Security Configuration Analytics Using Video Games

Mohammed Noraden Alsaleh and Ehab Al-Shaer
University of North Carolina at Charlotte
Charlotte, NC, USA
Email: {malsaleh, ealshaer}@uncc.edu

Abstract—Computing systems today have a large number of security configuration settings that enforce security properties. However, vulnerabilities and incorrect configuration increase the potential for attacks. Provable verification and simulation tools have been introduced to eliminate configuration conflicts and weaknesses, which can increase system robustness against attacks. Most of these tools require special knowledge in formal methods and precise specification for requirements in special languages, in addition to their excessive need for computing resources. Video games have been utilized by researchers to make educational software more attractive and engaging. Publishing these games for crowdsourcing can also stimulate competition between players and increase the game educational value.

In this paper we introduce a game interface, called NetMaze, that represents the network configuration verification problem as a video game and allows for attack analysis. We aim to make the security analysis and hardening usable and accurately achievable, using the power of video games and the wisdom of crowdsourcing. Players can easily discover weaknesses in network configuration and investigate new attack scenarios. In addition, the gameplay scenarios can also be used to analyze and learn attack attribution considering human factors. In this paper, we present a provable mapping from the network configuration to 3D game objects.

I. INTRODUCTION

Security hardening is the process of discovering and measuring security risks, identifying weaknesses in security controls, and developing risk mitigation plans. It has been widely cited that this is the most challenging problem in the area of cyber security [3], [5], [12]. This issue becomes particularly challenging due to the computational and space requirements that grow non-linearly with the size of the system. Moreover, configuration management and verification tools are complex and require special knowledge and training. Furthermore, a great deal of uncertainty is unaddressed during the process of security hardening due to the lack of complete knowledge and the lack of requirements, which leads to the existence of unknown risks. On the other hand, misconfigurations and vulnerabilities widen the attack surface of computing systems. In order to defend against attacks, we need to characterize them and study the behavior of attackers, as well as the misconfigurations that can lead to their success.

Recently, computer games have been widely investigated in many disciplines ranging from business and health to education in order to complement existing learning and analysis methods [17], [23]. Computer games are considered a cost-effective way to transfer knowledge and skills among players and enhance their response by virtually simulating incidents. Literature surveys of the effectiveness of computer games revealed that they have a positive effect on learning [6]. We believe that game-based interfaces for security analytics and hardening are more usable than logic-based interfaces, as in traditional tools. Hence, network administrators can discover risks and vulnerabilities through gaming methods faster than by merely using formal analysis techniques for large-scale networks. Moreover, a crowd sourced gaming approach to discover security problems in large-scale networks may produce more comprehensive reports (i.e. discover a larger attack surface) than using traditional verification tools. Particularly, the crowd can hit the configuration weaknesses or unknown vulnerabilities that are not discoverable through formal verification techniques due to lack of resources or knowledge. We aim at integrating both crowd sourced games and formal verification methods, so we can significantly reduce the search space. Using the traces collected from gameplay, we can learn significantly more about network behavior, risk, and attack attributes. The game can also explore human factors that drive the attackers’ behavior and investigate the resistance of networks against attackers under various motivations and persistency levels, capabilities, and financial abilities.

Our contributions featured in this paper aim to provide a framework to build a maze-like game out of the network topology, configuration and attack models. The game will be publicly available for the crowd. The framework provides the means for administrators to define the security requirements and the attack goals, which are reflected as objectives in the game. The players (unaware of playing against a real network configuration) need to beat the challenges in game levels. Beating a challenge corresponds to a success of an attack. The presented game-based interface provides the following:

- It provides more usable interface for the administrators than does traditional verification techniques to better understand their network configuration and, at the same time, validate the configuration against attack models.
- It gives a unique chance for exploring attack attribution by utilizing the human intelligence of the players. This is accomplished by a comprehensive mapping of the attackers’ attributes to appropriate attributes of players in the game. Attackers in real life may find themselves in a situation in which they have to choose from different options to proceed in their attacks. To select an appropriate option, they can rely on their own intelligence or they can employ other analytics and systematic approaches to assess their situation and select the best option [15]. We believe that we can mimic this in the game and put the player in a similar situation in which he may follow the same decision process as the attackers do in real life. Analyzing the players’ behavior under such situations can shed light on the attackers’ way of thinking.
- The game is appealing and provides the challenge as well as the fun for the players in order to attract large crowds.
who can collaboratively help discover vulnerabilities and misconfigurations in large scale networks.

