PenQuest: a gamified attacker/defender meta model for cyber security assessment and education

Robert Luh, Marlies Temper, Simon Tjoa, Sebastian Schrittwieser & Helge Janicke

Journal of Computer Virology and Hacking Techniques

e-ISSN 2263-8733

J Comput Virol Hack Tech
DOI 10.1007/s11416-019-00342-x

Journal of Computer Virology and Hacking Techniques
https://doi.org/10.1007/s11416-019-00342-x

ORIGINAL PAPER

PenQuest: a gamified attacker/defender meta model for cyber security assessment and education

Robert Luh[1,2,3] · Marlies Temper[2] · Simon Tjoa[1,2] · Sebastian Schrittwieser[1,2] · Helge Janicke[3]

Received: 10 July 2018 / Accepted: 11 November 2019

Abstract
Attacks on IT systems are a rising threat against the confidentiality, integrity, and availability of critical information and infrastructures. At the same time, the complex interplay of attack techniques and possible countermeasures makes it difficult to appropriately plan, implement, and evaluate an organization's defense. More often than not, the worlds of technical threats and organizational controls remain disjunct. In this article, we introduce PenQuest, a meta model designed to present a complete view on information system attacks and their mitigation while providing a tool for both semantic data enrichment and security education. PenQuest simulates time-enabled attacker/defender behavior as part of a dynamic, imperfect information multi-player game that derives significant parts of its ruleset from established information security sources such as STIX, CAPEC, CVE/CWE and NIST SP 800-53. Attack patterns, vulnerabilities, and mitigating controls are mapped to counterpart strategies and concrete actions through practical, data-centric mechanisms. The gamified model considers and defines a wide range of actors, assets, and actions, thereby enabling the assessment of cyber risks while giving technical experts the opportunity to explore specific attack scenarios in the context of an abstracted IT infrastructure. We implemented PenQuest as a physical serious game prototype and successfully tested it in a higher education environment. Additional expert interviews helped evaluate the model's applicability to information security scenarios.

Keywords Security model · Serious game · Education · Awareness · Game theory · Attack patterns · Controls · Malware · Intrusion detection

1 Introduction

IT systems are threatened by an ever-growing number of cyber-attacks. With the emergence of Advanced Persistent Threats (APTs), the focus shifted from off-the-shelf malware to multipartite attacks that are tailored to specific organizations or systems. These targeted threats are driven by varying motivations, such as espionage or sabotage, and often cause significantly more damage [55].

Today's cyber-attacks are typically conducted by dedicated groups within organized crime, industry, or nation state intelligence and increasingly affect less prominent targets as well. In 2013 alone, "economic espionage and theft of trade secrets cost the American economy more than $19 billion" [39]. 60% of espionage attacks threaten small and medium businesses whereas each reported data breach exposes over a million identities on average [54]. In the 2017 Official Annual Cybercrime Report by Cybersecurity Ventures [38], analysts speak of an estimated $6 trillion of annual damage that will be caused by cyber-attacks by 2021. The financial, utilities, and energy sectors find themselves in the crosshairs most often [42].

While APTs utilize software and techniques similar to more conventional attacks, their level of complexity and

✉ Robert Luh
robert.luh@fhstp.ac.at

Marlies Temper
marlies.temper@fhstp.ac.at

Simon Tjoa
simon.tjoa@fhstp.ac.at

Sebastian Schrittwieser
sebastian.schrittwieser@fhstp.ac.at

Helge Janicke
heljanic@dmu.ac.uk

1 Josef Ressel Center TARGET, St. Pölten University of Applied Sciences, St. Pölten, Austria

2 Institute of IT Security Research, St. Pölten University of Applied Sciences, St. Pölten, Austria

3 De Montfort University, Leicester, UK

sophistication is usually significantly higher. In general, it is becoming increasingly difficult to plan and implement an organization's defense in response to the wide range of possible threats originating in the digital domain. Detection and mitigation systems usually offer little in terms of contextual interpretation, which would help to better understand attacker motivations and objectives [26]. At the same time, attack pattern lists, vulnerability databases, and mitigation control catalogs often provide topically constrained information that can rarely be correlated, thanks to varying levels of granularity or abstraction. Most importantly, few models provide a means to map concrete system events as seen on affected assets to a semantic interpretation, known system flaw, or definitive countermeasure. Time and system interdependencies are rarely considered. In short, the increasing complexity of cyber-attacks makes it vital to explore novel approaches to attack/defense modeling, threat intelligence, knowledge extraction, and malicious activity detection on multiple layers, while preserving the flexibility needed to encompass new trends and scenarios.

Tailored awareness programs [50,56] and workable risk assessment procedures [4,23] have been identified as key components in any successful defense strategy. One approach to combine attack semantics with possible countermeasures and in-depth threat intelligence is the development of so-called 'serious games' [25], a topic that was first broached in the late 17th century as a feasible approach to education in general. Research [9,14,44] and security standards/guidelines such as the IT Grundschutz catalog [5] of the German Federal Office for Information Security emphasize that such games are well-suited to teach information security and awareness principles to an audience of differing IT background.

PenQuest, the APT roleplaying game (RPG) introduced in this paper, is a serious game based on a multi-tiered attacker/defender model that uses gamification, i.e. "game design metaphors to create (a) more game-like and engaging experience" [29]. Players can utilize the RPG to learn in an entertaining way about digital threats and how they may affect different infrastructures. PenQuest also provides an abstraction layer that helps assess the risk of targeted attacks on IT systems by providing an extensible framework for simulating attacker and defender behavior in an adversarial setting. Shortcomings in current defense implementations are inadvertently highlighted during a game session. Managers are encouraged to explore new threats and are presented with possible organizational and system-level countermeasures as suggested by accepted security standards. Analysts and IT/security experts are provided with the means to depict concrete hacks through attack patterns and cyber-observables and link them to the output of intrusion detection systems currently in place at their organization. We use existing repositories and languages for many of PenQuest's inherent concepts to be in line with common industry practices and to enable the future development of interfaces between our meta model and existing platforms. Specific mappings include the Common Attack Pattern Enumeration and Classification (CAPEC) schema [32] for describing attacks, various actor and asset elements from the Structured Threat Information eXpression (STIX) language [35], exploit properties extracted from the Common Vulnerabilities and Exposures (CVE) database [33], and security controls for (federal) information systems & organizations from NIST Special Publication (SP) 800-53 [17].

Altogether, the gamified model introduced in this paper helps to raise awareness and supports risk assessment procedures while closing the gap between high-level kill chains, independently published countermeasures and policies, and concrete attacks depicted by real-word events. This versatility turns PenQuest into the solid foundation for an in-depth risk assessment tool as well as framework for simulating attacks based on real-world threat intelligence – in addition to supporting awareness education in higher education and corporate environments. Specifically, this work contributes by:

- presenting an easily expandable, time-enabled attacker/defender meta model for depicting and assessing advanced persistent threats;
- developing a set of dynamic, non-cooperative roleplaying game (RPG) rules representing APT campaigns with all their assets, actors, and actions;
- providing a link between various standards and formats, such as STIX-defined data observables, CAPEC attack patterns, as well as operational risk assessment and mitigation planning within a gamified setting;
- introducing a mapping mechanism for correlating attacker behavior to opposing security and privacy controls listed in the NIST SP 800-53 standard;
- presenting and evaluating a physical game prototype ready for deployment in higher education and awareness training;
- paving the way towards automated attacker and defender strategy inference as well as threat simulation.

The remainder of this paper is structured as follows: After exploring related work in both the intrusion modeling and game (theory) domains (Sect. 2), we discuss the theoretical aspects of our model and present definitions, a high-level view on the multi-layered approach employed, as well as core mechanics (Sect. 3). In Sect. 4, the rule system of the gamified model is presented in detail, ranging from actor definition to assets, countermeasures and vulnerabilities to the actual game phases, as well as offensive and defensive actions. Concrete examples are used to describe the interplay of attack patterns and controls. *Italicized* segments present a cohesive example of applied gameplay across all subsections. Section 5 specifies our practical approach to combining STIX vocabularies, CAPEC patterns, CWE and CVE entries, as well as NIST SP 800-53 controls into one model, thereby uniting existing data sources across the threat information domain. Our first physical prototype design is evaluated in Sect. 6. Features and limitations of the model are discussed in Sect. 7. Section 8 concludes the article and provides a research outlook. Acronyms and some of the more specific mappings and definitions can be found in the Appendix.

2 Related work

Since PenQuest incorporates concepts from attack modeling as well as serious gaming, we take a closer look at similar works in both areas.

2.1 Threat modeling

Several researchers have introduced attacker/defender models that consider numerous factors and properties. In the following, we discuss the most influential works for the development of PenQuest. Like our solution, the Diamond Model of Intrusion Analysis [6] establishes the basic elements of generic intrusion activity, called an event, which is composed of four core features: adversary, infrastructure, capability, and victim. It extends events with a confidence score that can be used to track the reliability of the data source or a specific event. While some of its premises are similar to our own work, the Diamond Model does not consider Enablers or Disablers (see Sect. 3.2) or any automation for determining specific actions on the attacker's and defender's side. While it is a powerful template in its own regard, PenQuest provides these mechanisms – and more. Its gamified core is synonymous to a ready-to-use framework for simulation and automated knowledge discovery. In summary, the Diamond Model and PenQuest share commonalities and could potentially benefit from each other in terms of feature modeling and terminology.

In the work by Syed et al. [53], the authors present a unified cyber security ontology (UCO) extending the Intrusion Detection System ontology by Undercoffer et al. [57]. UCO is a semantic version of STIX with a link to cyber security standards similar to the ones used in PenQuest. Real-world knowledge is appended using featured Google searches (Google knowledge graph) and various knowledge bases. Syed et al. provide little information about data retrieval mechanisms and general automation. The main use cases emphasized are the identification of similar software and the association of vulnerabilities with certain (classes of) products. Unlike PenQuest, UCO does not consider temporal information or measurements of uncertainty.

Most other works in this area do not attempt to model attack–defense dynamics in a holistic manner while maintaining the link to concrete on-system actions. Existing models usually focus on specific threats or information security aspects [37,48], formally depict attack–defense trees [21,45], or describe ontologies with different topical focus [12,58]. Refer to the surveys by Mavroeidis and Bromander [31] and Luh et al. [26] for additional related work on threat models, ontologies, and languages.

2.2 Game theory and serious games

On the side of game-theoretic approaches, models are usually separated into several classes, depending on a multitude of factors. Roy et al. [46] survey a number of approaches for the network security domain and categorize non-cooperative games into static and dynamic scenarios of varying levels of information quality and completeness. As the work provides a good starting point for literature review, we classified PenQuest in accordance to Roy et al.'s schema (see Sect. 3.3).

Cook et al. [8] present an overview of risk assessment strategies for industrial control systems (ICS) that includes, among others, the game theoretic approach. The authors conclude that there exists no unified risk model for hitherto unconsidered ICS scenarios that incorporates events, threats, vulnerabilities, and general consequences with a measure of uncertainty. Suggested future work includes the development of an environment to allow an intelligent adversary to test an ICS's defenses in a nondestructive (e.g. game theoretic) manner. This is specifically addressed by PenQuest. Similarly, Lewis [24] offers an introduction to risk assessment strategies for critical infrastructure protection, including an in-depth look at Bayesian belief networks and game theory applications.

Addressing a power grid scenario, Holmgren et al. [15] introduce a model for studying defense strategies against hostile actions leading to a power shutdown. Outages are split into stages ranging from prevention to recovery. Unlike the more flexible PenQuest, Holmgren et al.'s model is a perfect and complete information game that does not allow adding new components to the existing network of assets. In addition, only qualified attackers of static skill, determination, and existing access to the system are considered. The overall motivation of implementing the power grid game is still comparable to our approach: It aims to help with resource planning, risk screening, and with studying generic mechanisms that enhance the overall understanding of attacks against a specific system.

Contrary to the above, Nguyen et al. [41] formally describe an abstract zero-sum, incomplete information game for network security scenarios. Their network model is based on the concept of linear influence networks [36], which is represented by two weighted directed graphs that signify the relationship of security assets as well as denoting vulnerability correlation among the nodes. The resulting influence matrix describes the contribution of an asset to overall security. If an asset is taken down by the attacker, its node is removed from the network, lowering the security rating for all connected entities. For common assets, compromising one node changes the probability that another linked asset will come under attack. Unlike PenQuest, Nguyen et al. do not consider specific actors, assets, scenarios, or data sources. Since compromised nodes are removed entirely from the network, the model does not directly support stepped attacks or semi-successful attacker actions. As a pure zero-sum game, it also does not offer the same level of flexible payoff as the system introduced in this paper.

In the area of serious games, Shostack [51] present Elevation of Privilege, a game designed to motivate people to engage in threat modeling. It is based on the STRIDE mnemonic [20] and includes the Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege attack categories. Defense actions and further threat specifics supported by our RPG are not part of the game.

Another example is Operation Digital Chameleon [43], a red-team exercise turned into a board game. Unlike PenQuest, the game asks of the players to build an attack and defense strategy without the guidance of rules. A game master is responsible for assessing both teams' solutions and encourages discussion. While this offers flexibility and is suitable for dedicated workshops comprising large groups of participants, Operation Digital Chameleon does not provide a model for APT representation and data mapping. With its formal approach, PenQuest paves the way for future scenario computation and automated mitigation planning.

More specialized learning solutions include What.Hack [59], a game revolving around phishing defense, an approach to generate social engineering awareness [2], and a game called OWASP Cornucopia,[1] intended for use in software development.

On the side of light entertainment, several vendors have released strategy or card games centered around hacking. Examples include d0x3d,[2] Control-Alt-Hack,[3] as well as Hackers and Agents.[4] Neither of these games attempt to model overly realistic infrastructures or provide a holistic view on the topic of information security. With PenQuest, we seek to remedy these shortcomings and provide a gamified model that offers realistic scenarios while still being classified as edutainment.

[1] https://www.owasp.org/index.php/OWASP_Cornucopia.

[2] https://www.thegamecrafter.com/games/-d0x3d-.

[3] http://www.controlalthack.com/.

[4] http://www.hackersandagents.com.

3 Attacker/defender model

Our gamified approach is based on a novel attacker/defender model tailored to depict interconnected services that maintain – and operate with – various types of information. Consuming and providing services are represented by parent and child entities that enable the modeling of arbitrary, interdependent systems and infrastructures. At the same time, PenQuest incorporates on-system events that define nominal and anomalous behavior, thereby establishing a link to actual attacker behavior in the form of individual actions performed.

In this section, we take a closer look at the foundational components of PenQuest. This includes numerous definitions and the axiomatic interplay between elements: The *base model* defines information and events in the context of the game and outlines how an infrastructure of services can be modeled. The *game model* provides the classification and definition of an "action" as performed by an actor – with all its properties and requirements. Lastly, the *rule model* introduces the game-theoretic properties of the RPG. The specific rules describing how the game is played are introduced thereafter, in Sect. 4.

3.1 Base model

The PenQuest base model consists of three main layers – service, information, and event – that are influenced by attacker and defender actions. Figure 1 provides an overview of the interplay of the three main layers and sketches the link to the PenQuest game model. In the following, we present the components that describe the cornerstones of our approach.

3.1.1 Service layer

In most modeling scenarios, services are synonymous to assets a defender seeks to protect. They will most typically be IT systems maintained by an organization but might also represent a single process running on a computer system.

In the model, a service has 4 properties: Type, status, exposure, and ownership. The *type* of a service can be understood as combination of a unique designation and the description of its purpose (functionality). In many cases, the type property will simply contain the name of the service for reference.

Status describes the current level of confidentiality, integrity, and availability of a service. It is the measure of a service's ability to provide its functionality to a consumer, be that another service or human user. In this context, *confidentiality* [52] describes the preservation of secrecy of information associated with the service. If confidentiality is compromised, the respective knowledge or configuration (see Sect. 3.1.2 below) becomes known to the attacker. *Integrity*, on the other hand, maintains the immutability of information belonging to the service. If integrity is lost, entities can no

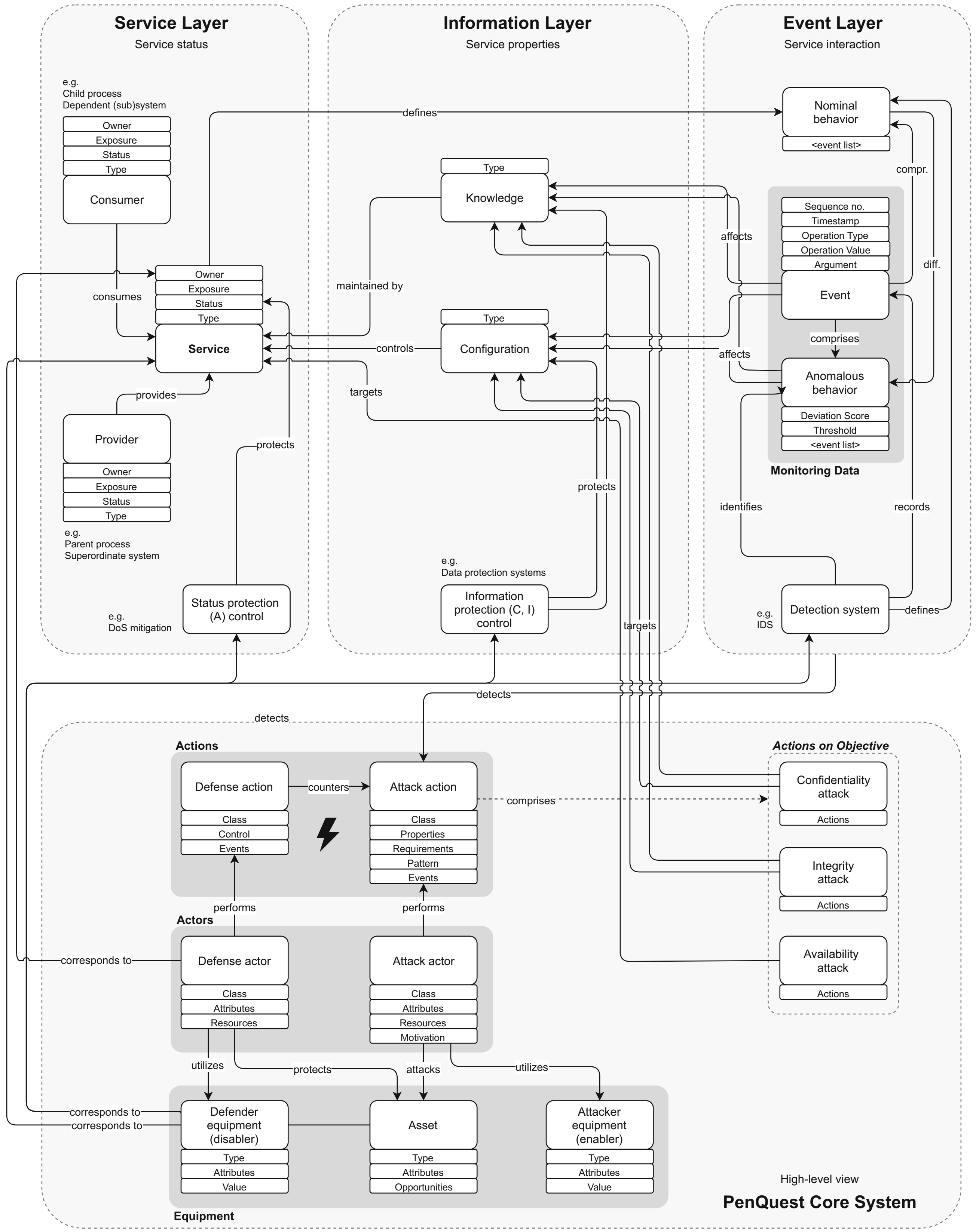


Fig. 1 High-level view on the PenQuest meta model. The top half represents the layers of the base attacker/defender model while the lower portion depicts the gamified rule system, which is discussed in detail below

longer trust in the information being correct. Lastly, *availability* denotes the prevention of system failure or degradation of service functionality. If availability is compromised, a service can no longer be used in an unimpeded fashion.

