in parallel in the explore phase [9].

Fig. 3. Pre-requisite & Explore Phase of PNPSC.

The places and transitions that correspond to a component subnet within the explore phase are shown in Fig. 5. There is one input place which is also found in the PNPSC design and is used as a “plug-in” and there are three output places, one of which (A11) is not part of the usual explore phase design, however must be included with the possible future design of PNPSC’s.

By having two different types of components there are selection strategies that correspond to each one which is discussed next.

III SELECTION

Yuan et al. describe the development of a tool that is used to retrieve relevant CAPEC attack patterns for software development, based on patterns that are relevant to STRIDE (Microsoft’s SDL, Security Development Lifecycle, tool). The tool is able to retrieve relevant attack patterns of a certain STRIDE threat category, sorted from most relevant and useful to least relevant and useful. They proposed a metric $U$ that measures the usefulness or importance of an attack pattern based on information found in the CAPEC report and STRIDE categories. They also mentioned that the tool was being modified to allow for attack patterns to be searched based on keywords. Their solution focuses on the development cycle of an application [17]. In this study, the goal is to model and simulate cyberattacks to any systems based on system vulnerabilities, including those already in production. Part of being able to model and simulate these cyberattacks is the model selection process.

A. Repository

The purpose of the repository is to be able to have a place where specific cyberattack models can be searched for, based on filtering capabilities provided to the user to narrow down the list of models that are available from the 500+ patterns that are documented through CAPEC. The repository could be used by an attacker who is trying to figure out which attacks he should attempt. Therefore, it must be used by a defender who is trying to figure out ways that an attacker can cause a breach, or just a regular user who is wanting to see if their system is susceptible to any breaches.

The repository holds the CAPEC records and information of designed PNPSC models. The interaction with the repository utilizes a web interface which allows the user to manipulate data from the database.
The user stories that are kept in mind for the creation of the repository are as follows:

- filter the database by specific attack criteria
- enter new attack entries into the database
- search for attack patterns with matching purposes
- search for attack patterns by:
  - level of CIA impact
  - level of typical severity
  - the likelihood of exploit
  - method of attack
  - abstraction
  - attack motivation
  - associated vulnerabilities
- store PNPSC models
- modify PNPSC models
- enter new PNPSC models
- import attack pattern models (.dot files) into the repository
  - be able to press a button to browse file explorer to select a .dot file
  - be able to drag and drop the file into the GUI to select a file
  - know if the file selected is valid (file extension, syntax, empty file)
  - after verifying the selected file is valid, be able to hit a submit button to push to repository
  - see the results of any submission (error check)

### B. Coarse Grain Selection

The selection process identifies the target system and therefore its weaknesses or the CAPEC report identifies specifically the model number to be used.

Example: A search on the consequences associated to the attack patterns, in the repository known as CIA Impact, that consists of “confidentiality, access control, authorization, and integrity” will return a list of 158 attack patterns with completeness scores that allow for the pattern to be modeled. Selecting the “Graph/Relationship” for Leveraging Race Conditions we see that it has relationships to Leveraging Race Conditions via Symbolic Links and Leveraging Time-of-Check and Time-of-Use Race Conditions, as show in Fig. 6.

![Fig. 6. Selection of Related Attach Patterns](image)

If the Graph/Relationship is clicked here for CAPEC pattern 27 or 29 it will lead back to pattern 26. Pattern 26 is a parent of both 27 and 29. Fig. 7 shows the completion report for attack patterns 26, 27, and 29. If the Completion Report is reviewed, one can see that only descriptions are provided for reports 26 and 29 while 27 is a fully complete pattern.

With all three patterns being identified as at least minimally complete to be modeled they can be selected as coarse grain components to be composed into a more broader attack pattern instead of individual single patterns. With the metadata that is used to identify these patterns, the selection process becomes a simple list search for matching CAPEC patterns and therefore its complexity is order $k$ where $k$ is the size of the repository. The algorithm CGS (Coarse Grain Selection) shows the necessary steps to identify this selection approach.

**Algorithm CGS:**

T: Model to be composed
R Component repository
$S = \{\text{Sub-model ID's used in CAPEC report of } T\}$
$C = \emptyset$

For all models $M$ in $R$

- If $M \in S$ 
  - & completion_score > user_threshold, 
    - extract model $M$
    - $C = C \cup M$


### C. Fine Grain Selection

In a fine grain level composition, the selection process will need a more specific algorithm where a greedy solution is used (heuristics). The target system is identified and based on its weaknesses, the attacker goals are established. The selection process starts by selecting all the goal phase components that match the target system vulnerabilities. This will be order $k$. After these components have been identified, their input places are matched against exploit phase components output places (there must be a match in order to have interaction and therefore composition).

This selection is also of order $k$ for each goal element previously selected (resulting in a $k^2$ complexity). The heuristics proceed with identifying experiment phase components whose output places match the selected exploit components, again order $k$ for each selected exploit component. Next, the heuristics will select explore components matching the experiment components. However, an attacker may have enough knowledge to where they do not have to explore a system, therefore the input places for the experiment phase must match prerequisite knowledge that the attacker must have to be able to perform the experiments. If the attacker is not capable to skip the explore phase, the last step is to find the input places for the explore phase which must match prerequisite knowledge that the attacker must have to be able to accomplish those techniques. The best case where the attacker does not need to go through an explore phase makes this equivalent to a two-level nested loop at each selection

![Fig. 7. CAPEC Report Completion Scores](image)
phase, resulting in a complexity of $k^2$. Fig. 8 displays the bottom up approach graphically.

Algorithm FGS:
T: model to be composed
R: component repository
Tg: Attacker’s goal
G = $\emptyset$, E= $\emptyset$, Exp = $\emptyset$, Expl= $\emptyset$;
For all goal phase components X in R
If $X_{exitplaces} = T_{g}$ then $G = G \cup X$
For all exploit phase components Y in R
For all X in G
If $Y_{exitplaces} = X_{entryplaces}$ then $E = EUY$
For all experiment phase components W in R
For all Y in E
If $W_{exitplaces} = Y_{entryplaces}$ then $EXP = EXP \cup W$
For all explore phase components Z in R
For all W in EXP
If $Z_{exitplaces} = W_{entryplaces}$ then $EXPL = EXPL \cup Z$
Do fine grain composition of G, E, EXP, EXPL (equivalent to $T = G \cup E \cup EXP \cup EXPL$)

IV CONCLUSION

The concept of components for the use of reusability is a well-known Software Engineering concept. The decomposition of single cyberattack models to create fine grain coarse components allows for the new composition of other undocumented patterns, while the coarse grain components allow for a more in-depth and full cyberattack pattern. Petty et. al have shown the composition selection problem to be NP-complete. However, with specific requirements and heuristics it has been shown that the component selection problem can be solved in polynomial time. This study shows that it is possible to select components of cyberattack models to compose more in-depth models in polynomial time and specifies the requirements and heuristics that must be followed for this to occur.

REFERENCES