Unlike traditional maze games, the actions of players should comply with a set of constraints derived from the network configuration. We use Binary Decision Diagrams (BDDs) to encode the network configuration and extract these constraints. The framework models the full network configuration (topology, traffic control devices, and end hosts) and maps them into the game domain as game objects. This allows for a one-to-one translation between a game action and a user activity in the network. Attack attributes (pre-conditions and post-conditions) are also mapped to players’ capabilities. To draw the world layout from the network physical topology, we use a graph drawing algorithm based on the lombardi drawing style [7]. We will use the Unity game engine [22] to model the game in the 3D world.

The rest of the paper is organized as follows. In Section II, we present the network configuration and attack models. The game model is presented in Section III, and in Section IV, we explain the mapping of objects between network and game domains. Section V provides the general architecture of our framework and the implementation. In Section VI, we provide two case studies to evaluate the framework. The related work is presented in Section VII. Finally, we conclude our analysis in Section VIII.

II. NETWORK AND ATTACK MODELS

In this section, we describe the network and attack models. The network model consists of its topology and policies. The attack model describes the exploits that can be performed by attackers.

A. Network Behavior

Figure 1 shows an example of network configuration. The network is modeled as the 4-tuple \( G = (V, I, L, P) \) where

- \( V \) is a set of network devices including firewalls, routers, IPSec gateways, intrusion detection systems, and end nodes (hosts). Each device is associated with several interfaces and a policy.
- \( I \) is a set of communication interfaces. The interface controls the traffic between two devices based on its policy and can generally perform one or more of the following functions: forwarding (routers), filtering (firewalls), transformation (IPSec), or inspection (IDS).
- \( L \) is a set of links between devices.
- \( P \) is a set of policies. If the device has more than one interface, the policy determines which rule applies to which interface.

Table I shows how the configuration is translated to policies for the network shown in Figure 1. The policy shown in Table I defines only the values that restrict the traffic flowing in each interface.

![Network Model Example](image)

TABLE I: Policies for interfaces shown in Figure 1

<table>
<thead>
<tr>
<th>Interface</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_{R1} )</td>
<td>( \text{Src}_ip = 20.0.0.5 )</td>
</tr>
<tr>
<td>( I_{R2} )</td>
<td>( \text{Dest}_ip = 20.0.0.5 )</td>
</tr>
<tr>
<td>( I_R )</td>
<td>( \text{Dest}_ip \neq {20.0.0.5, 10.0.0.5} )</td>
</tr>
<tr>
<td>( I_{RF} )</td>
<td>( \text{Dest}_ip = 10.0.0.5 )</td>
</tr>
<tr>
<td>( I_{FR} )</td>
<td>( \text{Src}_ip = 10.0.0.5 )</td>
</tr>
<tr>
<td>( I_{F2} )</td>
<td>( \text{Src}_ip = 20.0.0.5 \land \text{Dest}_ip = 10.0.0.5 )</td>
</tr>
<tr>
<td>( I_{F2} )</td>
<td>( \text{Src}_ip = 10.0.0.5 )</td>
</tr>
</tbody>
</table>

The network configuration also includes the hosts configuration. A host definition consists of three major sections: applications, vulnerabilities and settings as shown in Table II. The applications section contains a list of applications and services running on the host. Each application or service has a unique name that can, optionally, include the vendor and version number. Standard names from the Common Platform Enumeration (CPE) can be used for common platforms. We assume that the same names are used in hosts definitions and exploits definitions. The second section contains a list of identified vulnerabilities in the host. The vulnerabilities are collected using vulnerability scanning tools and can be referenced by their standard names in the Common Vulnerabilities and Exposures (CVE) dictionary. Finally, the settings section contains a list of standard configuration settings with their values. Each entry is defined as a \(<\text{name}:\text{value}>\) pair. The Common Configuration Enumeration (CCE) names can be used for standard settings.

![Host Configuration](image)

TABLE II: Host Configuration

| Host: accounting_db_1 (192.168.20.30), assets: $10,000 |
| Applications | Windows server 2008 |
| Vulnerabilities | Microsoft SQL Server Enterprise edition |
| Settings | CVE-2013-3172, CVE-2011-1894 |

B. Attack Model

The attack model in our framework is defined as a set of exploits where each exploit has a list of attributes that includes: cost, conditions and effects. Attackers assess their situation and the available exploits to select their next step.