Exposure determines the placement, and by extension, the exposure of a service to attacks originating outside the infrastructure. Exposed services can be accessed directly, while internal services can only be attacked once an exposed service in the path of attack has been compromised. This principle generally represents the vector used by an adversary to gain access to a specific system.

The *owner* of a service is synonymous to the defending party in our attack/defense model. Owners can be organizations, physical and legal persons, or even abstract entities. In the game rules, the owner is often referred to as Player 2, or Bob.

Services can be reliant on other services for maintaining their functionality. Superordinate services are called *providers*, while *consumers* are in turn dependent on the service in question. The loss of confidentiality, integrity, or availability of the provider may in turn lead to the loss of confidentiality, integrity, or availability of the consumer. More information about service dependencies can be found in the game rules in Sect. 4.3.

3.1.2 Information layer

PenQuest distinguishes two types of information: Knowledge and configuration. *Knowledge* represents potentially sensitive information relevant to the entity owning and/or using the service. Services are often designed to safeguard knowledge – be that that their main purpose or only implied functionality. Attacks against the confidentiality status of a service seek to access restricted knowledge against the owner's will, while integrity attacks aim to (covertly) change that information.

Configuration denotes the technical or organizational settings and policies controlling a service's functionality as well as its level of exposure. It influences how a service performs its work. Confidentiality attacks against a service's configuration will lead to the disclosure of settings and associated service properties (type, status, exposure). A loss of configuration integrity, on the other hand, might introduce undesired functionality and can change a service's exposure level.

In short, a service is controlled by configuration while maintaining knowledge. Services and information are protected by controls (technical or organizational measures aiming to protect the service from adversary activity) that focus on safeguarding confidentiality, integrity, and availability.

3.1.3 Event layer

Events are the link between the base attacker/defender model, the PenQuest RPG game/rule system, and actual real-world monitoring data. They formalize activity that leads up to service compromise and describe what an attacking actor is actually doing. Service operation itself is described by events as well, providing a baseline of nominal behavior that can be used for anomaly detection.

Events have a number of properties, including arguments, type, value, temporal information, and an incrementing number in a potential sequence of events. In combination, events can be used to describe a wide range of happenings, starting from undesired operating system (OS) process behavior to more high-level activity influencing an entire (IT) system.

For example, a user-land OS process might trigger the event `E(sequence, operation, argument)` = `(1, create-file, dropped.exe)`, followed by the event `(2, start-process, dropped.exe)`, which results in an IDS anomaly *A* with a specific value denoting its deviation from a set baseline: (`A(deviation, threshold) = (22.4, 10)`). This occurrence would affect the confidentiality of knowledge that is associated to the modeled service of e.g. `browser.exe`, thereby changing the confidentiality status of the service to 'compromised'.

On a less granular IT service level, a generic event `E=(1, change-configuration, htaccess)` caused by the purposeful alteration of a web server's security settings would affect the configuration on an integrity level, changing the service's integrity status to 'compromised' and altering the information controlling the service in the process.

Since we do not only want to sketch such simple attack scenarios but also bridge the gap between actors, assets, vulnerabilities, concrete attack patterns, events, and countermeasures, we extended the model with formally defined *actions* (see Sect. 3.2) which, in turn, shape the foundation for the PenQuest ruleset introduced in Sect. 4 and beyond.

3.2 Game model

In this section, we discuss the game model that is superimposed onto the base model by means of various definitions and, ultimately, the game rules themselves.

As a model that adopts several concepts from the genres of role playing and strategy games, PenQuest contains a few base building blocks that need to be understood before going into detail: The sides of the attacker/defender contest are represented by *actors*, or characters, that have a certain class. *Classes* describe the background and affiliation of the actor and come with certain properties such as skill, motivation, and monetary resources, which, in turn, are named attributes. *Attributes* are the main determining factor when it comes to defining an actor's 'strength'. They can change under certain circumstances and are modified through equipment. *Equipment* comprises physical appliances, tools, and more general policies – everything that will provide bonuses or penalties to one side or another. Such *modifiers* typically directly or indirectly influence the chance of success of actions performed in the game. *Actions* are the bread and butter of PenQuest and describe what each actor is actually doing to attack or defend the infrastructure of services that is emulated by the scenario. Actions often come with specific *requirements* in terms of attributes, which reflects the difficulty of the actor's endeavour. The final decision whether an action succeeds usually entails a random factor which is resolved by 'rolling the dice', i.e. computing the outcome according to the modified probability of success.

In the following, we discuss the definitions that are make up above concepts. Depicted in the lower half of Fig. 1, PenQuest differentiates three main elements: actions, actors, and equipment. The glue between is provided by meta information in the form of categorization, various action requirements, and the annotation of events as well as (third party) attack patterns that are used to populate the model.

3.2.1 Actions

Actions are at the core of the model and tie together all the other components. They also link real-world service and actor behavior to concrete data points such as observable attack patterns or event sequences. Formally, an action X is defined as n-tuple of typical length $n = 11$, whereas the model's flexibility allows for the omission of unneeded elements. Simply put, an action is performed by an actor and is further enabled or disabled by various types of equipment. It is assigned a category within the model, which includes usage requirements, properties pertaining to its success, and a detection chance. Attack patterns and associated events tie it to real-world data points.

$$\begin{aligned}
X = \langle\ & \\
& \langle AttackActor\langle Class, Motivation, Attributes, \\
& \quad Resources\rangle\rangle, \\
& \langle DefenseActor\langle Class, Attributes, Resources\rangle\rangle, \\
& \langle Enabler\langle Type, Effect, Attributes, Name\rangle\rangle, \\
& \langle Disabler\langle Type, Effect, Attributes, Name\rangle\rangle, \\
& \langle Victim\langle Type, Name, Exposure, Parent, \\
& \quad VectorParent, Configuration, \\
& \quad Knowledge, Status, Integrity\rangle\rangle \\
& \langle AttackClass\langle Stage, PatternClass, Mode\rangle\rangle, \\
& \langle DefenseClass\langle Category, ControlClass, \\
& \quad ActionClass\rangle\rangle, \\
& \langle Requirements\langle *Actor\langle Attributes, Resources\rangle, \\
& \quad Victim\langle Exposure, Integrity\rangle\rangle, \\
& \langle Properties\langle Sophistication, SuccessChance, \\
& \quad DetectionChance\rangle\rangle, \\
& \langle AttackPattern\langle Impact, ID\rangle\rangle, \\
& \langle Event\langle Type, Time, ScoreSequence, Parent, \\
& \quad Operation, Argument\rangle\rangle\rangle
\end{aligned}$$

Below, we briefly introduce the sub-classes of action X. A full definition and a more in-depth explanation can be found in Appendix B.

3.2.2 Actors

Actors are the players of PenQuest. The defending actor (service owner) tries to fend off an attacker while his or her adversary seeks to compromise the status of a targeted service. Our model differentiates two actor classes: The *AttackActor* and the *DefenseActor*, which correspond to the actor types introduced in the `ThreatActorType` vocabulary schema of the STIX threat information language [1]. Each actor is described as a unique class with their own motivation, primary attributes that represent an actor's skill, motivation, and financial resources, as well as operational resources such as time and knowledge about the opponent.

The *AttackActor* has a range of possible motivations. These goals represent the overall objective of the attack and provide additional context for determining attacker attributes and skills. They correspond to the STIX Threat Actor `Motivation` vocabulary and encompass various ideological goals. The actor creation routine of PenQuest provides percentages defining likely actor/goal combinations (see Sect. 4).

3.2.3 Equipment

There are two types of equipment in PenQuest. So-called *Enablers* represent attack tools and vulnerabilities employed by the aggressor. They have specific effects on the target service and come with a wide range of properties (e.g. level of impact or maturity of a vulnerability-type Enabler (see CVE/CVSS mapping in Sect. 5)) and prerequisites (e.g. privileges and level of user interaction required) that need to be considered.

Defenders use *Disablers* to thwart their adversary. Such defender equipment encompasses assets to be protected and various appliances that prevent, detect, delay, or generally hinder an attack. The respective effects aim to reverse or mitigate the damage caused by attacker equipment and actions. Specific examples can be found in the game rules in Sect. 4.2.

Lastly, the *Victim* class describes the system that is targeted by the attacker, typically a specific service (i.e. asset). Victim information differentiates between internal and exposed systems – which can be accessed directly from within the DMZ – and identifies service status, integrity, and systemic as well as attack vector parents that represent consumers, providers, and systems that have to be compromised before the victim can be attacked directly. In each scenario, the attacker attempts to achieve his or her goal by compromising a victim (target) in a specific fashion. This translates to the attempted compromise of one the defender's assets in accordance to the CIA triangle of information security [52]: Confidentiality attacks (theft of information), integrity attacks (altering of information), and availability attacks (service status changes), which are all part of the *AttackClass* meta component introduced below.

3.2.4 Meta information

Each action is assigned an *AttackClass* by the model. Specifically, this class contains the categorization into the APT kill chain stage [16,27], and discriminates various types of attack pattern. These types correspond to the mapping of granular actions to CAPEC attack patterns [32], which is discussed in detail in Sect. 4.5.2. *AttackClass* information helps to abstract specific attack actions and provides a means to link adversary behavior to possible defense measures.

The corresponding *DefenseClass* contains our abstraction of NIST SP 800-53 controls [17], which we use to link attack patterns to specific technical and organizational countermeasures. See Sect. 5 for more information.

Requirements identify the minimum actor attributes and resources needed to conduct the attack, as well as other prerequisites in the form of system exposure. Requirements are optional and depend on the complexity of the respective attack, which is defined through its *Properties*, namely the minimum required actor skill and the chance of success as well as detection. This information is largely gleaned from CAPEC and CVE.

Each action corresponds to an observable *AttackPattern*, which is identified by an ID and its impact on the CIA triad. Attack patterns link the modeled action to specific hostile activity as described in the CAPEC schema. Ultimately, attack patterns are mapped to a set of monitored *Events* (cyber observables) with specific underlying operations and arguments, triggers (parents), timestamps, anomaly scores, and sequence numbers. The modeling of unique events closes the gap to the data layer and allows us to lower the level of abstraction to the actual systemic representation. Refer to Sect. 5 for a mapping example.

Refer to Appendix B for more information about the classes inherent to the game model. In the following, we discuss PenQuest's rule model, i.e. its game theoretic background and core mechanisms.

3.3 Rule model

The definitions discussed above provide the distinct elements that are part of the model and introduce relationships and concrete class definitions. Before these components can be assembled into a playable game, we need to outline the game-theoretic principles of PenQuest as well as its core mechanics.

3.3.1 Game principles

Instead of relying solely on decision models, adversarial behavior can be expressed in the form of game situations [24]. Techniques associated with game theory, which is defined as *"the study of models of conflict and cooperation between rational decision-makers"* [40], can be used to model multi-player scenarios that allow participants to pursue goals and rewards by acting in the most favorable, risk-adverse, or cost-effective way possible. The outcome of a game presents the actors with a strategy for resource allocation, risk minimization, or a general approach to meeting a certain objective.

Even though PenQuest is a model/game hybrid, it can be categorized according to several principles of game theory, as summarized by Roy et al. [46]:

- **Non-cooperative nonzero-sum game**: Opposition between players is an integral part of the design: Player 1 (attacker) always combats Player 2 (defender) and tries to achieve adverse goals by stealing information, manipulating the integrity of data or systems, or by shutting them down entirely. Even though actions are not typically assigned points that are symmetrically gained/lost (making it nonzero-sum), it can be argued that the mechanism of asset compromise is in fact a zero-sum game, where the defender loses points describing integrity and status, while the attacker gains a corresponding advantage. In other situations, win/loss is represented by an increase or decrease of attributes or action success and detection chance. These bonuses are one-sided, yet always shift the balance between the players away from equilibrium.
- **Asymmetric strategy**: The strategy sets of the two players are not identical – the attacker draws from a different pool of actions than the defender. This stems from the difference in goal and purpose: Attackers will attempt to penetrate a system using malware or by exploiting vulnerabilities, while a defender tries to counter these actions by implementing technical and organizational controls.
- **Dynamic/extensive game with static elements** [46]: While the game uses sequential moves characteristic for dynamic games, the second player typically remains unaware of the first player's actions, making the model bear some resemblance to a strategic setting where players act simultaneously and in secret. At its core, PenQuest remains dynamic – emphasized by its multi-stage nature.

- **Imperfect, incomplete information**: As stated above, Player 1 does not necessarily know the moves previously made by the attacker, and vice versa. It is in fact vital to players' success that performed actions remain secret, thereby potentially causing the other party to make imperfect decisions. At the same time, the general set of strategies is known to both sides. The exact payoff in a certain situation, however, is not, due to the lack of information about past activity and their impact on success and detection chances (incomplete information).
- **Bayesian formulation of static elements**: In PenQuest, players have incomplete information on the other players, especially when it comes to actions and strategies, which are derived from the attacker's type and ultimate objective. There is, however, a fixed probability that players, being one of n available classes, need to conduct/defend against one of three kinds of attacks on a finite set of assets in order to win the game.
- **Finite & discrete**: While some action combinations are continuous in nature, the general action/reaction game follows a discontinuous sequence. The number of game turns is limited by an exhaustible resource – Initiative (i.e. time efficiency).

Following Roy et al.'s taxonomy [46], our game can be classified as non-cooperative, dynamic game with imperfect and incomplete information, that draws from static elements of Bayesian formulation. According to You and Shiyong [60], our two-player hybrid zero-sum game G is defined by the triplet:

$G = (A, D, F)$, where:

- A is a set of strategies of Player 1 (attacker),
- D is a set of Player 2 (defender) strategies, and
- L is, for select game principles (see below), a real-valued function $A \times D$. Therefore, $L(a, d)$ is a real number for every $a \in A$ and $d \in D$.

3.3.2 Core mechanics

The core mechanics of the game include the model's sole zero-sum component describing target compromise and a full attack–response mechanism for achieving hostile or defensive goals. The central part of the model is synonymous to the process of picking a $\langle Victim \rangle$ service and executing an attack of a certain $\langle AttackClass \rangle$ corresponding to the CIA triangle mentioned above, followed by a matching defensive response. We have modeled this basic building block as a Workflow net (see Fig. 2) to identify inconsistencies in the model. A workflow net (WF-net) is a strongly connected Petri net (PN) with two unique input (source) and output (sink) places as well as a reset transition r.

Following the notation of Esparza et al. [11], a Petri net can be defined as a 5-tuple $N = (P, T, F, W, m0)$ where P is a set of places or states, T is a set of transitions with $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation, $W : (P \times T) \cup (T \times P) \rightarrow N$ is a weight function satisfying $W(x, y) > 0 \iff (x, y) \in F$, and $m0 : P \rightarrow N$ is a mapping called the initial marking. In our case, $m0$ equals the initial state where a victim has not yet been picked (called "Start"). A reset transition r leads back to the beginning of the attack process and repeats until the victim has been successfully compromised.

Specifically, an attacker picks a $\langle Victim \rangle$ that is either exposed or in-vector (see Sect. 3.1). Subsequently, a mode of attack (C, I, or A) is chosen. As depicted in Fig. 2 the initial impact of such an attack is determined based on the $\langle Enabler \rangle$ resource and various other properties and attributes. Independently from the level of success of the action, the defender first attempts to detect the activity. In the Petri net, this is again modeled as exhaustible resource that corresponds to the $\langle Disabler \rangle$ class, where detection systems (Det) can be found. If an attack was detected, the defending entity can attempt to counter the initial impact by utilizing further $\langle Disablers \rangle$. Ultimately, initial impact and the degree of success in reducing that impact is resolved in PenQuest's only zero-sum game element:

Here, Player 1 chooses action $a \in A$ targeted at victim $v \in V$ ($\langle Victim \rangle$ in our class structure), which is kept secret unless Player 2 meets the detection requirements (see Fig. 3). If action a_v is conducted successfully, Player 1 gains a number of points $L(a_v)$ representing the level of victim compromise, which are directly or indirectly deducted from the respective defender's tally (payoff). Independently from the outcome of the detection attempt, Player 2 chooses $d \in D$ for victim $v \in V$, attempting to counter the (assumed) action a_v. Points $L(d_v)$ are computed, fully negating $L(a_v)$ in the best of cases.

This game principle applies to victim system integrity and status (see Fig. 3 as well as the $\langle Victim \rangle$ definition in Sect. 3.2), the success chance of hostile attacks and defensive actions (see $\langle Properties \rangle$ and $\langle Enablers \rangle$/$\langle Disablers \rangle$ below), and some other attribute and resource bonuses and penalties discussed in the next section.

We have again modeled this principle as WF-net, describing the process of generating $L(a_v)$ as well as $L(d_v)$, respectively. This second net, with a total of 33 places and 65 transitions, can be used for simulating arbitrary actions generating $L = 0..3$ on both the attacking and defending side. The unveiling of actions is again part of the parent WF-net discussed above.