1) Attacker Attribution: The attacker behavior is driven by several factors that influence his decisions in each step of the attack. We summarize these factors by the following.

- **Motivation.** This may be the most crucial factor in the analysis of attacker’s behavior. Several types of motivations has been identified in literature [18] including: money, entertainment, ego, ideology based on culture or social issues, and status among hackers groups.
Fig. 2: High level view of the game world. In the full implementation of the game, this world will be extended by including other non-functional objects such as buildings and trees to enrich the environment and make it more realistic and engaging.

- **Resources.** Exploits are associated with costs, the computing resources or financial capacity of attackers may restrict his ability to achieve the attack goals. Attackers need to manage their resources to keep going.

- **Skill Level.** The attacker’s experience of network and systems’ security increases the probability of successful attacks. Different exploits require different skill levels.

- **Persistence.** Security exploits vary in the effort required to perform them. Some require continuous effort and some may require long time. Attacker’s persistence captures his tendency to select such exploits. For example, script kiddies are most likely looking for quick results, while attackers motivated by money are more persistent.

- **Temerity.** This factor captures the prowess of attackers to conduct exploits as some of them may have negative consequences and may expose him to law enforcement authorities. Attackers with different motivations and in different locations need to determine the risk they are willing to tolerate before conducting exploits. For example, students at a school will risk their enrollment if they were caught abusing their privileges to attack school network. Other legal aspects may also abate attackers from performing particular exploits on particular systems.

Attackers adopt several approaches ranging from manual analysis to more sophisticated analytics such as game theory, optimization, or simulation to model these factors and drive their decisions. Moreover, large organized attackers’ groups and criminal communities provide venues for sharing strategies, skills and tools and for combining forces to launch more effective and stealthy attacks.

2) **Exploits:** Several attack description languages have been proposed to comprehensively express attacks such as ADeLe [16] and LAMBDA [9]. ADeLe is an attack description language to model a database of attack scenarios. ADeLe also provides means to express schemes for attack implementation, detection, reporting, etc. For the purpose of our framework, we use a description language adapted from ADeLe to describe the exploits. The adapted language is limited to the description of the exploits in terms of: costs, conditions and effects.

**Definition 1.** An attack action is defined as the transition \( \tau : S_i \rightarrow A \rightarrow S_f \), where \( S_i \) and \( S_f \) are two system states.

A state \( S = \{ S^a, S^n, S^h \} \) represents the state of three components: the attacker \( S^a \), the network configuration \( S^n \), and the hosts \( S^h \). The attacker state \( S^a \) is characterized by a set of constraints of the form \( <\text{loc, priv}> \) which specifies that the attacker has at least the privilege \( \text{priv} \) on the device \( \text{loc} \). The network configuration state \( S^n \) is represented as a set of reachability constraints expressed by the operator \( \text{CanReach} \). A constraint of the form \( <\text{CanReach}(\text{src, dest})> \) specifies that host \( \text{dest} \) is reachable from host \( \text{src} \). The \( \text{src} \) and \( \text{dest} \) parameters can be represented as an \( <\text{address/name: port}> \) pair. The host configuration state \( S^h \) describes the services/applications running on each host, the configuration settings, and the vulnerabilities. A constraint of the form \( <\text{loc, srv, conf, vul}> \) specifies that the host \( \text{loc} \): (i) is running the set \( \text{srv} \) of services, (2) has the configuration settings \( \text{conf} \) (as pairs of \( <\text{setting,value}>) \), and (3) has the set \( \text{vul} \) of vulnerabilities.

Each exploit has also a set of attributes \( \mathbb{A} \) includes: the type, the cost, the required skill level, the time required to perform the exploit, and for some exploits the probability of success. We assume two types of exploits based on the attacker location: local (can be performed only if the attacker is logged in the machine itself) and remote (can be conducted on a remote target). The cost is a monetary value. The time and probability of success capture the required persistence level. The exploit may require large time or it may need to be performed many times to be successful.

<table>
<thead>
<tr>
<th>Action:</th>
<th>client-scripting.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>remote.</td>
</tr>
<tr>
<td>Cost:</td>
<td>$500.</td>
</tr>
<tr>
<td>Skill:</td>
<td>0.</td>
</tr>
<tr>
<td>Prob:</td>
<td>1.</td>
</tr>
<tr>
<td>Network</td>
<td>CanReach(Src, Tar : 80)</td>
</tr>
<tr>
<td>Host</td>
<td>Tar: XEplorer 5.0</td>
</tr>
<tr>
<td></td>
<td>Tar: &lt;HTML-Scripting, true&gt;</td>
</tr>
</tbody>
</table>

**TABLE III: Attack action example**

Table III describes an example of exploits. The variables \( \text{Src} \) and \( \text{Tar} \) are used to indicate the current location of the attacker and his target respectively. The first column describes the constraints of the current state. The second column describes the next state which captures the effects of performing the exploit. The effects might include an escalation in attacker’s privileges, a disclosure or corruption of critical information, hindering the operation of a particular service, injecting false data, creating backdoors, etc.

III. GAME MODEL

We model the network topology as an interactive maze. The traditional model of maze games can be represented as a combination of trails organized in such a way that confuses the player making it hard to accomplish his goal. The player should navigate through the trails trying to find his way out of the maze. Limited time or number of steps may be required to make the challenge harder. The model we use for our game is
different than the traditional maze models in order to comply with the nature of computer networks. The objective of the player is not always to get out of the maze as we explain later.

A. Philosophy

The completeness of mapping the network configuration and attack models to game structures is essential to return the expected analytical value of the game. The game model should be rich enough to provide the following three functions. First, precise mapping of the network including its topology, core and end-nodes configuration (routing, access control, intrusion detection), as well as services and assets distribution on the hosts. Second, mapping the attacker behavior in terms of decision making. The exact decision space including the major human, financial, and environmental factors that influence the decisions of attackers need to be mapped to appropriate factors that drive the player’s decisions in the game. Finally, the game model need to allow for crowd sourcing and multiplayer collaboration.

B. Game Mechanics

We model the game world as a network of warehouses. The game world contains the three entities: players, warehouses, and items. The player can own, manage, visit subscribe to particular warehouses. The player can progress in the game by utilizing the rewards of performing game actions. The game actions include: trading items among warehouses, installing upgrades, advertising items, or subscribing to particular warehouses. Several items and upgrades may be available for the player at the same time and he has to select between them. Each item or upgrade is associated with a Cost and a Reward. The Cost can be expressed as an arbitrary value or it can be defined in terms of inventory artifacts. The player can select an item or an upgrade if he has sufficient money and he has the required artifacts in his inventory. The Reward of trading an item or installing an upgrade specifies the effects on the player and the game world. Any attribute of the entities shown in Figure 3 may be affected. Table IV illustrates the possible effects for each attribute.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Possible Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player.Wealth</td>
<td>Increase/Decrease by an arbitrary value</td>
</tr>
<tr>
<td>Player.Skill</td>
<td>Increase the skill level of the player</td>
</tr>
<tr>
<td>Player.Warehouses</td>
<td>Discover new warehouses</td>
</tr>
<tr>
<td>Player.Roles</td>
<td>Escalate/Diminish the role in a particular warehouse</td>
</tr>
<tr>
<td>Player.Score</td>
<td>Increase/Decrease by an arbitrary value</td>
</tr>
<tr>
<td>Player.Inventory</td>
<td>Add/Remove defined artifacts</td>
</tr>
<tr>
<td>Warehouse.Items</td>
<td>Discover new items</td>
</tr>
<tr>
<td>Warehouse.Upgrade</td>
<td>Add/Remove upgrades to particular warehouse</td>
</tr>
<tr>
<td>Item.Cost</td>
<td>Increase/Decrease the item cost</td>
</tr>
<tr>
<td>Item.Reward</td>
<td>Modify the return list of a particular item</td>
</tr>
</tbody>
</table>

TABLE IV: Effects of moving items.

2) Game Objectives: The game world consists of a set of game levels. Each level is associated with one or a set of objectives. The objectives are defined in terms of player attributes. Several objectives may be defined for each level. The game operator may also require to accomplish them in a specific order to beat the level.

Definition 3. A game objective is a logical expression over a set of constraints on the player’s attributes. A constraint is
defined as the 3-tuple \((\text{Attribute}, \text{Op}, \text{Value})\). The constraints are combined using logical operators \((\land, \lor, \neg)\).

Where \(\text{Attribute}\) is a player attribute belongs to \{\text{Wealth, Skill, Warehouses, Roles, Score, Inventory}\}. The operator \(\text{Op}\) depends on the type of attribute and can be \{\(=, \neq, >, <, \leq, \geq\)\} for numerical attributes, \{\(\subseteq, \supseteq, \exists\)\} for list attributes such as \(\text{Inventory}\). The following represents an example of game objectives

\[(\text{Inventory} \ni \text{art-1}) \land (\text{Role} (\text{Main}) = \text{Owner})\]

This example specifies that the player needs to find art-1 and become the owner of the warehouse called Main to meet the objective.