It is important to bear in mind that the RPG's victory conditions are only indirectly affected by these shifts in L, and that an increased numeric distance from the equilibrium of factors other than victim system integrity and status does

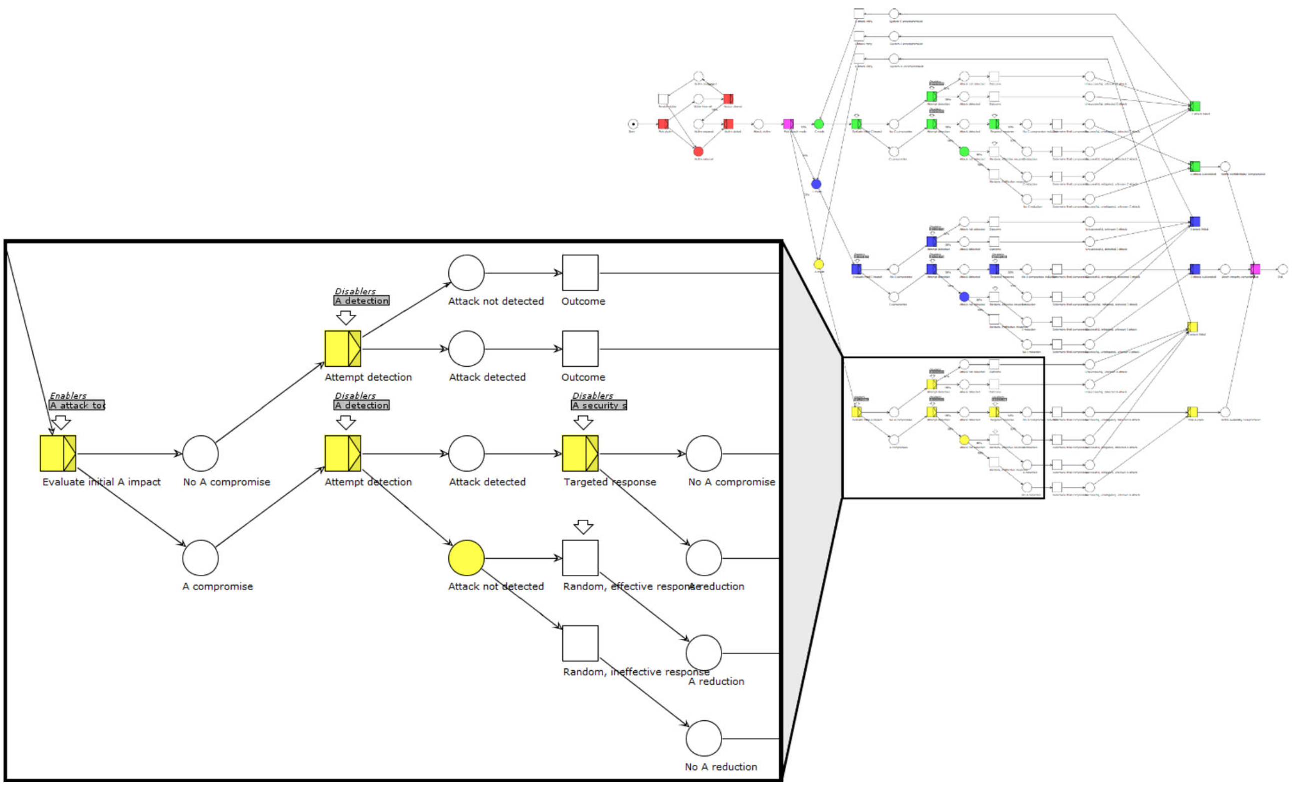


Fig. 2 WF-net describing the process of picking and attacking a victim. The net consists of 64 places, 51 transitions, and a total of 133 arcs. Soundness was determined by analyzing the net in WoPeD [13]

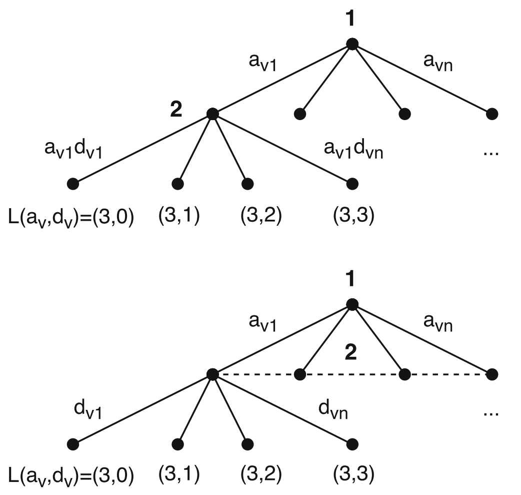


Fig. 3 Information set of the defending player for victim system compromise through attacker action a_{v1}, shown in extensive form [22]. In the above case, Player 2 successfully unveils a_{v1}, giving him the chance to specifically counteract the gain $L(a_v)$ of Player 1 by increasing his own score $L(d_v)$ through defense action d_{v1}. Below tree represents the limited information set for Player 2, when a_{v1} remains undetected

not always guarantee one player's domination over the other. In fact, victory is determined by the exhaustion of available (temporal) resources or the successful compromise of the chosen victim asset within an allotted time window.

In the following, we build upon these mechanics and construct a full-fledged strategy game for simulating real-world attacks and their possible countermeasures.

4 Game rules

In order to transform the model into a playable format while exemplifying both education and data enrichment purposes, we opted to cast the rules of PenQuest in the mold of a game manual. This approach is also owed to the fact that our system adheres to the principle rules of a roleplaying or board strategy game for two players that follows a set of stages to determine the course of a play-through. Conditional decisions are made by rolling dice, which is representative for outcomes that come with a certain chance of success as well as a random element.

In the following, we present the three main aspects of PenQuest, which build the foundation for our physical prototype. In the preliminary stage, dubbed character or *actor creation*, the players specify the two opponents, their basic attributes and skills, as well as attack/defense goals corresponding to specific objectives and motivations. Once the

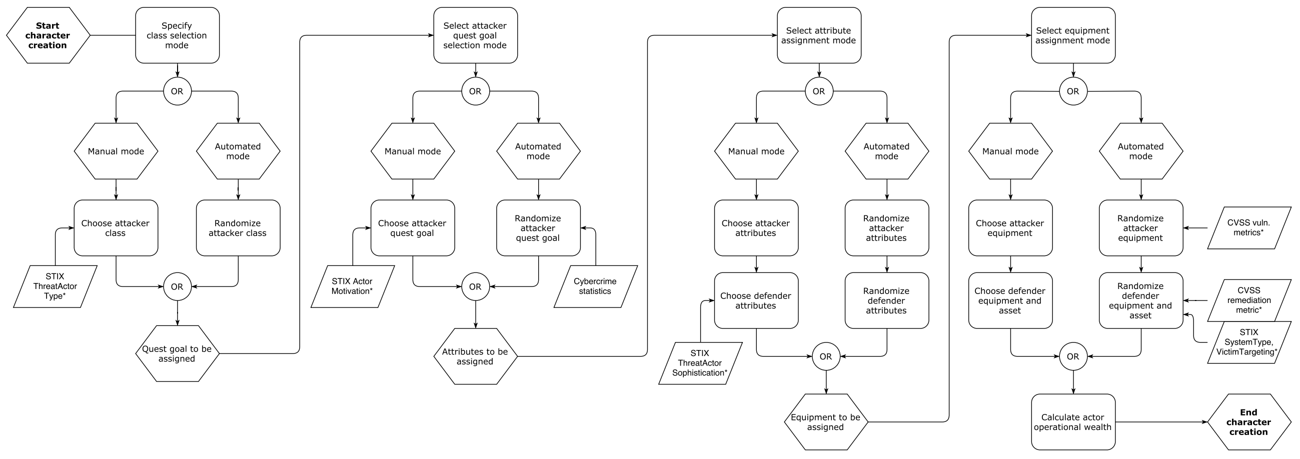

Start character creation
Specify class selection mode
OR
Manual mode
Automated mode
Choose attacker class
Randomize attacker class
STIX ThreatActor Type*
OR
Quest goal to be assigned
Select attacker quest goal selection mode
OR
Manual mode
Automated mode
Choose attacker quest goal
Randomize attacker quest goal
STIX Actor Motivation*
Cybercrime statistics
OR
Attributes to be assigned
Select attribute assignment mode
OR
Manual mode
Automated mode
Choose attacker attributes
Randomize attacker attributes
Choose defender attributes
Randomize defender attributes
STIX ThreatActor Sophistication*
OR
Equipment to be assigned
Select equipment assignment mode
OR
Manual mode
Automated mode
Choose attacker equipment
Randomize attacker equipment
CVSS vuln. metrics*
Choose defender equipment and asset
Randomize defender equipment and asset
CVSS remediation metric*
STIX SystemType, VictimTargeting*
OR
Calculate actor operational wealth
End character creation

Fig. 4 Actor creation process overview. Data imports marked with an asterisk (*) are used in both the respective manual and automated mode of actor creation

actors are defined, the game is ready to start. Specific activities within the boundaries of this system are represented by *conflict actions* that, combined into a full play-through, constitute an attack targeting a service asset. See below for detailed information on each of these core game rules.

4.1 Actor creation

Characters are the main actors involved into an attack/defense scenario. Since PenQuest is a very flexible game system, there are many possible combinations of opponents and scenarios. Instead of offering a fixed set of adversaries, we provide a full-fledged character creation system that supports both manual and automated actor design. Generally, it is important that the player portraying the attacker keeps all their properties secret – neither class, goal, attributes, nor the available equipment should be known to the defender. In contrast, the class of the defender is public knowledge. Defender attributes and equipment are initially kept secret, but may be uncovered during the reconnaissance phase (see Sect. 4.4). See Fig. 4 for a full overview of the character creation process.

4.1.1 Class and motivation

The respective player starts by picking in secret a *class* for the attacking side ($\langle AttackActor \rangle$). Classes follow the `ThreatActorType` vocabulary schema of STIX [1]. Third party eCrime actors such as money laundering services and malware developers are not modeled, since we want to focus on active factions that are likely to directly target the assets of a victim. The available classes (STIX name) are:

1. **Thief *TH*** (Cyber Espionage Operations): Cyber spies are entities aiming to steal information such as intellectual property or other private data (knowledge).
2. **Explorer *EX*** (Hacker, White hat): This class of hackers does not seek to cause damage but acts to improve current security measures by e.g. raising awareness.
3. **Rogue *RO*** (Hacker, Gray hat): Grey hat hackers sometimes violates regulations but usually lacks the malicious intent typical for a black hat hacker.
4. **Raider *RA*** (Hacker, Black hat): Black hats are malicious cyber-actors that do not operate within legal or moral boundaries.
5. **Crusader *CR*** (Hacktivist): These hackers assume the moral high ground and typically want to prove a point or leave an ideological statement.
6. **Operative *OP*** (State Actor/Agency): State actors can be agencies with various missions in the areas of (clandestine) intelligence or counter-intelligence.
7. **Infiltrator *IN*** (Insider threat): Insider threats are individuals of varying skill that seek to undermine their own employing organization.
8. **Protester *PR*** (Disgruntled customer/user): This class of actors represent a generally unsatisfied third party.

Each character class has a range of possible *motivations*. These represent the overall objective of the attack and provide additional context for determining attacker attributes and skills. Motivations correspond to the Threat Actor `Motivation` vocabulary that is part of STIX as well as the *Motivation* class of our attacker/defender model. Possible motivations $\langle AttackActor \langle Motivation \rangle \rangle$ are:

– Ideological (*id*): The actor acts out of their ideological belief in a cause, such as anti-corruption, anti-establishment, environmental, ethnic/nationalist, infor-

Table 1 Typical motivation of the various attacker classes

	TH	EX	RO	RA	CR	OP	IN	PR
Ideological	0–10	0–25	0–16	0–10	0–40	0–10	0–15	0–30
Ego	11–29	24–45	17–32	11–20	41–50	11–15	16–35	31–50
Financial	30–65	46–55	33–49	21–50	51–55	16–40	36–55	51–70
Military	66–70	56–60	50–66	51–65	56–60	41–65	56–60	71–75
Opportunistic	71–90	61–80	67–83	66–80	61–70	66–75	61–90	76–90
Political	91–100	81–100	84–100	81–100	71–100	76–100	91–100	91–100

The numbers represent the lower and upper bound of the range. Ideological sub-goals are currently not covered, but may be easily added for additional granularity

mation freedom, religious, security awareness, or human rights.
– Ego (*eg*): The attacker wants to prove a point to others or herself.
– Financial or Economic (*fi*): The attack is motivated by financial goals.
– Military (*mi*): The actors wants to achieve a military victory or gain an strategic/tactical advantage.
– Opportunistic (*op*): The attacker ceases an unexpected opportunity to strike against a target.
– Political (*po*): The adversary acts out of political motivation.

While a player can pick any motivation from the above list when manually creating a character, not every attacker is equally likely to have a certain motive. Disgruntled customers will not typically have political agendas, while black hat hackers are unlikely to be driven by an environmentalist ideology. The likelihood of combinations is modeled by the RPG rule system as percentage values denoting the likelihood of a character/motive mapping. Randomization is achieved by rolling two ten-sided dice (2D10) and then consulting the probability Table 1, whereas the first D10 determines the tens digit and the second D10 represents the unit position. Currently, these motivations are largely cosmetic and help to flesh out the actors. However, it is possible to expand the rule system by incorporating cyber-crime statistics that reflect typical motivations.

Armed with attacker class and motivation, we can specify the defending actor (defender class, $\langle DefenseActor\rangle$) of the hostile campaign by simply choosing from below list or by rolling another D8. The list of actors is inspired by STIX' `Victim Targeting by Sector`, which leverages the external CIQ (Customer Information Quality) standard published by OASIS.[5] However, there is no exhaustive `Industry Type` list provided by STIX. We therefore compiled our own list of target actor types:

1. **Company, Primary Sector *CP***: Private company operating in the primary sector (e.g. mining, farming, forestry).
2. **Company, Manufacturing *CM***: Company in the business of non-infrastructure manufacturing and processing.
3. **Company, Services *CS***: Company in the services sector.
4. **Infrastructure *IF***: Organization maintaining (critical) infrastructure.
5. **Military *MI***: (Para-)military organization with strategic or tactical focus.
6. **State Actor/Agency *SA***: State-sponsored organization with possible intelligence background.
7. **Education *ED***: Schools, universities and other organizations in the education sector.
8. **Private individual *PI***: Person without relevant external affiliation.

In the following and throughout this paper, we use an example scenario with the actors of Alice and Bob to explain PenQuest's rules. Alice personifies the attacker and Bob represents the defending actor.

Alice decides to use automated character creation to put together an APT scenario. She rolls an eight-sided die (D8) to determine attacker class and scores a '5', making her offensive actor a Crusader (hacktivist). When it comes to motivation, she rolls '6' and '5' on a D10. Consulting the probability table, she determines the attacker motivation to be Opportunistic, meaning that the Crusader simply chanced upon the occasion. Next, Alice rolls a '7' for the defender class. Her chance victim (Bob) seems to be in the education sector.

4.1.2 Attributes

PenQuest uses *attributes* to distinguish different levels of actor skill, motivation, and resources. These can be set manually for a custom game or determined randomly. Attributes have ratings that range from 1 to 5, whereas 1 corresponds to an undeveloped state and 5 to the highest possible maturation level. Attributes in PenQuest are *Sophistication*, *Determination*, and *Wealth*:

[5] https://stixproject.github.io/documentation/idioms/industry-sector/.

Table 2 Default attribute modifier table for all actors

	TH	EX	RO	RA	CR	OP	IN	PR	CP	CM	CS	IF	MI	SA	ED	PI
SO		+1				+1	−1			+1		+1		+1		
DE				+1	+2		+1	+1			−1	+1				+1
WE	+1	−1			−2	+1	−1		+1	+1	+2	+1			−1	−2
INI	−1		+1		+1	−1		+1		−1		−1	+1			+2
INS	+1	+1					+2	−1				−1			+2	

For custom gameplay or simulation scenarios with a reduced need for balancing, these values can be freely changed and should be based on the latest available studies

Sophistication ***SO*** – Measures an actor's capabilities, determines Initiative (number of actions during conflict, see Sect. 4.1.3 below), the base success chance of actions (see Sect. 4.5.1), and limits the use of more sophisticated equipment. SO is each actors' primary attribute, as defined by the STIX threat actor vocabulary `ThreatActorSophistication`. We have opted to use the following levels of Sophistication: Aspirants (1) show little to no technical capabilities and are usually accidental perpetrators (attacker) or common users (defender). Novices (2) have basic computer skills. Practitioners (3) are versed in using automated tools. Defending practitioners have a workable knowledge of the best practices and utilize pre-configured security solutions. Experts (4) have in-depth knowledge about system internals and operational security. Innovators (5) are masters of their field and create their own tailored tools and solutions.

Determination ***DE*** – Denotes the actor's overall motivation and drive. It influences Initiative and, in limited deck mode (see Sect. 4), the number of actions available to the players. For attackers, Determination specifies how intent they are to reach their goal, while a defender's motivation simply quantifies the typical level of effort invested into protecting a given asset. We define five levels of Determination: Indifferent (1) actors are not particularly interested in achieving their attack or defense goal – their willingness to invest time or resources is next to nil. Casual (2) determination represents a "for fun" attitude and might be representative of an intrinsically and extrinsically barely motivated individual [30]. Willing (3) actors will invest an average amount of resources into their mission without sacrificing other duties. Attackers or defenders classified as Devoted (4) will disregard work hours or will dedicate significant resources to their task. Finally, Zealous (5) actors will do everything in their power to achieve their goal, including breaking laws or disregarding personal health.

Wealth ***WE*** – Specifies the actor's available monetary resources that can be spent on equipment (see below). Wealth is typically dependent on the respective attacker and defender class. Certain organizations are more likely to have financial constraints for cyber operations or information security than others, as is apparent in Table 2. Like any other attribute, Wealth comes in 5 levels: Destitute (1) actors do not have any noteworthy budget, the Provident (2) rating is used for the financial assets of a typical middle-class individual, Endowed (3) describes the average operational budget of a small company, Rich (4) actors command funds of a successful medium-sized business, and Prosperous (5) organizations/individuals can easily spend corporate-size budgets on a campaign or defense objective. Specific numbers may be attached to these ratings, if desired. For simplicity's sake, each point of Wealth translates to 5 fictional credits that can be spent on $\langle Enablers \rangle$ and $\langle Disablers \rangle$ during character creation and setup.

Certain classes may come with modifiers to their attributes, representing an increased or decreased likelihood of having a distinctively high attribute score. These modifiers are listed in Table 2.

Continuing our above example, Alice now needs to specify the attributes for her Opportunistic Crusader and the victim education organization. She rolls a D6 six times, re-rolling eventual sixes. She scores '3', '2', and '4' for the attacker. As the Crusader class has modifiers as listed in the Table 2 (+2 Determination, −2 Wealth), the final attribute levels are Sophistication (3), Determination (4), and Wealth (2). On the defender side, there are additional modifiers. Bob's rolls of '4', '2', and '4' are translated into the final values of '4', '2', and '3' (-1 Wealth).

4.1.3 Resource pools

Next to the actors' main attributes, there are two additional actor resources that come into play when pitting two characters against each other. The first is *Initiative* (*INI*) or *time efficiency*. Initiative is a derived attribute that determines the number of overall actions each character can perform. Each of the actions introduced in Sect. 4.5 has an Initiative requirement and also uses a certain number of Initiative points when performed, making this attribute the game's main resource for attack and defense activities. There are actions that increase or decrease Initiative. This is representative for the time required by a hostile action and for how certain (counter)measures might influence the amount of time an actor has left to successfully thwart the attack. The game ends in a victory for the defender when the attacking actor has exhausted their Initiative without compromising the target asset.

The second derived resource is *Insight* (*INS*). Both attack and defense actions benefit from knowledge about the opponent. The more one party knows about the other, the more likely his or her chance of successfully conducting an action. The initial Insight Pool is zero, but might be modified by certain class properties. In the course of the game, the Insight Pool will gradually increase for both actors – enabling new actions and giving them the chance to better react to their adversary. The base detection chance of the defender is linked to the action category of the attack (see 4.5.1).