IV. Configuration to Game Mapping

The objective of the game is to verify if an attacker can succeed in achieving a particular goal. The attacker conducts his attack provided a set of exploits and capabilities related to the network and hosts configuration, as well as the attack model. The following explains how these configurations are mapped to the game.

A. Network Configuration

The network core configuration is mapped as end-end connectivity rather than mapping the network exact core devices (i.e. routers, firewalls, etc.). The hosts are mapped to warehouses where the paths between them are represented as roads in the game world. In order to find if a path exists, the global network configuration including the forwarding tables, access lists and transformation rules is considered. We use model checking to answer reachability inquiries. The warehouses network is modeled automatically from the configuration files, no human intervention is required.

The basic information of host definition that includes the name and the address of the host is explicitly mapped to the game. If the warehouse name is not provided in the definition, a random name is generated. We do not use the real hosts’ names in the game to avoid leaking information about the actual network. However, we need to keep a mapping between the warehouse name and the host name and address since the host information is used to run the reachability analysis. The remaining sections of the host definition that include the applications, vulnerabilities, and configuration settings are used along with the network configuration to discover the items and upgrades that can be applied to each warehouse. The host configuration mapping is automatic and does not require the operator intervention except for the optional specification of warehouses’ names.

B. Exploits

The exploits are mapped to upgrades and items based on their types (i.e. Local or Remote). The local exploits are mapped to upgrades, while the remote exploits are mapped to trading items. Each item or upgrade is assigned a random name. In addition, recall that exploit definition includes constraints on the current and next states in terms of attacker privileges, network, and host configuration. When the player changes his location, the framework evaluates the constraints of all exploits and extracts those that can be performed in the current state of the game. The extracted exploits are displayed as upgrades or trading items for the player. Performing an exploit affects the attacker by escalating his privileges. This is mapped by elevating the role of the player in warehouses to become a manager or an owner. Exploits can also affect the connectivity of the network by altering the configuration or it can affect the hosts by starting/stopping services, activating new vulnerabilities, or altering the configuration settings. As a result, new exploits may become available for the attacker. This is mapped to the game by reevaluating the available exploits after executing a game action and updating the list of upgrades and items accordingly.

Every upgrade or item is assigned a cost, a skill level, a time, a success probability and a set of rewards. These attributes need to be defined by the game operator and read from the exploits repository. We provide a tool to help the game operator configuring these values. We also provide the option to infer the cost and rewards automatically from the effects of exploits. However, the automatic inference is based on simple static rules and may not be enough to accurately reflect practical attack behaviors. The inference rules are expressed as \textit{IF-THEN-ELSE} rules and can be modified by the game operator. Examples of inference rules include: "\textit{IF the attacker gained root privilege THEN increase his score by the asset value of the target machine\textquoteright}, or "\textit{IF the attacker stopped IIS service THEN add special artifact to his inventory\textquoteright". The time attribute is mapped as constraints on the maximum speed of the player or a waiting time upon delivery.

In practice, attackers learn about the network through scanning and fingerprinting before conducting the exploits. This is mapped by introducing two special game actions \textit{Advertise} and \textit{Subscribe}. The \textit{Advertise} game action is equivalent to scanning the network for reachable hosts. For players in the game, executing the \textit{Advertise} action from a particular warehouse results in the discovery of neighboring warehouses. The player advertise himself by placing phone calls in any order he likes. He also can place bulk calls for a range of phone numbers. The \textit{Subscribe} game action is equivalent to scanning a particular host for services and vulnerabilities. By subscribing in a warehouse, the player identifies which items can be moved to it. The \textit{Advertise} and \textit{Subscribe} game actions have costs set by the game operator. They can be set separately for each warehouse.

C. Attack Goals

The attack goals are mapped to game objectives. The framework does not provide automatic mapping for attack goals. The operator needs to have clear understanding of the game mechanics, as well as the attack model, in order to accurately define the objectives. The attack goals can be distributed into multiple game levels with multiple objectives for each level. The game operator can optionally set the order of levels or the order of objectives in each level. Objectives are defined in terms of player attributes according to definition 3. In the following, we provide a couple of examples for mapping attack goals to game objectives.