Alice and Bob determine Initiative for their two sides: With a Sophistication and Determination rating of 3, Alice's Crusader has an Initiative score $(SO * 2) + DE$ *of* 9*. The education institution targeted is rated at Initiative* $(4*2)+2 = 10$*, which is boosted to 11 by the planning policy that Bob implemented during setup. The unmodified Insight Pool for the attacker is 0 (no class bonuses), while the defender receives a +2 class bonus.*

4.2 Equipment

Equipment in the game serves multiple purposes. It represents the target of a hostile action, the systems used to support or thwart such an attack, and general tools and policies that can be bought and implemented. Equipment always comes at a monetary cost measured in credits and derived from the Wealth attribute of the actor (see Sect. 4.1.2). Tables 3 and 4 provide an overview for the respective equipment types. While it is still possible to purchase equipment during later stages of the game (see Sect. 4.4), a baseline of an actor's technical capabilities has to be established right off the bat. Both attacker and defender procure equipment individually without informing the opponent.

Attacker equipment mostly comprises attack tools ($\langle Enablers \rangle$) as per our base model) that modify the chance of success against specific target assets or actors by increasing their operator's attributes or base success chance values. Other tools (hacking suites, vulnerabilities) come with a Sophistication rating of their own, which can substitute for the attacker's attribute for operations not directly executed by the actor.

Defender equipment ($\langle Disablers \rangle$) encompasses tools that increase security, detect, or mislead the attacker. The effectiveness of a piece of equipment is measured by its contribution to detection or evasion as well as its action success chance modification (± percentage points).

In general, equipment is divided into the following categories, denoted in the game model as $\langle\langle\langle Attack\ Actor \rangle Enablers \rangle Type \rangle$ and $\langle\langle\langle DefenseActor \rangle Disablers \rangle Type \rangle$.

4.2.1 Disablers

The following presents the types of equipment available to the defending actor, which includes security solutions, policies, and the systemic assets themselves.

Assets (*AST*) are determined by the defender's class and are awarded automatically at no cost. It is the goal of the attacker to compromise one of these assets in order to win the game. See Sect. 4.3 for more information and Table 5 for character–asset allocation.

Security solutions (*SEC*) – Security solutions are technical systems and tools intended to prevent, detect, delay, or generally hinder hostile actions. In the context of the game, they are physically or logically connected to an asset such as the network or a user workstation, making it harder to successfully attack the service and its consumers. Specifically, we differentiate the following systems (also see Table 3):

– Prevention solutions (*Pre*) – increase the difficulty of successfully attacking an asset by assigning a penalty to the action's success chance (*decSC*). See Sect. 4.5 for more information on action success.
– Detection measures (*Det*) – boost the defender's ability to identify hostile actions by directly increasing the defender's detection chance ($incDC$).
– Delay solutions (*Del*) – increase the amount of time and effort required by the attacker to perform the hostile action. In game terms, delay systems increase the Initiative cost of the respective attack (*incINI*).
– Recovery solutions (*Rec*) – decrease the impact of hostile attacks after the fact. Specifically, recovery solutions can reduce the level of compromise of C, I, or A attacks (*decIMP.**).
– Countermeasures (*Cnt*) – describe generic technical solutions that improve the defender's ability to counter hostile attacks. Countermeasures provide a flat boost to the defending actor's Sophistication attribute in specific scenarios (*incSO.**).

Security policies (*POL*) – Security policies are non-technical measures that increase security on an organization level. Similarly to security solutions, they increase defender attributes, resources, or grant special abilities. However, the bonus applies to the defending organization as a whole, making them a powerful tool for establishing a secure baseline. Policies cost time and funds to implement. Below list is taken directly from the NIST SP 800-53 standard ([17], see Sect. 5 for more information about mappings and Sect. 4.5.3 for

Table 3 Exemplary list of security solutions (*SEC*). Level II systems always cost the double amount of credits to implement but also offer twice the bonus ($\langle EffectValue(EV)\rangle$)

$\langle Type\rangle$	$\langle Name\rangle$	$\langle Effect\rangle$	$\langle EV\rangle$	$\langle ET\rangle$	Cost
Pre	Host-based IPS I	*decSC*	5	H (all)	1
Pre	Network-based IPS II	*decSC*	10	N (all)	2
Pre	Content-based IPS I	*decSC(D.D)*	15	H (all)	0.5
Pre	Rate-based IPS II	*decSC(A.A)*	30	N (all)	1
Pre	Web application firewall I	*decSC(R.S,D.I)*	15	Web	0.5
Det	Host-based IDS II	*incDC*	10	H (all)	2
Det	Network behavior analysis system I	*incINS*	1	N	0.5
Det	Log analysis system II	*incDC*	10	H	1
Del	Sandbox I	*incINI*	1	H	0.5
Rec	Failover network II	*decIMP(A)*	2	N	2
Cnt	Stateful firewall I	*incSO*	1	all	2

Systems with $\langle EffectTarget(ET)\rangle$ = * (all) apply their benefit to all disablers of that type, while others can only be attached to one asset or security solution

Table 4 Exemplary list of attack tools. Level II systems always cost the double amount of credits to implement but also offer twice the bonus ($\langle EffectValue(EV)\rangle$)

$\langle Type\rangle$	$\langle Name\rangle$	$\langle Effect\rangle$	$\langle EV\rangle$	$\langle ET\rangle$	SO	Cost
MPT	Host exploit kit I	*incSC*	5	H (all)	1	1
MPT	Pentesting software II	*incSO*	2	all	2	4
Sca	Mobile OS scanner I	*decDC*	5	M (all)	-	0.5
Sca	ICS analysis tool II	*decDC*	10	I (all)	-	1
VSc	Cloud vuln. scanner I	*reduces VUL.cpx*	low	T (all)	-	1
NSc	Network mapper	*incINS*	1	N (all)	-	1
Pwd	Password cracker II	*incSC(D.I)*	10	H,N,I,M,T	-	1
Mal	Rootkit I	*decDC*	5	H,N,I,M,T	1	0.5
Mal	Ransomware II	*incCR*	4	H	2	1
Mal	Spambot I	*decINI(D.D)*	1	H	1	0.5

Systems with $\langle EffectTarget(ET)\rangle$ = * (all) apply their benefit to all disablers of that type, while others can only be used to attack one asset or security solution

more details about NIST-based defense actions) and cover the following aspects (NIST control ID):

- Access control (AC-1): Combines access control systems, policies, and session control mechanisms.
- Awareness & training (AT-1): Increases defender Sophistication by implementing organization-wide awareness and training measures.
- Audit & accountability (AU-1): Relates to audit-based non-repudiation measures.
- Security assessment & authorization (CA-1): Evaluation of controls and their adherence to e.g. external standards.
- Configuration management (CM-1): Encompasses establishing and maintaining performance and functionality of systems throughout their life cycle.
- Contingency planning (CP-1): Describes contingency plans such as fail-overs to alternate storage or processing sites.
- Identification & authentication (IA-1): Manages session control and (cryptographic) identification and authentication measures.
- Incident response (IR-1): Encompasses procedures, handling, monitoring, and reporting of incidents as well as their follow-up response.
- Maintenance (MA-1): General maintenance of systems, such as update policies and downtime schedules, are part of this policy.
- Media protection (MP-1): Subsumes access, designation, storage, transportation, sanitization, use, and downgrading of physical media containing relevant data.
- Physical & environmental protection (PE-1): Physical access control and environmental protection enforcement.
- Planning (PL-1): Meta-policy for the design of information security architecture, secure operation, and central management.
- Personnel security (PS-1): Manages personnel screening, transfer, tracking, and termination.
- Risk assessment (RA-1): Policy for the risk assessment and vulnerability scanning process.
- System & services acquisition (SA-1): Revolves around the system development life cycle, the allocation of

resources, as well as the procurement of (third-party) systems and services.
- System & communications protection (SC-1): Manages defense measures related to DDoS protection, shared network resources, crypto policies, VoIP, wireless link protection, I/O device assess and usage restrictions as well as numerous other factors contributing to secure (inter-)system communication.
- System & information integrity (SI-1): Includes countermeasures to malware, system monitoring, alerts, function validations, error handling, and other, primarily remediation-centered information protection activities.

4.2.2 Enablers

Listed below is the equipment available to the attacker:

Attack tool (*ATT*) – Attack tools are malicious programs coded to perform reconnaissance (scanners) circumvent security (hacking tools) or automate certain offensive tasks. They either provide a flat bonus or need to be attached to attack actions as one-time modifiers. See Table 4 for an exemplary list. Attack tools are categorized into:

- Multi purpose tool (MPT): These hacking tool-sets allow the attacker to automatically probe and exploit known vulnerabilities without purchasing a specific attack. If the attacking actor operates such a tool, they can decide to use the tool's Sophistication attribute for related actions instead of their own. The specific capabilities (actions that can be automated) differ from tool to tool and range from an increase in success chance ($incSC$) to an increase in Insight ($incINS$) or Sophistication ($incSO$).
- System scanner (Sca): Specific to each class of equipment (see Table 4), these tools increase the knowledge about a system ($incINS$) or aid via a reduction of the detection chance ($decDC$), thereby enabling more complex attacks. There are scanners for every type of asset that need to be coded or procured individually for each $\langle EffectTarget \rangle$ category.
- Vulnerability scanner (VSc): Vulnerability scanners determine the existence of weaknesses in host-based systems. In game terms, they reduce the complexity requirements of vulnerability-based attacks performed by the attacker. See Sect. 4.2.3 for more information on vulnerabilities and exploits.
- Network scanners (NSc): Tools in this category (packet sniffers, port scanners) intercept network traffic and thereby grant insight into the network environment and its connected systems ($incINS$). They additionally expose security solutions installed within the network context.
- Password cracker (Pwd): Primarily used to bypass account security, password crackers either brute-force passwords or attempt to login using a prepared list of likely secrets. Using them increases the success chance (*incSC.D.I*) of Intrusion type (*D.I*) attacks. See Sect. 4.5.1 for more information about attack phases.
- Malware (Mal): Malware summarizes all software that mirrors the harmful intent of an attacker in an automated fashion. In PenQuest, malware is 'attached' to a successful Delivery action (*D.**). We differentiate Rootkits (decrease detection chance of a subsequent attack, $decDC$), Backdoors (increase chance of success and decrease detection risk (*incSC*, *decDC*)), Ransomware (generate credits (*incCR*)), Trojans (similar to multi purpose tools and some scanners), as well as Botnet Zombies (reduce Initiative costs for certain kill chain phases (*decINI.req*). Like some multi purpose tools, malware has its own Sophistication level for determining its success. With the exception of backdoors, all malware can only be used once and expires after it has been triggered.

4.2.3 Exploits and fixes

Exploits and fixes have a special role in PenQuest. While also equipment by definition, exploits provide the means to lower the e.g. Sophistication requirements of an attack by aiding the attacker in her efforts. Fixes available to the defender directly counter specific exploits. The concept is detailed in the following:

Exploit (VUL) – A (technical) vulnerability is a flaw in a specific asset or security solution that makes it easier for the attacker to breach the system by exploiting it. They require a certain level of Sophistication to use and might demand additional privileges. Vulnerabilities generally exist in all types of defender equipment, but are hard to exploit without knowing of their existence. Depending on complexity cpx, they need a successful run of a vulnerability scanner on the victim resource (see above) before they can be used. Vulnerabilities are key to compromising the victim asset and therefore for deciding the outcome of the game.

Vulnerabilities as modeled by the RPG follow the metrics defined by CVSS: `Attack Complexity` cpx (high or low) determines the actor Sophistication requirements, `Privileges Required` prv (true or false) additionally decide the need for pre-existing user privileges, `User Interaction` usr (none or required) define whether the vulnerability can exist stand-alone without an accompanying action, and `CIA Impact` $imp.*$ represents bonuses (high (+2), low (+1), or none (+0) for each triad factor) that determine the modifier to the accompanying attack's effect. Each vulnerability is additionally rated in accordance with its `Exploit Code Maturity` mat, which determines a one-time Sophistication SO bonus and monetary cost of the exploit. Vulnerabilities without user interaction are assigned an SO value of their own, as hinted at by the Temporal Metric Group of CVSS. This directly affects the success chance of exploiting the respective vulnerability. In short, vulnerabilities also come with a Sophistication attribute or modifier that is derived from their CVSS score, which is linked in turn to the CAPEC attack pattern via their entry in the Common Weakness Enumeration (CWE) database [34]. See Sect. 5 for more information about model mappings.

In PenQuest, vulnerabilities are generally used together with an attack action for additional supporting or enabling effects. Corresponding exploits can either be coded by the attacker ($W.C$ action, as discussed in Sect. 4.5.1), or purchased (see above). Either way, a successful Reconnaissance ($R.*$) or $W.P$ action is needed to enable the use of vulnerabilities. In our current iteration of the game, we have implemented a number of real-world vulnerabilities including some well-known ones such as ShellShock,[6] GHOST,[7] and Heartbleed.[8]

Fix (FIX) – Fixes directly counter vulnerabilities or address other weaknesses in existing assets. They come in several classes as per the `Remediation` metric of CVSS: Official Fix, Temporary Fix, and Workaround. Official fixes (high SO, restore integrity from 'compromised' to 'nominal' (3 increments, $incINT$)) and workarounds (low SO, restore from 'compromised' to 'affected' or 'highly affected' to 'nominal' (1 increment)) are effective indefinitely but are not able to counter zero-day exploits. Temporary fixes (medium SO, restore by 2 increments) are only effective for one game turn and might come with side effects. Workarounds generally come with a lower chance of success, conditional on their Sophistication. See Sect. 4.3.1 for more information about compromise levels.

In our example, the attacking Crusader's Wealth rating is 2. Translating to 10 credits' worth of funds ($Wealth * 5$), this gives the attacker only a few options to prepare for her campaign. Alice spends the immediately available 50% of the money on a host exploit kit (1 credit), a level II vulnerability scanner for database systems (1 credit), and a 'functional' vulnerability for database systems for 2 credits. For the defender, Bob uses 7 of his 15 credits to implement an access control policy (2.5 credits), a planning policy (0.5 credits), a level II host-based intrusion detection system (2 credits), and a level I stateful firewall (2 credits). During the game, the remaining credit amount can be spent on additional equipment. Therefore, Alice's Crusader has an operational budget of 6 credits, while the education organization can spend 8 credits during play.

[6] https://nvd.nist.gov/vuln/detail/CVE-2014-6271.

[7] https://nvd.nist.gov/vuln/detail/CVE-2015-0235.

[8] https://nvd.nist.gov/vuln/detail/CVE-2014-0160.

4.3 Assets and topology

Before we select the victim of the campaign, we have to take a look at the types of assets available in the game. Assets are determined by the defender's class and are awarded automatically at no cost (see Table 5). It is the goal of the attacker to compromise one or several of these assets in order to win the game. Assets are part of the defender's infrastructure and operate within the context of a network (which is an asset itself). Below, we take a closer look at specific assets and some of the more general equipment concepts in regards to the modeled topology.

We generally differentiate the following asset types: Host-based (H), network-based (N), industrial (I), mobile (M), and third-party (T) assets. Model-wise, we use a simplified version of the `SystemType` vocabulary of STIX' `VictimTargetingType` TPP schema to model individual assets (denoted in brackets):

- Application server (App, internal, H) (Enterprise Systems–Application Layer): Generic server running an organization-relevant application.
- Database server (DB, internal, H) (Enterprise Systems–Database Layer): Generic data and/or configuration store.
- Network (Net, internal and exposed, N) (Enterprise Systems–Network Systems, Enterprise Systems–Networking Devices): Underlying network connecting all other assets. We differentiate an exposed demilitarized zone (DMZ), a local area network (LAN), and an industrial network (subsumed under the term 'SCADA'). A dedicated internal network is optional for the PI defender class.
- Web server (Web, exposed, H) (Enterprise Systems–Web Layer): Server hosting the public web presence of an organization or individual.
- Communication system (Com, internal and exposed, H) (Enterprise Systems–VoIP, Enterprise Systems–Web Layer): Communications infrastructure including, but not limited to, telephony, e-mail, and instant messaging.
- Industrial control system (ICS, internal, I) (Equipment Under Control, Operations Management, Supervisory Control): System controlling industrial equipment such as manufacturing plants.
- Industrial safety system (ISS, internal, I) (Industrial Control Systems–Safety, Protection and Local Control). Safety systems for prevention and mitigation of disadvantageous scenarios affecting human health.
- Mobile system (Mob, exposed, M) (Mobile Operating Systems, Near Field Communications, Mobile Devices): Mobile devices and (individual) short-range communications tools.
- Third-Party service ($3Pa$, exposed, T) (Application Stores, Cloud Services, Security Vendors, Social Media, Software

Table 5 Mapping of defense actor classes to controlled assets

	CP	CM	CS	IF	MI	SA	ED	PI
Application server (App*)	✓	o	✓	o	o	✓	✓	❈
Database server (DB*)	✓	✓	✓	✓	✓	✓	✓	
Network segment (Net)	o	o	o	o	o	o	o	o
Web server (Web)	❈	✓	o			✓	o	✓
Communication system (Com)	o	o	o	o	o	o	o	o
Communication system (Com*)		✓	✓	✓	✓	✓	✓	
Industrial control system (ICS*)	✓	✓		✓	❈			
Industrial safety system (ISS*)	❈	❈		✓	❈			
Mobile system (Mob)			✓	❈	✓	✓	❈	o
Third party service ($3Pa$)			✓			❈	✓	✓
Workstation (WS*)	o	o	o	o	o	o	o	o

o...no Sophistication requirement (all actors of this type control at least these assets), ✓...Sophistication 2-3, ❈...Sophistication 4-5. Asterisk (*): Internal system (LAN or industrial network)

Update): Services such as cloud storage, outsourced web services, and supplier systems.

– User workstation (WS, exposed for actor PI, otherwise internal, H) (Application And Software, Workstation, Removable Media): Physical or virtual machine operated by the end-user.

When it comes to targeting assets, there are three key concepts that define how a system can be attacked: asset compromise, attack vector, and asset dependency.

4.3.1 Asset compromise

Asset compromise is the key principle that governs how an attack on a system is modeled in PenQuest. As stated in Sect. 3.1 and formalized in Sect. 3.3.2, there are three kinds of attacks with a $\langle Mode\langle Rating\rangle\rangle$ ranging from 'low' (1), 'medium' (2), to 'high' (3). The available mode and its rating is dependent on the action being used; not all actions provide the same level of systemic impact (see Sect. 4.5 for more information). There are three modes of attack of which the attacker can choose one at the time of the hack:

– **Confidentiality** attacks with a rating of 'high' (3) increase the attacker's Insight pool (*incINS*), but have no further effect on system integrity or status. The effect is cumulative over time: Three successful 'low'-rated (1) attacks or one 'medium' (2) plus one 'low'-rated attack accumulate to the same effect. A successful 'high' (3) level confidentiality attack is necessary to win a game with a data theft (confidentiality) scenario.
– **Integrity** attacks of 'low' (1) and 'medium' (2) rating set the targeted service's integrity (*decINT*) to 'affected' or 'highly affected', respectively. An attack rated 'high' (3) will change integrity to 'compromised', which is required to progress along the attack vector and to win sabotage scenarios. The effect is again cumulative.
– **Availability** attacks target the victim's status ($\langle Victim \langle Status\rangle\rangle$): One or several successful attacks (again dependent on the rating) set the target's status to 'stopped' ($\langle Enabler\langle Effect\rangle\rangle$ = *decSTA*), representing a system that is no longer operational.

Any successful Integrity compromise (rating 3) of an asset or Disabler within the 'exposed' or 'internal' exposure domain enables the attacker to target connected 'internal' systems, which would otherwise be inaccessible (see Fig. 5). See the next section for more information about the attack vector.

4.3.2 Attack vector

We model network topology by differentiating between *exposed* and *internal* systems (see Sect. 3.1). In order to attack an internal system, the attacker has to first compromise an exposed parent asset or security solution or proceed along a predefined *attack vector*: The attack vector describes the path an attacker must take in order to reach the desired victim, meaning that any attack on a still uncompromised topology has to target the mobile system (Mob, type M), third party service ($3Pa/T$), external comm system (Com/H), web server (Web/H), user workstation (WS/H), or the exposed demilitarized zone network (Net/N) itself. Incidentally, these are also the systems the defender should do his or her best to wall up. Once a system has been fully compromised integrity-wise (see Sect. 4.5.2), the attacker can target any system connected to the one he or she just hacked. Figure 5 depicts the concept as bold black arrow.

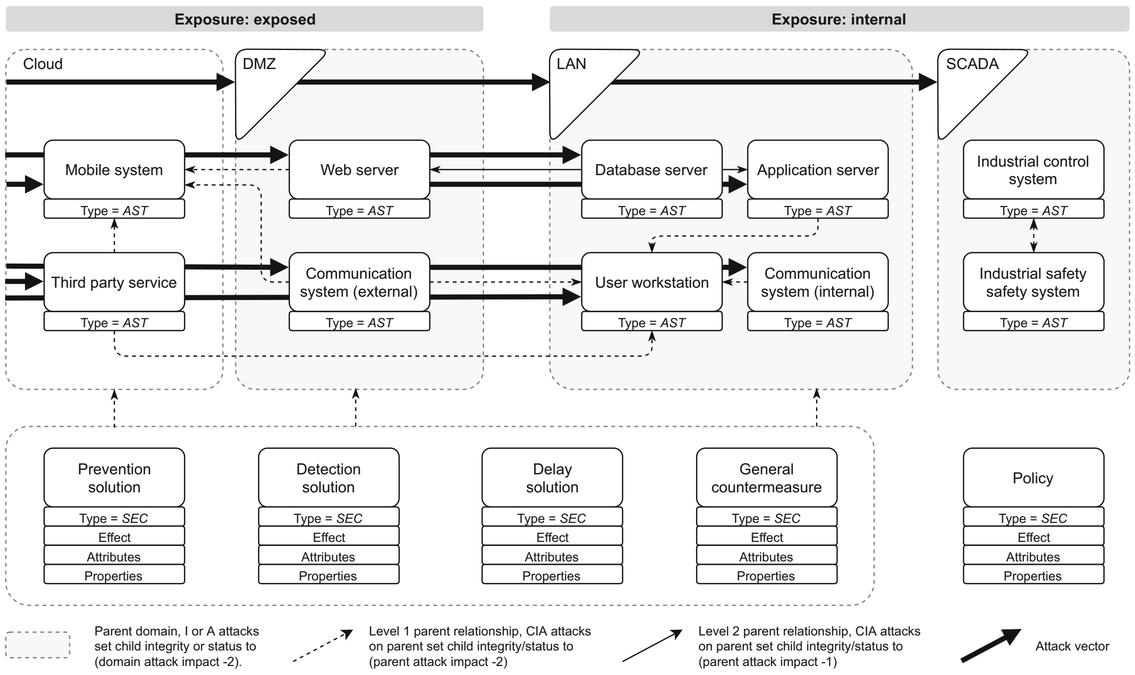

Exposure: exposed
Exposure: internal
Cloud
DMZ
LAN
SCADA
Mobile system
Type = AST
Web server
Type = AST
Database server
Type = AST
Application server
Type = AST
Industrial control system
Type = AST
Third party service
Type = AST
Communication system (external)
Type = AST
User workstation
Type = AST
Communication system (internal)
Type = AST
Industrial safety safety system
Type = AST
Prevention solution
Detection solution
Delay solution
General countermeasure
Policy
Type = SEC
Effect
Attributes
Properties
Parent domain, I or A attacks set child integrity or status to (domain attack impact -2).
Level 1 parent relationship, CIA attacks on parent set child integrity/status to (parent attack impact -2)
Level 2 parent relationship, CIA attacks on parent set child integrity/status to (parent attack impact -1)
Attack vector

Fig. 5 Attack vectors and parent–child relationships for both victim exposure levels in the abstracted topology. Security solutions are parents to assets; attacking them on the integrity level can set child integrity to 'affected'. The effect of CIA attacks on assets and other disablers is further detailed in Sect. 4.5.2

4.3.3 Asset dependency

Assets and security solutions may be *parent* (provider) to another system. If a parent is successfully attacked, the effect of the compromise is passed on to all children (consumers) at a reduced rate. Specifically, a 'high' (3) level attack on a parent will also have 'medium' (2) impact on the child for level 2 relationships, and 'low' (1) impact for level 1 relationships. An overview of attack vectors and parent–child relationships is depicted in Fig. 5.

The assets that Alice might be after are provided automatically. In the case of our education organization with a Sophistication attribute of 4, Bob the defender controls an application server, a database server, a web server, a communication system, third party services, user workstations, an industrial control system for educational purposes, and various network resources (see Table 5).

4.3.4 Victim selection

Before the game starts in earnest, the attacker picks the actual *Target* ($\langle Victim\langle Type = \{Asset\}\rangle\rangle$) in the attacker/defender model) and *Mode* ($\langle AttackClass\langle Mode\rangle\rangle$), i.e. one of the defender's assets to compromise by changing its $\langle Victim\langle Integrity\rangle\rangle$ to 'compromised' or its $\langle Victim\langle Status\rangle\rangle$ to 'stopped' (also see Fig. 5). There are three kinds of possible attacks on any asset: Confidentiality attacks (theft of information), integrity attacks (altering of information), and availability attacks (system status changes). See Sect. 4.5.2 for more information about attack actions.

For custom games, it is perfectly feasible to specify more varied or numerous goals. An attacker could e.g. attempt to first steal information off an asset before shutting it down. Other scenarios include penetration tests of all assets within a network segment or a DoS attack on two systems simultaneously.

Alice decides that the game will end once she successfully alters the User Database asset of the defender (integrity attack of the 'System manipulation' (Action on Objective–Integrity attack) category (see Sect. 4.5 *below), changing some of the grades in the process.*

4.4 Game phases

At this point we are ready to play. We differentiate two distinct game modes: In *limited deck* mode, attacker and defender randomly select actions (i.e. a card) from the pool. After an action was used it is discarded from play. At the end of a turn, players draw a new card for each action spent. Limited deck mode has been designed with gameplay mechanics and balancing in mind and is intended to be used in casual games.

In *simulation mode*, players may freely choose from the entire action pool at any given time. Attack actions are only discarded if the defender successfully spotted and identified the hostile activity. This mode represents a more realistic approach where attackers and defenders always have the full strategy set at their disposal.

As depicted in Fig. 6, the game starts off in *Stealth phase*. This means that the defender is unaware of the looming threat and has to rely on his initial equipment to keep his assets safe. Stealth phase lasts until the defender manages to successfully detect an attacker action. Actions can be performed for free during Stealth phase, meaning that they do not deduct from the Initiative total. Purchasing equipment is done normally at the end of the round. To reflect the reduced activity of the defender during this stage, he or she generates a certain amount of credits every two rounds, providing the actor with additional procurement options. Each round, attacker and defender may spend one Initiative point worth of actions. Typically, the attacker uses this phase to conduct reconnaissance to increase his or her Insight Pool. Defenders typically spend a part of their operational budget to improve security systems, conduct spot checks (use defense actions on systems deemed likely victims) or simply hope for their detection systems to pick up the attacker's activity. If the defender manages to detect a hostile action, the action and its resulting level of compromise of the targeted system is unveiled and Stealth phase ends with the current turn.

Alice wants to stealthily assess the target system by conducting a reconnaissance 'Scan' action (see Sect. 4.5). The rounded down success chance of this action is 60% (Alice's base Sophistication rating +3), meaning that she has to score 6 or less on a ten-sided die. Alice rolls her D10 and scores a '5' – well within range of the success threshold of 1-6. As a result, her Insight pool is increased by 1, boosting the base success chance by 5% and enabling a greater range of invasive actions for her subsequent turns. Bob now rolls another D10 to determine if the action was caught by his monitoring systems. With his base detection chance (30%) and an Insight pool of 2 (+10% chance of detecting a hostile action) he needs to score a '4' or less to spot Alice's scan. Equipment on both sides would additionally modify this target value (see example in Sect. 4.5.2).

With the end of the Stealth phase the game enters *Conflict phase*. Initiative is now deducted normally for each action performed, which represents the ticking clock. There are seven possible action types that correspond to the APT cyber kill chain by Hutchins et al. [16]. Every APT stage is further split into subcategories [27] that are ultimately linked to specific attack patterns. See Sects. 4.5.1 and 5 for more information about the interplay and mapping of offensive and defensive actions.

Attacker and defender continue to act alternately. Each attacker action comes with a success and detection chance (for specifics, see Sect. 4.5) that is modified by equipment on both sides. Whenever the defending actor succeeds in detecting a hostile action, the action itself and the current level of compromise of the system are unveiled, giving the victim the chance to understand the attacker's goals and react accordingly with a defensive action, which, similarly, has a certain chance of being successful.

In addition to triggering a normal attack or defense action, the actors can use their remaining credits to procure equipment from a randomly drawn pool of equipment cards (limited deck mode), or from the full range of enablers/disablers (simulation mode). Equipment is available for use at the beginning of the next turn to model necessary implementation efforts.

Each hostile action further increases the chance of successfully compromising the target asset, while each reaction of the defender will make it harder for the adversary to penetrate the system. Misdirection and timing are key, just like in the real world: It is vital for the attacker to conceal his or her ultimate goal for as long as possible and to strike when the chance of success is greatest. All the while the defender has to protect their assets to the best of their abilities until all Initiative has been used up and the attack has been averted.

4.5 Actions

In this section, we discuss PenQuest's action categories as well as the individual actions used to compromise or defend an asset. The core mechanics introduced in Sect. 3.3.2 are hereby specified and linked to accepted vocabularies and standards.

Actions are at the heart of PenQuest. They represent the concrete attack or countermeasure being used at a given point in time. Each action comes with a success chance and a detection chance modified by actor attributes and equipment currently in play. The attacker first rolls a die (typically a D10 or D100 to represent percentages) to determine whether the action is successful. If the result of the roll is equal or lower than the success chance threshold, the action succeeds. The defender now rolls another die in an attempt to detect the event after the fact. If that succeeds, the current action of the attacker is unveiled.

In the following, we introduce concrete attack and defense actions, which are based on the Common Attack Pattern Enumeration and Classification (CAPEC) dictionary and classification taxonomy by MITRE [32] as well as on the mitigation controls listed in NIST Special Publication (SP) 800-53 [17]. Attack actions are linked to the APT kill chain by Hutchins et al. [16], which we expanded with several subcategories modeled in the TAON ontology [27]. Figure 8 in

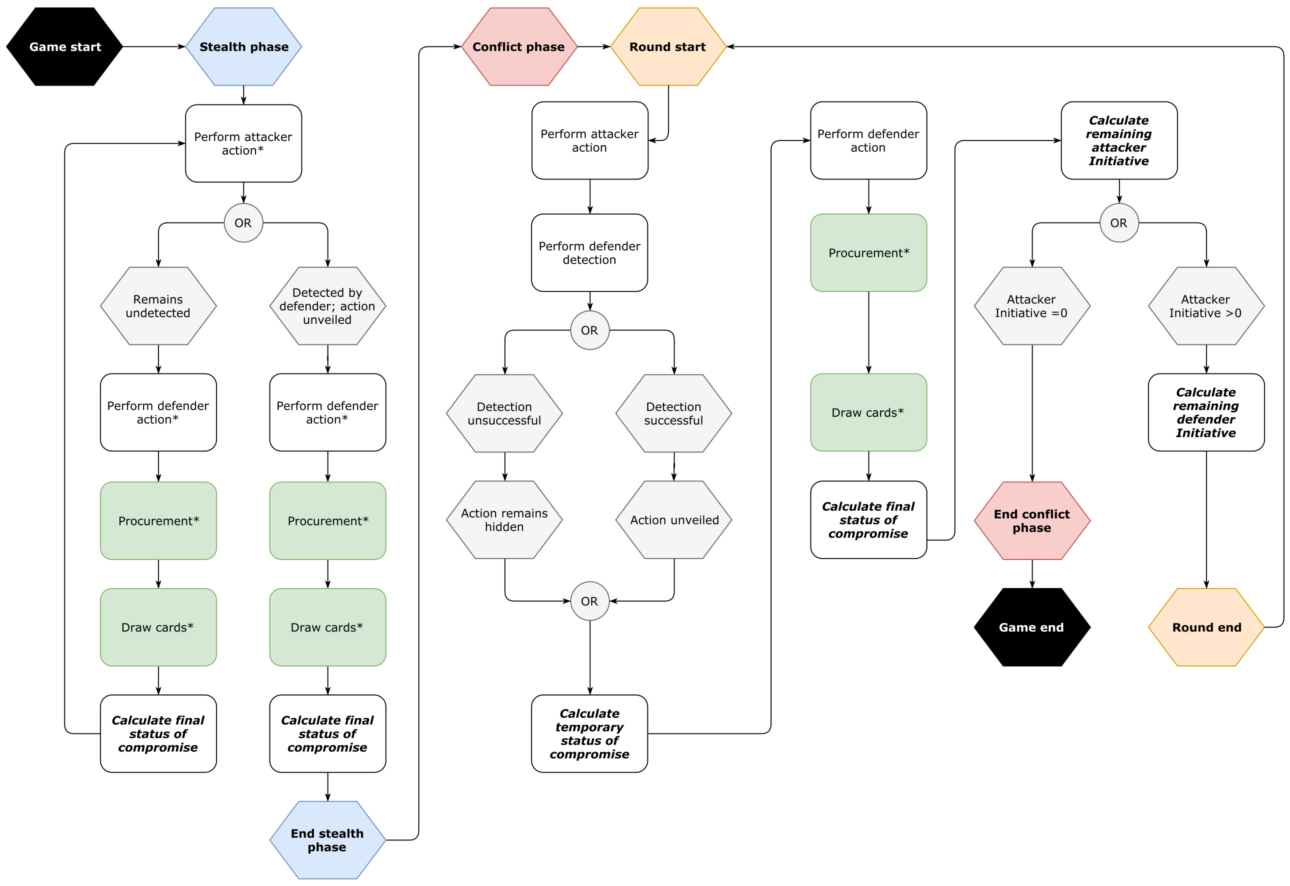


Fig. 6 Game phases and round progression overview. Actions marked with an asterisk (*) do not reduce the respective actor's Initiative pool. Initiative and equipment status of compromise is recalculated after both actors have completed their actions

Sect. 5 summarize the link between kill chain, attack mechanisms, and mitigating controls.

4.5.1 Action categories

Action categories encompass and classify the individual attack operations conducted by the aggressor. Each subcategory of these kill chain actions has a base detection and success chance ranging from 0 to 100%. The Sophistication attribute of the actor or tool, employed vulnerabilities, special attacks, and various countermeasures might modify that base value. The percentage is currently defined manually by a group of IT security experts, but may easily be altered to match newly released information in future iterations. Some kill chain stages require the successful completion of an attack belonging to another stage prior to execution (see Fig. 7), depending on desired game complexity: In *APT mode*, dependencies include persistence and the establishment of a C2 channel. *Quick mode* omits these stages and only requires a successful 'Launch' action ($I.L$).

Each specific attack action (see Sect. 4.5.2) is assigned one or several action categories. Specifically, these APT kill chain categories are:

- **Reconnaissance** ($R.*$): Research into the target and scanning of related assets for information. Subcategories include Research ($R.R$) using public search engines, Identification ($R.I$) of systems through e.g. fingerprinting, and Scan ($R.S$), where a victim system is actively scanned for weaknesses and topological properties. Successful reconnaissance enables the procurement of vulnerabilities.
- **Weaponization** ($W.*$): Preparing exploits and weaponizing code. Weaponization mostly takes place at the attacker's premises and is therefore nigh impossible to detect. Its subcategories are Preparation ($W.P$), which includes exploit searches and targeted research, the Coding ($W.C$) of exploits and tools, as well as Embedding ($W.E$) the prepared or purchased malware in websites, mail messages, or other, ostensibly harmless media.
- **Delivery** ($D.*$): Delivery actions describe the process of gaining access to or smuggling payload into the victim's perimeter. Specifically, we differentiate Deception ($D.D$) attacks that use logical or physical social engineering to fool the victim, and straightforward Intrusion ($D.I$): Here, the attacker actively tries to penetrate the target's system using technical means.

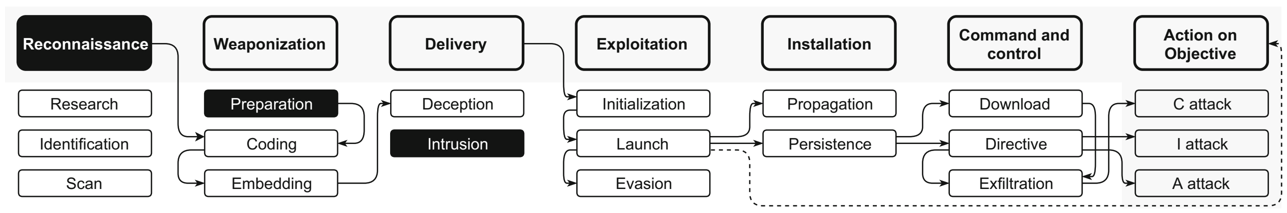


Fig. 7 APT stage completion dependencies. The arrows represent which stage (i.e. action associated to the stage) needs to be completed before another stage action can be executed ('followed by'). The dashed arrow shows the simplified dependency for quick (non-persistent) attacks. Black boxes represent possible start points for kill chain traversal

- **Exploitation** (*E.**): In this stage, a payload or attack code is actively executed on the system. During Initialization (*E.I*), malware or an exploit is prepared for launch by abusing a system weakness. Launch (*E.L*) describes worker processes, threads, services, or modules that are being started, marking the point in time where malicious code commences operation. The Evasion (*E.E*) subcategory encompasses techniques that hinder or prevent the analysis of an ongoing attack.
- **Installation** (*I.**): This stage covers Propagation (*I.Pr*), which is all about spreading malware infections and the vertical traversal towards the target. Persistence (*I.Pe*) attacks, on the other hand, attempt to establish a permanent foothold in a system.
- **Command and Control** (*C.**): The C2 channel of an APT is responsible for communication between the victim and the malicious controller. This stage consists of the Download (*C.Do*) category, which includes patching and update mechanisms that alter or expand the original function of malware or exploits, the Directive (*C.Di*) category, which subsumes commands sent via the C2 channel that potentially alter an attack's original purpose, and the Exfiltration (*C.E*) aspect, which includes smuggling out of e.g. previously stolen information.
- **Actions on Objective** (*A.**): These actions encompass the actual victim attack task performed after going through some or all of the above kill chain stages. They again correspond to the CIA triangle of information security, which is also referred to as C, I, and A impact. Every attack action with a suitable CIA impact other than 'none' can be used as $A.*$ action.