- The attack goal "\textit{The attacker gains root privilege on the server <server-name>\textquoteright}" can be mapped to the objec-
tive “Get the ownership of the warehouse <warehouse-name>”.

- The attack goal "The attacker compromises 25% of the network assets" can be mapped to the objective "Achieve a score of 2500". We assume here that the rewards of exploits are configured to increase the score of the player by the victim’s asset value and 2500 represents the 25% of the network assets.

In particular cases, the player’s inventory can be used to capture special attack events. For example, to express the attack goal "stop a particular service running on a critical server", the exploit of stopping the service can be configured to add a particular artifact to the player’s inventory. In this case, the objective for the player will be something like "Capture the artifact <artifact-name>".

### D. Attacker Attribution

Considerable amount of research has been conducted in developing profiles for attackers based on intrusion detection logs and reported incidents. These profiles are used to build models that are incorporated in security hardening or risk management processes [10], [18]. These techniques are good for automated security analysis. However, in most of the cases it is hard to collect complete information about attacks. Moreover, attacker behavior models are based on assumptions that do not always hold. Most of game theoretic models assume that the attackers always play rational for example.

In our framework, we have human players who are the crowd. To study attacks, we do not need to build models that emulate attackers’ behavior; rather, we can utilize crowd intelligence to reveal how attackers will behave if they were in similar situations. The following describe how the factors (discussed in Section II-B) that affect attackers’ behavior are handled in the game.

- **Motivation.** The motivation of players depends on the context in which the game is employed. The game may be played as assignment in classes, in this case the players are motivated by their grades in the class. The game may also be published on game hosting websites and the players can be motivated by a prize. Tournaments may be arranged among online games communities. In these communities players also need to keep high status among other players as it is the case in attackers communities.

- **Resources.** Each player starts with an initial profile that determines his wealth. The wealth determines his ability to conduct game actions. The game actions may decrease or increase his wealth which means he has to manage his decisions in the game to preserve his wealth.

- **Skill Level.** We maintain a skill level for players in the game. The player skill level is raised when he successfully performs game actions and accomplish game objectives. Some game actions or game levels require special skill level. This needs to be configured by the game operator.

- **Persistence.** The persistence of players is reflected as the two game action attributes: the time and the probability of success. The time is handled by enforcing a maximum speed or waiting time upon items delivery. If the probability of success is not 1, the item may be corrupted upon delivery and the player needs to perform the task again.

### V. IMPLEMENTATION

We have implemented a simple functional prototype to demonstrate the idea and evaluate the proposed method. Figure 4 shows the architecture of our framework. The framework takes three main inputs: the network and host configuration, the exploits repository, and the game specification. The first two are described in section II. The game specification contains the game levels and their objectives in addition to the attacker initial profile. The rest of this section provides technical and implementation details for the framework components.

#### A. Exploits Repository

The exploits repository contains the formal descriptions of known exploits expressed in XML. We use the tag `<exploit>` to define exploits. Each exploit has a number of attributes define its ID, type, execution time and probability of success. The attribute `item` may optionally specify the name of the exploit that is shown to the players in the game. The definition also contains a set of constraints on the current and next states of the system. We use the tag `<con>` to define a constraint. The `<con>` attributes: `state` and `type` specify the state and component on which the constraint is applied. Listing 1 shows an example of exploits definitions.

```
Listing 1: Attack action definition

1. <Exploit id="iis−buffer−overflow" type="remote"
   item="wood" time="60" prob="1">
2.   <Con state="c" type="a">SU:ser , r00t</Con>
3.   <Con state="c" type="c">S.T,80</Con>
4.   <Con state="c" type="h">T:w3svv</Con>
5.   <Con state="n" type="a">T:root</Con>
6.   <cost type="wealth">50</cost>
7.   <reward type="wealth" op="add">100</reward>
8.   <reward type="inventory" op="add">art−2</reward>
9.   <reward type="score" op="add">T. asset</reward>
</Exploit>
```
The constraint type can be set as one of three values: attacker, configuration, or host. Constraints of type a are defined as <host:privilege 1, privilege 2, etc.> and used to specify the privileges of the attacker on a specific host. We use \{S, T\} to express the current location of the attacker and his target host respectively. Constraints of type c are defined as the tuple <src, dest, port> and specifies that the destination dest should be reachable by the source src on the port number port. Constraints of type h define the applications and configuration settings of the hosts. They are defined as the pair <host:app> which specifies that app exists on the host.