4.5.2 Attack actions

Attack actions represent dedicated hostile behavior, which aims to compromise a target asset or security solution in order to steal information, alter its operational parameters, or negatively influence its availability. The main bulk of these actions and the aforementioned CIA attacks performed by the hostile actor (see Sect. 4.3.1) is taken from the `Mechanisms of Attack` described in the CAPEC classification. These mechanisms encompass several levels of hierarchy and a description of possible countermeasures – subsequently translated to the defender's arsenal via the NIST Security and Privacy controls (SP 800-53) standard ([17], see Sect. 4.5.3 below). In our game model, this mapping links the APT stages with their base detection and success chances to an existing database of usable attacks as well as numerous possible countermeasures. All actions, with their classification into confidentiality, integrity, and availability attacks, are linked directly to the individual mechanisms of attack through the CIA `Impact Rating` provided by the CAPEC standard. Similarly, the mapping between TAON's APT kill chain subcategories and our primary classes of attack is done partly via CAPEC's `Purpose` information: Attack patterns are separated into reconnaissance, penetration, and exploitation categories, which directly map to the kill chain's Reconnaissance, Delivery–Intrusion, and Exploitation stages. The remainder of links (also see Fig. 8) is assigned manually.

The types of attacks (*primary attacks*) mapped to the [APT kill chain] include (official CAPEC terminology, where differing):

- **Information Gathering** ***IG*** [Reconnaissance–Identification, Scan] (Analysis): These attacks include interception, finger- and footprinting and various reverse engineering and buffer manipulation tasks aiming at generating a better understanding of a target system.
- **Injection** ***IN*** [Exploitation–Initialization, Launch]: Injections control or disrupt the behavior of a target or enable the installation and execution of malicious code.
- **Social Engineering** ***SE*** [Delivery–Deception]: These actions increase the trust in the malicious entity by spoofing legit content or identities through social engineering.
- **State Attack** ***SA*** [Exploitation–Initialization, Launch] (Time and State): State attacks try to illegally change the state or timing of an application to gain access to otherwise protected resources.
- **Function Abuse** ***FA*** [Exploitation–Initialization] (API Abuse): The abuse of existing API and protocol functionality typically aims at information exposure, vandalism, degrading or denial of service, or the execution of arbitrary code on the target.

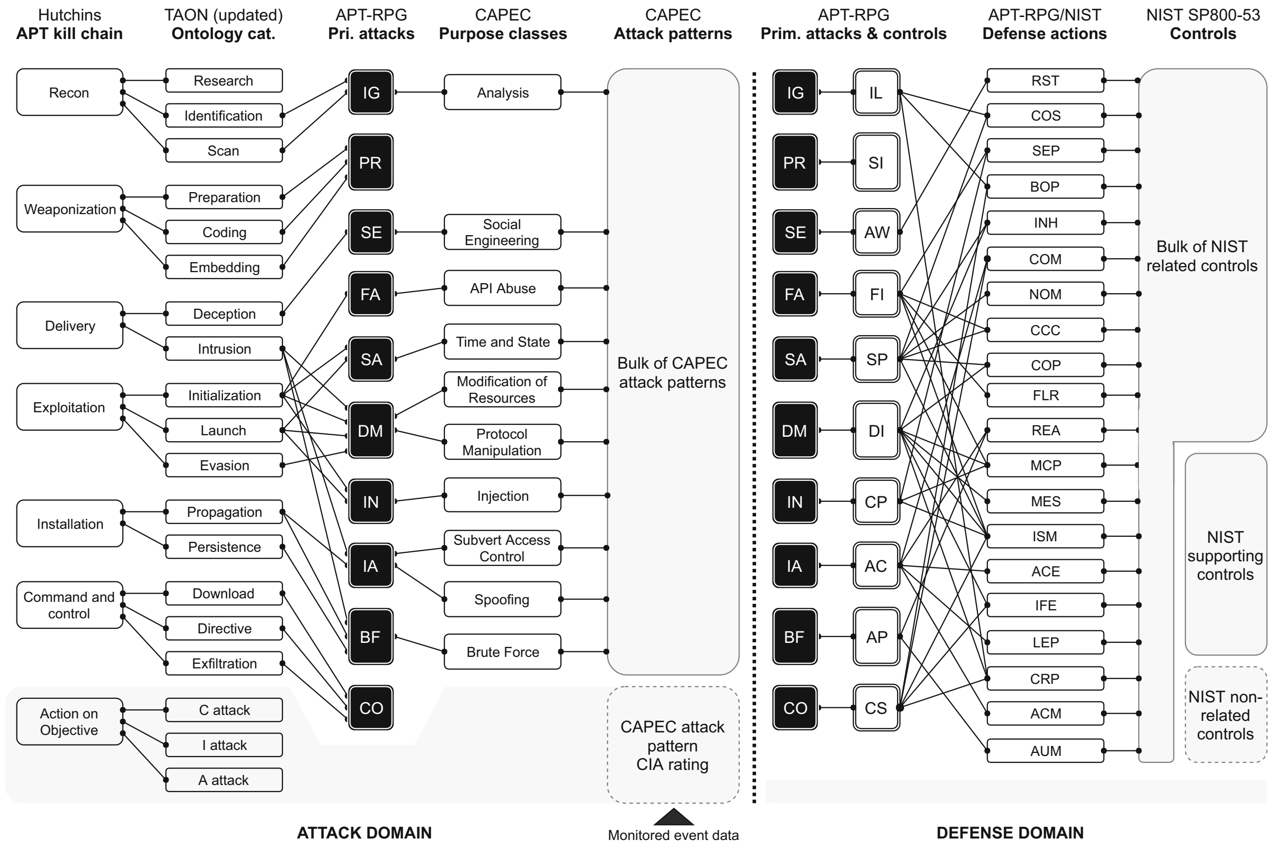


Fig. 8 Mapping of NIST controls to CAPEC attack patterns via extended APT kill chain. The introduced link categories based on CAPEC are highlighted in black

- **Brute Force** ***BF*** [Delivery–Intrusion, Installation–Propagation, Persistence]: These techniques explore and overcome security measures of the target by e.g. brute-forcing passwords.
- **Illegal Access** ***IA*** [Delivery–Intrusion, Installation–Propagation] (Subvert Access Control, Spoofing): In this large class of attacks the adversary attempts to bypass access control mechanisms to gain control over a system or data store.
- **Data Manipulation** ***DM*** [Delivery–Intrusion, Exploitation–All] (Modification of Resources, Protocol Manipulation): Attack actions of this category exploit the characteristics of data structures to gain illegal access or to interfere with the secure operation of a system. They may also alter the system's integrity by manipulating software, files, or otherwise interfere with the operation of an infrastructure.

The two remaining APT stages, Weaponization and Command and Control, are maintained as individual categories independent from CAPEC. Their attack patterns are currently realized through manually defined actions, which will eventually be derived from other information sources.

- **Preparation** ***PR*** [Weaponization–All]: These attacks describe actions performed on the premises of the attacker to prepare attack tools, research information about the chosen target, and other preparatory tasks invisible to the defender. In the game, weaponization is typically used to generate Insight or reduce the costs of equipment by spending time on e.g. malware coding.
- **Communication** ***CO*** [Command and Control–All]: C2 traffic is generated whenever a piece of resident malware receives new commands from its malicious operator. PenQuest uses C2 actions to e.g. allow the attacker to change a previously triggered attack action with a reduced risk of detection. See APT kill chain above for more details.

Table 6 lists some exemplary attack actions and their in-game properties, as well as their APT kill chain mappings. $\langle ID \rangle$, $\langle PatternClass \rangle$, $\langle Stage \rangle$, and CIA impact ($\langle Mode \rangle$) are retained as part of the model. The remainder is used for linking APT kill chain elements and CAPEC patterns that constitute our attack actions.

Each action comes with a Sophistication requirement specifying the level of knowledge needed to execute an attack. It is directly derived from the `Attacker`

`Knowledge Required` information as identified in CAPEC. The attacker can reduce this prerequisite by employing exploits (see Sect. 4.2.3).

Eventually, Alice's tampering is caught by Bob, triggering the Conflict phase of the game. Alice managed to boost her Insight pool to an impressive 5 points during Stealth phase, which now allows her to be rather aggressive in her approach. Alice still has 9 Initiative points at her disposal (she managed to complete reconnaissance and weaponization during her stealth play), while Bob is left with 11 points for use during the open conflict.
Alice wants to take it slowly to maximize her chances. As she can't directly attack her internal prime target, she attempts to take over an exposed asset first. With her initial conflict action, she tries to place her previously weaponized backdoor malware on the education organization's web server for easier subsequent penetration. To upload her code, she needs to successfully complete a 'Delivery' action. Alice elects to use a D.I kill chain action, which translates to either a DM, IN, or PH primary attack. Going the data manipulation route (DM), she executes CAPEC attack pattern 105: "HTTP Request Splitting". This particular attack is rated C (medium), I (medium), and A (low). Since Alice is not after information stored on the web server, she opts for an integrity attack. To be successful, she needs to roll a test with her unmodified Sophistication attribute (3) plus Insight modifier (+25%). Thanks to her preparation and her host exploit kit (another +5%), Alice would have a 60% chance to succeed. Unfortunately, Bob's access control policy (−10%) represents a significant obstacle, reducing the final success chance to 50%. Rolling her die, Alice scores a '2' – easy victory. The integrity of the web server is temporarily set to 'highly affected' (2 increments). If unopposed, she would now only need one more 'low' rated attack (which is made significantly easier by the 'Backdoor' malware card she attached to the attack) to completely take over the server, thereby opening the path to her actual victim. However, Bob still has a chance to react – if he manages to detect the attack. Thanks to his Sophistication (3), his Insight (+10%), his packet filter (+1 SO), and his IDS (+10%), he has a 60% chance to detect Alice's hack. He scores a '5' – and succeeds. The attacker has to unveil her past action to him, giving Bob the opportunity to restore the affected system's integrity and to anticipate Alice's future actions.

4.5.3 Defense actions

Defender actions directly counter aforementioned attacks and represent the defensive actor's response to hostile activity modeled by PenQuest. In general, each implemented control (i.e. defense action) identified in NIST SP 800-53 translates to one of several defense actions that counter a number of attack actions in the context of our RPG. Some actions directly relate to equipment (especially policies, see 4.2) and allow the defender to implement organizational changes that improve security. Others describe the conduct of security scans, the restriction of access to data, spam protection, or the setup of a honeyclient.

In game terms, a successful defense action lowers the rating of the previously executed C, I, or A attack by a certain amount of points. For example, a defense action might lower the effect of a 'high' (3) level availability attack to 'medium' (2), therefore preventing the system shutdown intended by the attacker, while not entirely mitigating the attack.

To bridge the gap between attack actions that are part of CAPEC's vocabulary and the controls of the NIST security standard, we apply a similar, yet more comprehensive and, to a certain degree, automated approach to the one introduced by [10].

First, controls are split into two distinct categories that describe the scope of the security measure:

– **Organization level**: Controls that target the organization level apply to all assets and security solutions currently in play. The NIST standard [17] contains 167 organization-level controls, including policies. To losslessly reduce this amount to a more manageable number, we categorize them into *primary controls* and *defense actions*, both of which can be found below. Organization level controls are designed to cost an increased amount of Initiative (time) to implement.
– **Information system level**: If a control specifically relates to an information system, they can only be applied to one system of the defender's choice once an attack on that system has been spotted (successful detection). There are 57 information system controls in NIST SP 800-53. In the game's context, we support three modes for information system controls derived from the NIST standard:

  1. *Abstracted controls*: With a focus on accessibility, this mode of PenQuest implements an information system version of each of the below primary (organization level) controls. For example, the Account Management (*ACM*) primary control can simply be used as organization-wide or information system variant.
  2. *Related controls*: For increased modeling accuracy, we can utilize NIST's `Related Controls` asso-

ciated to below primary categories as system-level equivalent. See the Account management (*ACM*) primary control for an exemplary list.

3. *Control enhancements*: Players can also opt to use NIST `Control Enhancements` for each of the primary controls as information system-level defense measure, provided the primary control is not already an information system control (marked by an asterisk). This mode can be adapted to have control enhancements serve as sole defender action set, omitting organization level controls (except policies) entirely. PenQuest was implemented and tested using this mode.

In the next step, we identify key controls that can be used to accurately depict a class of actual countermeasures. While NIST provides 17 families with a total of 224 controls, these are typically too high level in their categorization to serve as links to the more technical attack patterns used as attack actions. If we would use families directly, the game's rules would have to be adapted to connect each individual control to an attack action, which does not scale well and negatively impacts the applicability of the model to other domains. We therefore specify a number of *primary controls*, which are the polar opposite of the primary attacks defined in Sect. 4.5.2:

- **Information Leakage Protection *IL*** (counters *IG*): This primary control prevents information leakage through diligent configuration and data protection. It is associated with NIST's Configuration Settings (*COS*), Boundary Protection (*BOP*), and Cryptographic Protection (*CRP*) defense actions (see below).
- **Context Protection *CP*** (counters *IN*): As control against injection attacks, this category protects from undesired functionality that lets the attacker break out of the current system, communications channel, or application. Associations: Security Engineering Principles (*SEP*), Malicious Code Protection (*MCP*), and Information System Monitoring (*ISM*).
- **Awareness *AW*** (counters *SE*): This group of controls helps the defending organization to raise awareness for social engineering attacks of any kind, including spear phishing and physical intrusion attempts. Association: Role-based Security Training *RST*.
- **State Protection *SP*** (counters *SA*): Protecting the state of an information system is a vital task spanning several groups of actions, ranging from backup systems to integrity protection measures and status monitoring. It is associated with Configuration Change Control *CCC*, Configuration Settings *COS*, Contingency Plan *COP*, Incident Handling *INH*, Nonlocal Maintenance *NOM*, and Information System Monitoring *ISM* actions.
- **Function Integrity *FI*** (counters *FA*): Similarly, function integrity controls make sure that the available functionality (API, commands) of an application are not in any way abused. NIST associations include: Configuration Change Control *CCC*, Security Engineering Principles *SEP*, Malicious Code Protection *MCP*, and Information System Monitoring *ISM*.
- **Authentication Protection *AP*** (counters *BF*): This control group is primarily concerned with managing authenticators such as passwords and tokens. Associated controls: Remote Access *REA*, Authenticator Management *AUM*.
- **Access Control *AC*** (counters *IA*): As a main countermeasure to a wide range of intrusion attacks, the access control family subsumes account management, enforcement strategies, and various access-related policies. Associated controls are: Account Management *ACM*, Access Enforcement *ACE*, Continuous Monitoring *COM*, Least Privilege *LEP*, and Remote Access *REA*.
- **Data Integrity *DI*** (counters *DM*): Maintaining the integrity of data is one of main tasks of information security. In our gamified model, this primary controls includes: Contingency Plan *COP*, Incident Handling *INH*, Cryptographic Protection *CRP*, Malicious Code Protection *MCP*, and Information System Monitoring *ISM*.
- **Security Intelligence *SI*** (counters *PR*): Preparation for an attacks works both ways: Potential victims use intelligence techniques to stay up-to-date with threats and prepare their systems for any eventuality.
- **Communications Security *CS*** (counters *CO*): The flow of information between internal and external system is a likely target for attack. In this group, we combine the following controls: Information Flow Enforcement *IFE*, Boundary Protection *BOP*, Continuous Monitoring *COM*, Cryptographic Protection *CRP*, and Information System Monitoring *ISM*.

We now map these controls to the following representative *defense actions*, which were automatically extracted (see Sect. 5 for details) from NIST's control catalog [17] by assessing their strength of association described through their official `Related Controls` property. The (official NIST control ID) and information system controls (*) are separately identified. Multiple mappings specify that several countermeasure classes and its related controls are effective in the respective scenario. The defense action listed below exemplarily includes related information system controls and control enhancements, which are finer-grained countermeasures within its context. Please refer to Appendix C for a full lost of control-to-action mappings.

The key component to finalizing the game model is the comprehensive mapping of simplified NIST families and controls (defense actions) to specific CAPEC mechanisms of attack (attack actions). See Sect. 5 for details about this process. In Table 7, we list some exemplary defense actions and their offensive counterparts.

With Alice's attack cycle complete, it is now time for Bob to deploy his countermeasures. As he earlier spotted an integrity attack targeting his web server, he assumes that Alice is trying to deface his institution's Internet presence. In response, Bob can now either implement a system-level 'data integrity' (DI) control for 1 Initiative point to counter the data manipulation (DM) attack, or an organization-wide control for 2 points. The latter would not complete within the same round, which poses too great a risk for Bob. He decides to do what he can and restore system integrity from 'highly affected' (2) to 'affected' (1). Depending on the game mode, he could either use an abstracted version of the COP, INH, CRP, MCP, or ISM controls, or implement related controls or control enhancements.

In this example, we use abstracted control enhancements: Bob checks each of the defense actions at his disposal and decides to use an enhancement of MCP (SI-3), namely SI-3(7): "Malicious code protection: Nonsignature-based detection". The default success chance of such actions is derived from twice his base Sophistication (6), Bob's Insight (+10%), and all equipment boosting either of the two. In our case, Bob is granted another point of Sophistication because of his packet filter, for a total defense success chance of 80%. Bob rolls a D10 and succeeds: The level of compromise of his web server is decreased by one increment to 'affected' (low) – a distinct setback for Alice.

The game will now continue until the attacker achieves her goal or runs out of Initiative, ultimately deciding the winner.

5 Data mapping

In this section, we specify the mapping of external data to the various classes and categories that are part of our model. This includes the formal link between actions and events, the mapping of the APT kill chain to CAPEC attack patterns, the CAPEC to CVSS mapping, and the association of controls to defense actions. Using below information, it is possible to easily extend or adapt the game system to new or updated scenarios in cyber-security and beyond.

5.1 Actions to events

Each action available in the PenQuest rule system can be modeled using the structure introduced in Sect. 3. The possible link between in-game actions and real-world events is an integral part of the model. We below exemplify this mapping using the star graph anomaly detection system that is at the heart of the AIDIS endpoint protection system [28]. The system is built for detecting and interpreting abnormal Windows OS behavior expressed through sequences of kernel events defined as $G = (U, V, E)$, where U and V are nodes (parametrized event type) and E is the respective edge (type of operation). Specific attacks can be learned by monitoring process activity and comparing it to a pre-established baseline of known process behavior. In a simplified fashion, a number of events contributing to an anomaly could look like the ones listed in Table 8.

To append the data to the model, we use the PenQuest game model (Sect. 3.2) to transform the list of events to a simple instance of the $\langle Event\rangle$ class of an action X, shortened here to two sequential events:

$$
\begin{aligned}
&Event = \langle\\
&\langle Type = Anomaly\rangle\\
&\langle Time\langle Start = 16.53.661, End = 16.53.729\rangle\rangle,\\
&\langle Score\langle Deviation = 112.4, Threshold = 16.0\rangle\rangle,\\
&\langle Sequence = 1\rangle,\\
&\langle Parent = \text{“}shell.exe''\rangle,\\
&\langle Operation = process\_start\rangle,\\
&\langle Argument = \text{“}drop.exe''\rangle\\
&\langle Sequence = 2\rangle,\\
&\langle Parent = \text{“}drop.exe''\rangle,\\
&\langle Operation = image\_load\rangle,\\
&\langle Argument = \text{“}library.dll''\rangle\rangle
\end{aligned}
$$

Since AIDIS classifies anomalies by their CAPEC attack pattern, the respective information can easily be added to the action definition:

$$
\begin{aligned}
&AttackPattern = \langle\\
&\langle Purpose\langle Exploitation = T, Obfuscation = F,\\
&\quad Penetration = T, Recon = F\rangle\rangle,\\
&\langle Impact\langle C = high, I = high, A = low\rangle\rangle,\\
&\langle ID = 207\rangle\rangle
\end{aligned}
$$

This `Purpose` information of CAPEC is then used to establish the link to our abstracted *primary attacks*, which is discussed below.

Table 6 Excerpt from the current pool of PenQuest attack actions

$\langle ID \rangle$*	Name*	Methods*	$\langle PatternClass \rangle$	$\langle Stage \rangle$	Kn.*	Purp.*	C*	I*	A*
100	Overflow Buffers	Analysis; Injection	IG; IN	R.I; R.S; E.I; E.L	1–3	Pen.; Expl.	3	3	3
103	Clickjacking	Spoofing; Social Eng.	IA; SE	D.I; I.P	3	Expl.	3	3	1
104	Cross Zone Scripting	Analysis; Injection	IG; IN	R.I; R.S; E.I; E.L	2	Expl.	3	3	3
105	HTTP Request Splitting	Proto. Man.; Analysis; Injection	DM; IG; IN	D.I, E.*; R.I; R.S	2	Expl.	2	2	1

Columns marked with asterisk (*) identify information mined from CAPEC attack patterns. 'Kn.' specifies the attacker knowledge required to perform the attack, ranging from low (1) to high (3). The C/I/A columns denote the respective impact of a successful use

Table 7 Excerpt from the current pool of PenQuest defense actions

ID*	Name*	$\langle Cat. \rangle$*	$\langle ControlClass \rangle$	$\langle ActionClass \rangle$	Related controls*	Control enh.*
AC-2	Account management	Org.	AC	ACM	AC-10, AU-9, IA-2, IA-8	AC-2 (1)..(13)
AC-3	Access enforcement	Sys.	AC	ACE	AU-9	AC-3 (1)..(10)
AC-3 (3)	Mandatory access control	Sys.	AC	ACE	AC-25, SC-11	n/a
SI-3	Malicious code protection	Org.	CP,SP, FI,DI	MCP	SC-26	SI-3 (1)..(10)

Columns marked with asterisk (*) identify information directly mined from NIST controls. Related controls only list information system level controls that are not in the primary category

Table 8 Example event sequence describing the process of disabling the Windows Firewall (CAPEC-207: "Removing important client functionality")

Start node (U)	End node (V)	Edge (E)
process-shell.exe	process-drop.exe	start (3)
process-drop.exe	image-library.dll	load (1.5)
process-drop.exe	registry-HKLM/Software/.../WindowsFirewall	open (0.25)
process-drop.exe	registry-DWORD(EnableFirewall=0)	add (0.75)

This simple interface makes it easy for analysts to add their own data to the game model. While our game rules use the CAPEC example, PenQuest remains flexible: By replacing the *AttackPattern* class definition, the level of abstraction and vocabulary can be freely specified – ranging from the discussed OS events to high-level behavior such as 'Set Fire to House'. See Sect. 6 for a full example in the context of a real IDS anomaly.

5.2 Kill chain to attack patterns

We use Hutchins et al.'s cyber kill chain [16] as foundation for the high-level view on our model. For further granularity, we expanded the 7 stages to a total of 19 subcategories that largely adhere to our APT ontology design introduced with TAON [27]. To link kill chain elements to CAPEC attack patterns, we introduced the concept of *primary attacks*, which were abstracted from CAPEC's `Purpose` classes and assigned attacked patterns to kill chain categories. Figure 8 depicts the mapping.

Since the list of attack patterns in CAPEC is partially incomplete or offers too little information in terms of abstraction required for a model, we only considered patterns of 'standard' and 'meta' abstraction level – the 'detailed' class was omitted in this initial iteration of the game. In addition, we opted to remove deprecated attacks and focus only on stable and draft patterns found in version 2.11 of CAPEC.[9] Incomplete patterns with no purpose classes were disregarded as well. Overall, this selection retained a total of 65 representative patterns out of 516. The remainder can easily be added at a later point.

5.3 Attack patterns to vulnerabilities

Vulnerabilities are a key component of any intrusion scenario. In order to automatically map CAPEC attack patterns to CVSS vulnerabilities, we took a closer look at these and other related MITRE information exchange standards. Ultimately, we use the `Related Weaknesses` information provided by CAPEC to map each pattern to specific weaknesses represented by the Common Weakness Enumeration (CWE) list. CWE *"provides a common language for describing security weaknesses in architecture, design, or code"*.[10] For example, CAPEC ID 1 ("Accessing Functionality Not Properly Constrained by ACLs") is related to CWE ID 276, 285, 434, etc.).

[9] https://capec.mitre.org.

[10] https://cwe.mitre.org/about/index.html.

To close the gap between weaknesses and the more concrete vulnerabilities, we use open source information to map CWE entries to their Common Vulnerabilities and Exposures (CVE) counterpart. For example, CWE-276 ("Incorrect Default Permissions") contains the vulnerabilities CVE-2005-1941, CVE-2002-1713, CVE-2001-1550, CVE-2002-1711, CVE-2002-1844, CVE-2001-0497, and CVE-1999-0426. Armed with this ID, the specific vulnerability can be looked up in the National Vulnerability Database (NVD[11]), where a specific CVSS score is assigned. This multipartite score is the basis for the requirements and impact calculations used in our model (see $\langle Enabler \rangle$ in Sect. 3). In our example, the score for CVE-2005-1941 can be easily retrieved.[12]

5.4 Primary controls to defense actions

The mapping of controls to defense actions and primary controls was automatically computed from the NIST SP 800-53r4 standard [17]. To achieve this, we extracted all links of relationship between the individual controls and modeled them as a graph. We subsequently separated each control into one of three categories by their level of degree. Controls with a degree $d >= 20$ were defined as parent *defense actions* (see Fig. 8). Controls with degree $d < 20$ and $d >= 1$ form the bulk of related controls which operate on information system and/or organization level and translate to the remainder of the defense strategy set for the respective parent action. The model can be further extended by adding control enhancements that, in turn, are associated with numerous parent and related controls. Figure 9 depicts the relationship between all 224 NIST controls. General measures (NIST suffix *-1) were converted into policies, which are part of the $\langle Disablers \rangle$ class (see Sect. 4.2.1 for a complete list).

For the mapping between defense actions and primary controls, a natural language approach has been investigated. By combining four IT and information security glossaries (Gartner,[13] Techopedia,[14] NIST-IR 7298 [19], and the North Carolina statewide glossary of information technology terms[15]) into a list of reverse stopwords (i.e. the remainder of words were removed from the corpus), we attempted to automatically link CAPEC pattern and NIST control descriptions using Quanteda for R [3]. However, the difference in terminology and writing style rendered the approach too inaccurate for practical implementation in this particular case. Instead, we opted to create primary controls (see Sect. 4.5.3) and directly assign them to the aforediscussed defense actions. Figure 8 depicts the complete mapping.

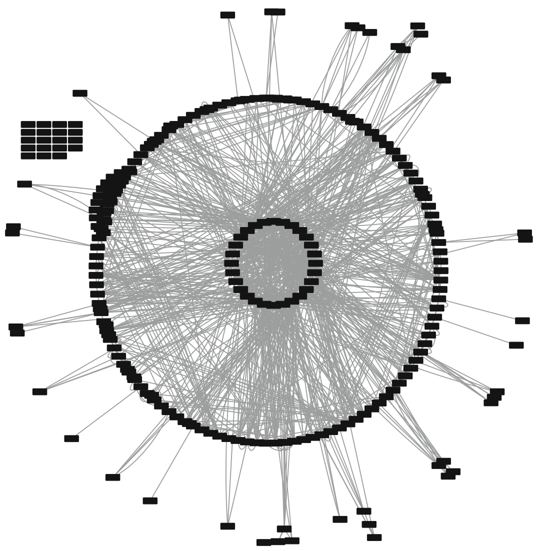


Fig. 9 Relations between NIST controls rendered in Cytoscape [49]. The inner circle represents controls with a degree of ≥ 20. Elements beyond the outer circle have a degree of 1, while the isolated nodes to the left are not linked to any other controls (orphan controls)

6 Preliminary evaluation

In this section, we briefly introduce our first physical prototype of the PenQuest RPG and present the quantitative and qualitative findings of our initial test games designed to determine the suitability of the gamified model for (awareness) education and threat explication. We also take a closer look at data-to-model mapping as well as strategy set distribution between the attacker and defender to measure the model's completeness. The discussion about future scenario simulation and IT-enabled automation can be found in Sect. 7.

6.1 Experimental setup

6.1.1 Prototype

Our first operational prototype uses a physical approach to present the game and its components to a predominantly information security audience. Actions and equipment are designed as cards containing all necessary information ranging from categories, requirements, and game impact. Progress tracking for asset compromise, APT kill chain traversal, and success modifiers (e.g. Insight, success chance) is realized through physical tokens placed on a printed game board. Similarly, levels of compromise, actor attributes, credits, and attack objectives are tracked using a combination of cards and tokens. Figure 10 depicts an early version of

11 https://nvd.nist.gov.

12 https://nvd.nist.gov/vuln/detail/CVE-2005-1941.

13 https://www.gartner.com/it-glossary/.

14 https://www.techopedia.com/dictionary.

15 https://it.nc.gov/document/statewide-glossary-information-technology-terms.

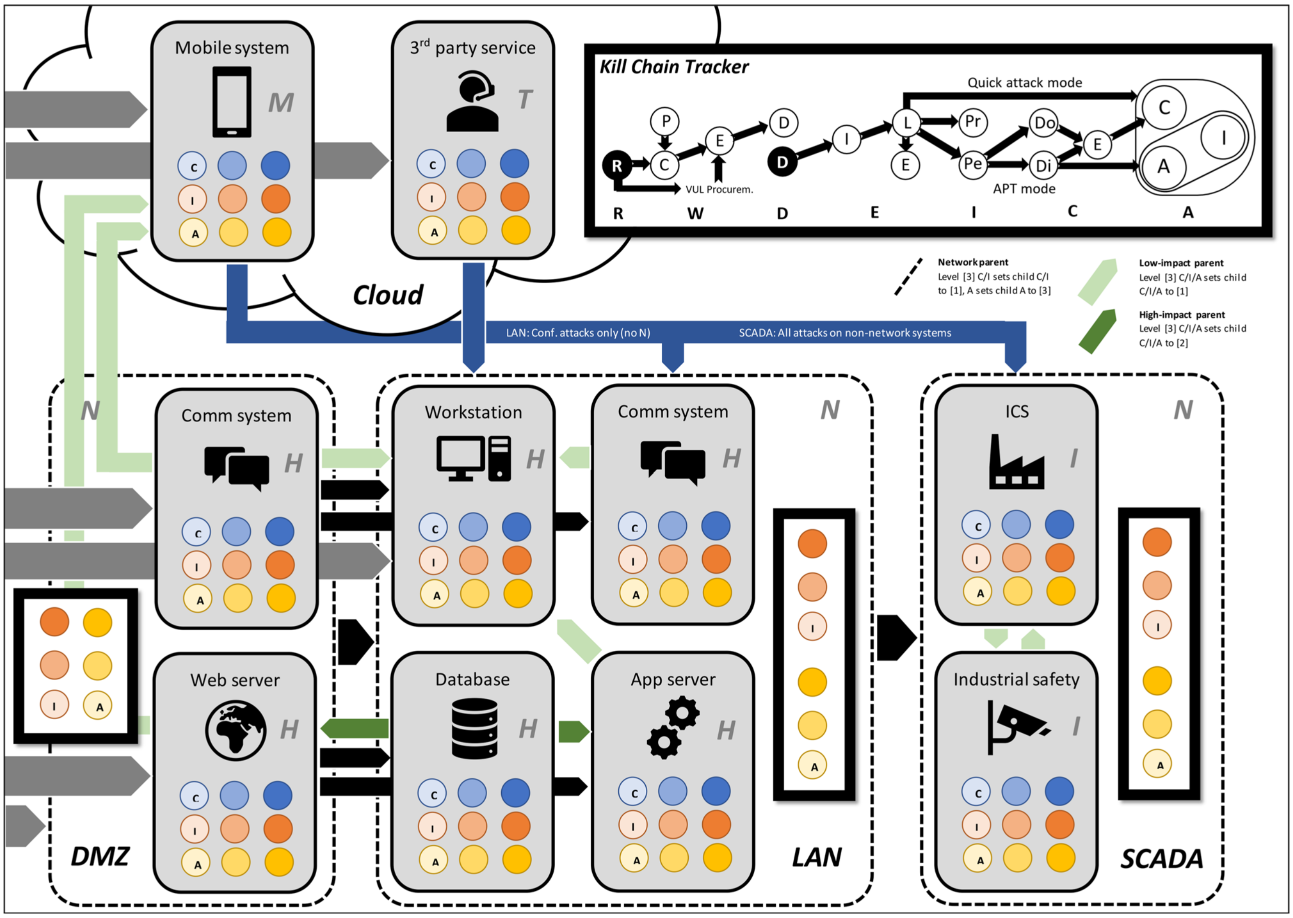


Fig. 10 Prototype game board used by players. Current (known) levels of compromise, the progress along the kill chain, and the attacker's primary goal are marked here. Current Insight and Initiative is tracked using tokens. The remainder of the game utilizes a total of 359 cards

the game board. Next to visualizing attack vectors and asset dependencies, it provides players with a kill chain tracker for planning their attack and defense.

A total of 100 attack actions and 70 defense actions were prepared for the test games. Existing CAPEC patterns and NIST control enhancements were complemented by a small number of placeholder actions for the weaponization and C2 phases of the APT kill chain, which are not currently covered by CAPEC. In addition, we designed 50 unique attack tools (enablers) and a total of 58 disablers, 18 of which represent organization-wide policies. This was complemented by a set of 25 representative vulnerability exploits and a total of 18 fixes substituting the CVE component of the game. In future automated simulation games that do not need to meet the same level of accessibility requirements, vulnerabilities will be retrieved directly from the NVD.

IT support for the current implementation already exists in the form of the WF-net validator introduced in Sect. 3.3 as well as a PoC prototype for a two-player browser app that incorporates actor creation and basic game board interaction. This does not mean that PenQuest's physical iteration is limited in terms of overall functionality, however: the game simply requires a human moderator to enforce some of its more advanced rules and keep track of point tallies – akin to many commercial strategy games and moderated red-team exercises [43].

6.1.2 Questionnaire

In order to evaluate the physical game prototype and the model's suitability for representing APTs we invited a number of test persons from a corporate and educational IT security background and presented them with both an introductory and a concluding questionnaire. Next to age, gender, and specific level of security expertise we asked participants to answer a number of simple questions about IT security topics to score their current level of knowledge. Answers to the following questions were ranked from 'strongly disagree' (0 points), 'rather disagree' (1 point), 'rather agree' (2 points), to 'strongly agree' (3 points):

1. I know how cyber-attacks typically play out

2. I am familiar with the APT kill chain
3. I am familiar with the NVD and its scoring system (CVSS)
4. I am familiar with the concept of IT system vulnerabilities
5. I am familiar with CAPEC or similar attack pattern schemas
6. I am familiar with standards like ISO 2700x and NIST 800-53
7. I am familiar with information security best practices
8. I am familiar with IT network topology
9. I am familiar with common types of malware
10. I am familiar with the functionality of common hacking tools
11. I know how an organization can defend against cyber-attacks

These questions were repeated after one play-through in order to evaluate lessons learned. In addition, the subjects graded the game's suitability for education and threat modeling in e.g. risk assessment scenarios. Specifically, we asked questions about accessibility (learning curve, handling of the prototype, abstraction level of attack and defense actions), realism (applicability to real-world scenarios, scope of actions/equipment), game balance (attacker/defender equilibrium in same-Sophistication games), and awareness benefit (personal takeaway, educational effect). For threat modeling, subjects were asked to rank the model's suitability for awareness building and threat representation (abstraction level, real-world application, incident representation). All games were documented to protocol action dynamics and session outcome. Note that testers playing more than one game were assessed only once and that none of them were involved in the creation of PenQuest.

6.1.3 Expert interviews

All participants of at least 'professional' IT security level (see demographics below) were also given the opportunity to provide individual textual and face-to-face feedback. While general comments were encouraged, we asked the remaining 7 practitioners to focus on criticism in the areas of game accessibility, balance, and design. To assess the underlying model, we also provided them with a copy of PenQuest's full documentation (base, game, and rule model) as well as with the game's ruleset in scientific and instruction guide format and requested specific feedback on each of the aspects. The results of these interviews are part of the qualitative evaluation below.

6.1.4 IDS data

To demonstrate IDS data-to-model mapping, we used classified anomaly events of the AIDIS endpoint intrusion detection system [28] as input. The system provides automated classification of anomalies as belonging to a specific CAPEC pattern, through which it becomes possible to directly link extracted behavioral data to the PenQuest meta model for further semantic enrichment, interpretation, and mitigation planning.

For evaluative demonstration, we use the CAPEC class with the most events while boasting a low misclassification rate, as disseminated in [28]. We specifically regarded CAPEC-112,[16] 'Brute Force', with a total number of 380 process anomaly reports and a misclassification rate of 0% in the context of AIDIS's operation. The concrete mapping is discussed as part of the qualitative evaluation below.

6.2 Quantitative results

In this subsection, we quantitatively evaluate the test games conducted with a number voluntary participants. While the number of players is arguably limited, the results still help to get an impression for the practical applicability of PenQuest's game component in awareness building scenarios. In addition, we enumerate the rule system itself to assess its coverage of various attack categories.

6.2.1 Test games

Of the 8 full test games played (in addition to approx. 12 partial sessions), 5 ended in victory for the attacker. The average session lasted for 7 Initiative rounds. Initial game setup and the first-time explanation of the rules to the participants new to PenQuest took an average of 45 minutes, while the actual playing session concluded in around 115 minutes. Stealth phase lasted an average of 1.88 rounds. On the kill chain, attacking players rarely exceeded the Exploitation ($E.*$) stage that marks the conclusion of a 'quick mode' game. In terms of scenario, the play sessions i.a. encompassed 'Operative vs. Military', 'Protester vs. Education', 'Infiltrator vs. Infrastructure', 'Thief vs. Manufacturing', and 'Raider vs. Services Sector' scenarios.

Demographically, most players (7) of the total 9 were male, aged 26 to 35 years. Dominant occupations were university lecturer/researcher, company employee (5 and 3 participants respectively), and 1 student, while the level of knowledge was evenly spread between 'Operator/Specialist' and 'Executive/Management' (4 and 4). One 'Assistant/Intern' participated in the testing. Occupation-wise, the I(C)T/ informatics sector was represented the most (7 participants), followed by marketing/media (1) and general education (1). All of the players claimed to have at least intermediate experience in the area of IT security. Of the 9 participants, 4 ranked themselves as experts (identified as $Exp*$ in Table 9), 3 as

[16] https://capec.mitre.org/data/definitions/112.html.