The exploit definition also defines the costs and rewards. Multiple <cost> or <reward> elements can be used to define multiple costs and rewards. The attribute type \in \{wealth, inventory, score, skill\} is used in cost and reward elements to determine whether the value is money, inventory artifact, score, or increase in player’s skill. The attribute op \in \{add, subtract, multiply, etc.\} specifies how the player attributes should be changed. Notice that, we can define the costs and rewards as functions of the attributes of the source or target hosts as appears in line 9. The score of the player is incremented by the asset value of the target host (asset is an attribute defined in the host definition).

B. Network Configuration

To run reachability queries, we build on top of an existing comprehensive network configuration model checker (ConfigChecker) [4]. ConfigChecker models the network as a single monolithic finite state machine. The transitions depend on the actions performed by the different networking devices including firewalls, IPSec gateways, and routers. ConfigChecker uses Binary Decision Diagrams (BDDs) to encode the global transition relation and provides a temporal language based on Computation Tree Logic (CTL) to run the reachability analysis among other applications. We have implemented a TCP interface and designed a simple communication protocol between the game engine and the model checker. The game engine builds the appropriate query based on player’s location and attributes and uses the TCP interface to collect the results.

We use XML to describe the hosts configuration. The root element host defines a host and its attributes. The child nodes represent a list of applications, services, vulnerabilities, and configuration settings represented as strings.

```
Listing 2: Host definition
<host id="IISServer" ip="10.10.10.10" asset="10k">  
  <app>W3svc</app>  
  <conf>scripting-enabled</conf>  
  <vul>CVE–2011–1894</vul>  
</host>
```

The example shown in listing 2 shows that the host IIS-Server is running the application w3svc and has the scripting setting enabled. The names of applications and configuration settings should be consistent in the hosts and exploits definitions.

C. World Generation

As stated before, The game world is a network of warehouses. New roads and warehouses are discovered as the player advances in the game and need to be dynamically allocated in the game world. We have developed an interface with the Lombardi Spring Embedder library [2] in which edges are represented as circular arcs between the nodes. The Lombardi drawing provides even distribution of edges around vertices. We believe that the even distribution of edges around the vertices along with the circular edges is closer to the real roads. Figure 5 shows an example of a graph drawn using regular drawing and based on Lombardi drawing style.

D. Game Engine

The game engine connects all the components together and interacts with the players. It handles players’ actions and communicates with the network configuration engine and drawing engine accordingly. We are in the process of developing a 3D game using Unity game development framework [22]. The engine will create a city that contains different warehouses where the player drives vehicles and trade items between the warehouses. Due to the expected long development phase for such a game, we have developed a simple graphical user interface as a windows application using C# .NET to illustrate the idea and run the preliminary evaluation.

VI. EVALUATION

We conducted a preliminary evaluation using a prototype developed to evaluate the functionality of the game. We designed two case studies to validate the ability of the game to discover attack scenarios and show the value of multiplayer collaboration. We omit the full details of the networks used in the case studies (applications deployed on the hosts and the exploits database) due to space considerations.

A. Case Study 1 (Information Leakage)

In this case study, we modeled the network shown in Figure 6 and asked eight of our lab members to play the game to accomplish two objectives. The objectives correspond to the following attack goals: starting from Attacker, (1) obtain the root privilege on SQL-Finance and Int-Web servers and (2) install a sniffer on the host Secretary or Emp-A.

The results show that three out of eight participants managed to accomplish all the objectives while the rest of them missed one. Figure 8 reports the number of steps and the total delays for each player. The delays depend on the game actions chosen by the players. We can clearly see the variation in the number of steps and delays between the participants. Moreover, the eight participants followed four different paths to accomplish the first objective and six different paths to accomplish the second. This supports our hypothesis that a large crowd will explore a large decision space as not all

![Regular and Lombardi drawing styles.](Image 5)
players will play rationally. Figure 9 reports the final scores and wealths of the players. The score affects the status of players in the game community, where the wealth determines their financial ability to progress in the game. The players choose whether they want to enrich their score or their wealth, which affects their decisions in the game. The results show that some of the participants chose one aspect over the other, while others kept a balance between them. This reflects different attacking behaviors followed by the participants.

B. Case Study 2 (Multiplayer Scenario)

In this case study, we show the value of multiple players playing at the same time. The players may, intentionally or unintentionally, collaborate to achieve goals that are impossible to accomplish in a single player setting. As shown in Figure 7, there are two possible starting points for attackers, namely Outside Attacker and Insider (shaded in red). The goal of the attack is to "Obtain root privilege at the Accounting database server". The happy day scenario for the outside attacker is as follows: (1) perform IIS-buffer-overflow exploit to obtain root privilege on IIS-Server, (2) run Client-scripting exploit to obtain a user shell on the HR machine, (3) perform LICQ-Remote-to-User exploit to obtain a user shell on the Accounting machine, and finally (4) conduct Local-setuid-buffer-overflow exploit that gives him root privilege on the Accounting machine. The Insider cannot achieve this goal alone since the network configuration does not give him broad access to the rest of the network.

Let us consider the scenario in which the setting HTML-Scripting is not enabled on the host HR. In this case, the happy day scenario will fail at step 2. Neither the Outside Attacker nor the Insider can accomplish their goals if they are working separately. However, if they collaborate together, they may accomplish the attack goal. Assume that the HR host is running a vulnerable service - ServiceX. And assume that the Insider can exploit this vulnerable service to enable the scripting settings while the outsider cannot. In this case, the insider can open the door for the outsider to continue on the scenario mentioned before. We ran this experiment in our lab and the players managed to accomplish the attack goal.

VII. RELATED WORK

Different approaches based on simulation and formal methods have been proposed for network security hardening. Attack graphs have been widely used for analyzing the inter-dependencies between vulnerabilities and security conditions, correlating intrusion alerts into attack scenarios, and automating the task of hardening a network against multiple intrusions [5], [20]. Static or formal analysis techniques were presented for identifying vulnerabilities and configuration conflicts and on-line debugging [4], [14]. However, formal approaches that can generate provable certificates such as SAT, SMT solvers, model checking, and theorem provers require an extensive search. Model checking techniques can suffer from state explosion in large scale problems. Due to this need for excessive resources, these problems go unsolved many times and many vulnerabilities go undiscovered. In this project, we are addressing the resource problem, and seek to show that a lack of resources should not translate into a lack of security.

On the other hand, researchers have investigated the value of employing games in various academic disciplines [19], [21], [23]. Control-Alt-Hack [11] is an interesting card game intended to increase the awareness of computer security needs and challenges and can improve the accuracy of peoples perception of computer security as a discipline and career choice. The game is targeting people with an affinity for computer science and engineering, but does not require significant computer security education or experience. CyberCIEGE [8] and CyberProtect [1] are two engaging video games that enhances computer security education by providing an environment for players to act as network administrators. Pipe Jam [13] is a puzzle game for software verification. The game provides the player with a set of ball-and-pipe puzzles span over multiple boards. The players goal is to ensure that the balls never get stuck in order to verify a security property.

Control-Alt-Hack and Pipe Jam games address problems different than ours. Control-Alt-Hack increases people’s awareness of security, but it is static and not specific for particular instances of security systems. Pipe Jam verifies a program against specific type of security properties. CyberCIEGE and CyberProtect are tailored toward computer and network security. However, they keep the player in the domain of network management and security as their objective is to enhance security education. The players need to have basic networking knowledge and should be interested in learning computer networking and security which limits the target audience. We aim to design a game that copes with larger crowd regardless of their educational background by moving the problem to a completely different domain.
Wealth (k)  
15  
25  
35  
0  
1  
2  
3  
4  
5  
6  
7  
8  

Score  (k)  
15  
25  
35  
0  
1  
2  
3  
4  
5  
6  
7  
8  

Participants 
Wealth Score

ONCLUSIONS

In this paper, we have presented a framework for utilizing the crowd intelligence in security hardening and attack attribution. We show that a proper mapping from network security domain to video games domain allows the players to imitate attackers' decision process. This helps in understanding attackers and identifying the weaknesses in network configuration. The results of our preliminary evaluation show that large crowd can explore large decision space and simulate different attack strategies.

We are now in the process of developing a full-fledged version of the game. To test large scale networks, the game needs to be published online to be accessible for large number of players. Proper motivations such as prizes may be offered and tournaments may be arranged to attract the players. We plan to study and deploy appropriate data anonymization techniques to avoid leaking any information about the network structure and configuration. Finally, one practical drawback for this framework is the assignment of costs and rewards to exploits. Although we have proposed a semi-automated approach to do that, we believe that it can be fully automated.

REFERENCES