Table 9 Total knowledge gain per participating player

Qn	Int1			Int2			Pro1			Pro2			Pro3			Exp1			Exp2			Exp3			Exp4			**Mean**
	B	A	+	B	A	+	B	A	+	B	A	+	B	A	+	B	A	+	B	A	+	B	A	+	B	A	+	
Q1	1	2	1	1	2	1	2	2	0	2	2	0	2	2	0	3	3	0	2	3	1	2	2	0	2	2	0	0.33
Q2	0	2	2	0	1	1	2	3	1	1	2	1	1	2	1	2	2	0	2	3	1	3	3	0	1	2	1	0.89
Q3	0	0	0	0	1	1	2	2	0	0	1	1	1	2	1	1	1	0	1	0	-1	2	2	0	3	3	0	0.22
Q4	0	1	1	0	1	1	2	2	0	3	3	0	2	2	0	3	3	0	3	3	0	2	3	1	3	3	0	0.33
Q4	0	2	2	0	1	1	1	2	1	1	1	0	1	2	1	0	0	0	0	0	0	2	3	1	2	2	0	0.67
Q5	0	1	1	0	1	1	2	2	0	2	2	0	3	3	0	1	1	0	0	0	0	3	3	0	1	2	1	0.33
Q6	1	2	1	1	2	1	2	2	0	3	3	0	3	3	0	3	3	0	2	3	1	3	3	0	2	2	0	0.33
Q7	0	2	2	0	1	1	2	2	0	3	3	0	2	2	0	3	3	0	2	3	1	2	2	0	3	3	0	0.44
Q8	2	3	1	1	2	1	2	2	0	2	2	0	2	2	0	3	3	0	2	3	1	3	3	0	3	3	0	0.33
Q9	0	1	1	0	1	1	2	2	0	1	1	0	1	2	1	3	3	0	2	3	1	2	2	0	3	3	0	0.44
Q10	1	1	0	0	1	1	2	2	0	2	2	0	2	2	0	3	3	0	2	2	0	3	3	0	3	3	0	0.11
Sum			12			11			2			2			4			0			5			2			2	4.44

The score is derived from a self-assessment conducted once before playing PenQuest (B) and a second time thereafter (A). There are 4 answer categories per question, ranging from 'strongly disagree' (0) to 'strongly agree' (+3). Each point of improvement ('+' column) represents the knowledge gain of the individual player, ranging from 0 to 3. The sum in the lowermost row designates the overall knowledge gain per participant

professionals ($Pro*$), and 2 as intermediate ($Int*$). Most players (5) did not have prior experience with learning games. Only one person designated herself as moderately experienced in serious gaming.

According to the returned questionnaires, the greatest learning effect was achieved in the areas of the APT kill chain, CAPEC attack patterns, network topology, and the functionality of hacking tools. Interestingly, the area of CAPEC was also the one where many participants (4 out of 9) showed no knowledge gain after playing the game. This is likely owed to the fact that most games were not contentually moderated – most of the time, the game master focused on explaining mechanisms and rules instead of information security aspects. Either way, understanding attack patterns has been identified as an area in high need of impartation beyond the single line of explanatory text currently printed on the card.

Non-experts benefited the most from playing the game, reporting the highest gain in domain knowledge. Experts still claimed a minor increase in topical insight in 3 out of 4 cases. See Table 9 for an overview.

When asked to grade their personal experience, all the players stated that it was overall positive. All participants agreed that the game showed significant promise for educational use in a higher education or company environment. The question as to whether the "Game/model is a suitable tool for security education" was answered positively most often, with an average score of 2.78 out of 3, followed by the statements "Game is fun to play" (2.56), "Game increases general security awareness" (2.44), and "Model accurately represents real-world security incidents" (2.44). In terms of learning curve, answers were the most controversial (1.56 out of 3), indicating a need to further improve accessibility for players unfamiliar with serious games. Figure 11 depicts the results of the final questionnaire.

6.2.2 Strategy set distribution

The distribution of actions between the attack and defense domains is both an indicator for control effectiveness (where few controls counter a large number of attacks) and model completeness. In Fig. 12, we show the distribution of attack/defense actions per primary attack/control. In the current prototype, state attacks (SA) were identified as underrepresented due to a lack of CAPEC patterns meeting the requirements specified in Sect. 5.2. This can be partially remedied by adding 'detailed'-level controls to the model.

In terms of APT kill chain coverage of the game, the CIA triangle is represented with a nearly identical number (60 to 66 each) of available attacks corresponding to the three information security factors. With the exception of availability attacks where 'low'-rated attacks are predominant, most of CAPEC's utilized patterns describe 'high' impact attacks, followed by 'medium' attacks. Stage-wise, most actions (120) belong to the Exploitation phase, with around 45 patterns linked to Delivery and Reconnaissance. The remainder of the kill chain is represented by everything between 16 and 39 CAPEC and placeholder pattern attacks. See Fig. 13 for a full breakdown.

Figure 14 quantifies the number of attack patterns per defense action category. Unlike the strategy set distribution, these numbers encompass all currently available categories as per the data mapping specified in Sects. 5.2 and 5.4. Here, the number of attack patterns countered by each defense action is largely well distributed. Media Storage

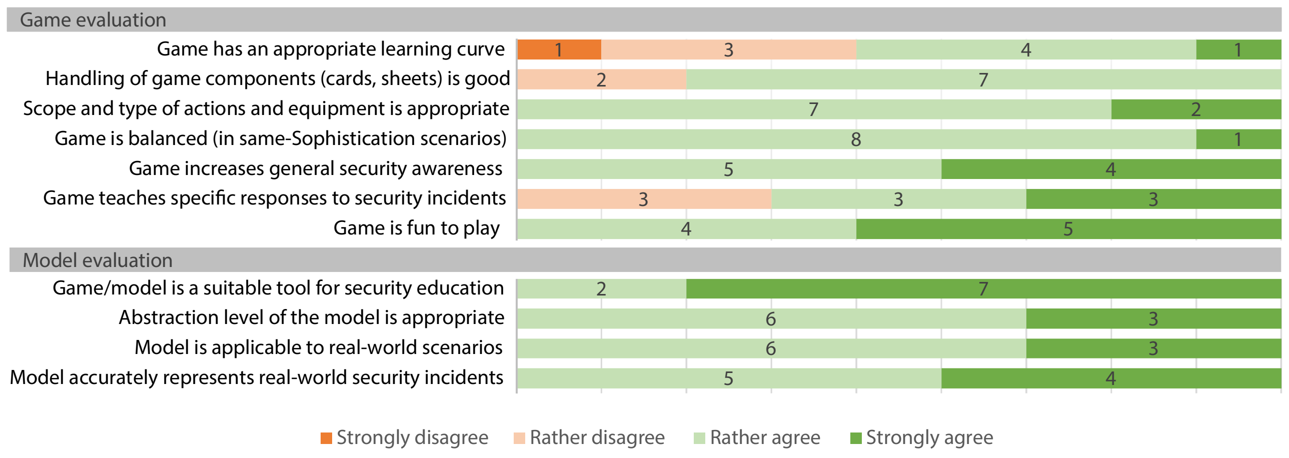


Fig. 11 Player feedback by question, ranging from 'strongly disagree' to 'strongly agree'. While most players would use the game as part of an awareness program, almost half of them find the learning curve to be rather steep

Fig. 12 Distribution of attack/defense actions per primary attack (black) and primary control (gray). Some attack actions (such as on-premises weaponization (PR)) do not have counterparts

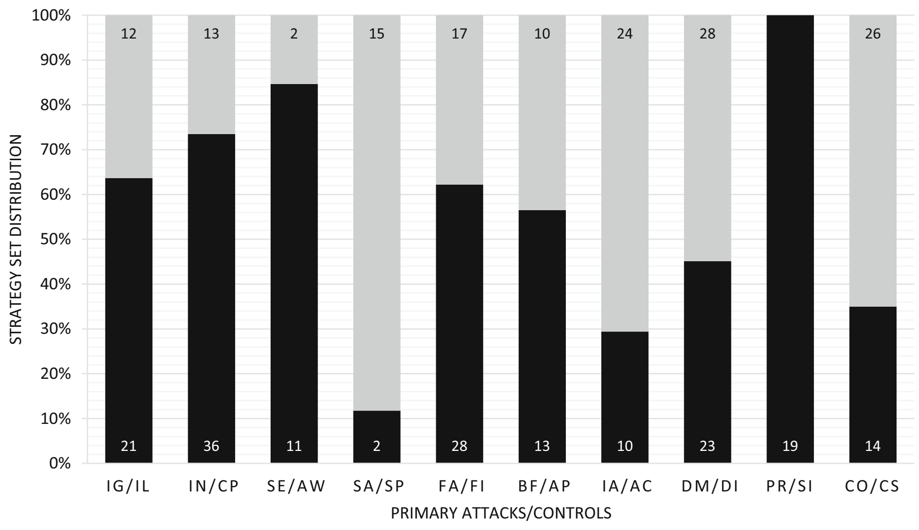
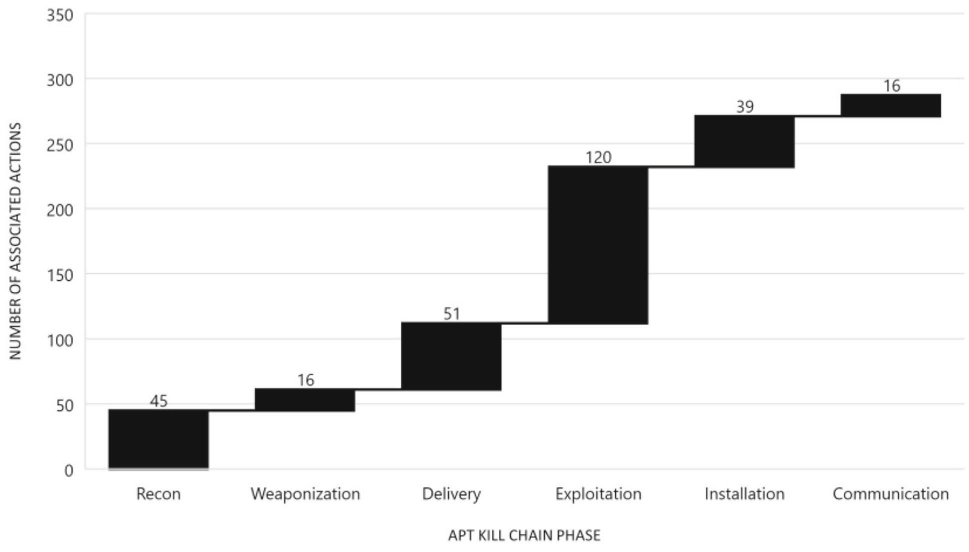


Fig. 13 APT kill chain coverage through available attack actions. The 'Action on Objective' stage is integrated into the other categories by means of the game model



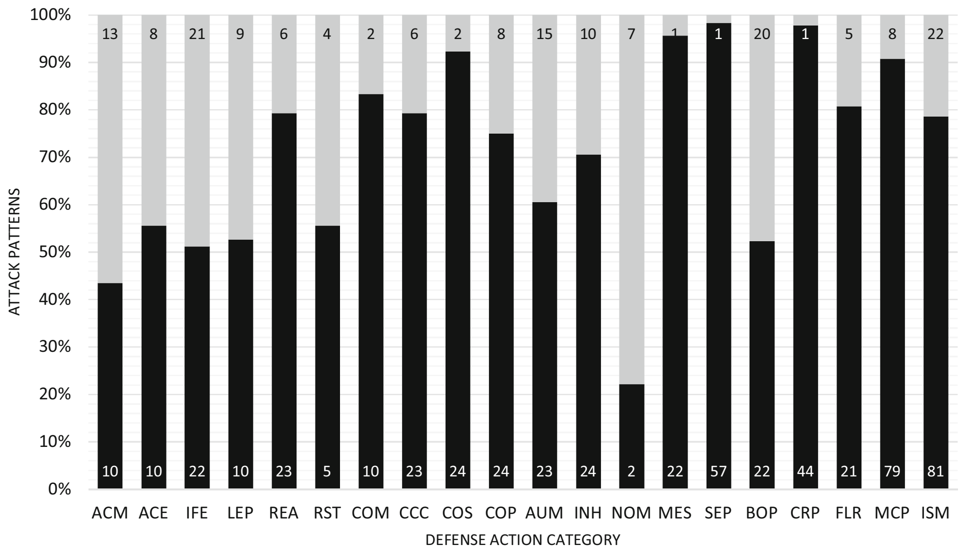


Fig. 14 Attack patterns (black) per defense action category (gray). This chart encompasses all actions and categories currently available for game development

(MES), Security Engineering Principles (SEP), and Cryptographic Protection (CRP) controls are not enhanced by more granular NIST countermeasures, making them particularly effective if employed at organization level. At the same time, there are only few attack patterns that can be countered by Nonlocal Maintenance (NOM), which is less concerned with deliberate attacks and more with system upkeep.

The distribution of the categories clearly shows where current standards offer fewer, if broader – and potentially more effective – guidelines in terms of mitigation. For many technical attacks NIST suggests only a few countermeasures without going into detail with extended controls. Next to above examples, this includes anti-malware solutions, configuration management, and (software) flaw remediation. Similarly, attack patterns are not equally easy to come by and tend to describe some types of attacks over others. Especially system state manipulation, spoofing, and the subversion of access control mechanisms are harder to specify in significant quantity than e.g. injection attacks.

6.3 Qualitative results

When asked to provide additional textual and oral feedback, security practitioners contributed valuable insight and suggested a number of improvements to both model and game. A summary can be found below. Lastly, we sketch how the mapping of IDS data to the model works in practice.

6.3.1 Game

In stage 1, we collected feedback aimed directly at the game and its prototype implementation. This information is particularly important for the ongoing development of PenQuest's digital iteration and primarily addresses three topics of interest: accessibility, balance, and design.

Accessibility – One of the major points raised in the expert discussion was that the current prototype aims at a target audience with at least a Bachelor-level degree in computer science or IT security. While terms and explanations used in the game correspond to the current business practice, the entry level for students or employees of other areas is too high. Several participants suggested that future versions of PenQuest – provided they also seek to target non-IT players – should paraphrase security concepts in simpler language and provide the original information as footnote pointing to an external reference.

It was also noted by a tester that people unfamiliar with complex strategy games might be overwhelmed by the amount of rules. In contrast, participants who plays board games regularly described PenQuest's rule set as "about average in difficulty" and added that it is common for such games to require 2 or more playthroughs to get a full grasp of the more advanced mechanics. Most of the interviewed stated that having a digital system for calculating system compromise and success probabilities would be beneficial in terms of accessibility.

Balance – Tester feedback was mostly in line with the results of our quantitative evaluation of PenQuest's strategy set distribution. While some actions were perceived as more effective than others, there were no trump cards that enabled an easy victory. Without exception, the participants agreed that the slight bias towards the attacking actor is both realistic and entertaining – and helps transport the message of awareness.

The majority of experts showed themselves surprised that even though the model uses real-world data sources for its loss and gain scores, the resulting game is naturally balanced. It was added, however, that this equilibrium would likely be lost without the limiting factors of Initiative and Wealth, which temporally and financially constrain the modeled adversarial campaign.

Design – Since the visual design of the game is not the focus of this article, we only briefly summarize tester feedback: While most game elements were deemed as practical and comprehensible, players suggested various improvements to iconography and game board design. In particular the CIA impact of attack actions and the representation of the APT kill chain were criticized as not immediately self-explanatory.

6.3.2 Model

The second stage of the qualitative feedback process revolved around the model on which the game is built. After playing at least one test game and having been given a detailed introduction to the base, game, and rule models as well as the game's resulting ruleset, participants were asked to provide criticism from their perspective as experienced security practitioners.

Base model – Testers agreed that PenQuest's base model serves well to introduce the core concepts of the gamified system and defines all terms needed to understand layer and component interplay. It was suggested, however, that future iterations of the model could benefit from additional detail, especially for the 'knowledge' and 'configuration' aspects found within the model's information layer. One expert recommended that the concept of vulnerabilities could be incorporated directly into the base model, yet admitted that detaching them from the 'equipment' definition would add an undesired layer of complexity.

Game model – Generally, PenQuest's use of attack and defense classes to abstract attack patterns and controls was lauded by the testers. Most experts emphasized that the approach helps to better understand the model's mapping procedures and enables security practitioners to plan an appropriate defense against threats even without understanding all the specifics of each individual attack or control.

Scrutinizing feedback included the aforementioned vulnerabilities and fixes, which are currently part of the $\langle Enabler\rangle$ and $\langle Disabler\rangle$ classes of action X. It was suggested by an expert to consider creating a class of its own for both, something that will be investigated for future versions of the model.

When discussing the game model's $\langle Event\rangle$ class, several of the participants independently confirmed that it provides the means to interface PenQuest to not only IDS data, but to established threat intelligence languages as well. This link and the construction of an ontology based on our meta model will be thoroughly explored in future research.

Rule model – All participants agreed that the game's core principles aptly capture the essence of the problem. The zero-sum component for tallying system compromise was deemed a suitable abstraction for the process of attacking an asset with a certain goal in mind.

Game rules – The discussion of the rules resulting from the above model components took up most of the allotted interview time. Testers predominantly agreed that the rules are a good compromise between a realistic simulation and an awareness-building game that is also entertaining its players. Feedback mostly related to future work and additions to the game, which included awarding situational modifiers depending on which $\langle AttackActor\langle Motivation\rangle\rangle$ the attacking player has chosen. It was stated that PenQuest would also benefit from replacing generic equipment with real-world appliances and tools, while another expert noted that such a move would be akin to advertising and should be considered carefully.

Several testers were enthused by the fact that the current, abstracted topology could be easily expanded to mirror more complex networks. At the same time, they suggested that threat simulation based on PenQuest would benefit from a formalized 'topology creation mechanism' that helps prevent design errors in terms of attack vector and asset dependency. Two participants suggested the use of a more streamlined version of the APT kill chain for modeling the typical sequence of attack actions. While its current level of detail was appreciated and deemed useful for information security audiences, the testers argued that a simpler representation without substages would be beneficial for the average player.

6.3.3 Data mapping

Concluding the qualitative evaluation, we take a look a specific use case where data captured by an IDS is mapped to the PenQuest model for semantic enrichment and mitigation planning. For this purpose, we first use PenQuest's $\langle Event\langle Type = Anomaly\rangle\rangle$ notation of X, as specified in Sects. 3.2 and 5. The exemplary event sequence extracted by AIDIS [28] has a time range of $\langle Time\langle Start = 0, End = 10\rangle\rangle$. As seen below, each $\langle Operation\rangle$+$\langle Argument\rangle$ pair with $\langle Parent = svchost.exe\rangle$ is appended in sequence.

$$\begin{aligned}
&Event = \langle\\
&\langle Type = Anomaly\rangle\\
&\langle Time\langle Start = 0, End = 10\rangle\rangle,\\
&\langle Score\langle Deviation = 62.7, Threshold = 45.8\rangle\rangle,\\
&\langle Sequence = 1\rangle,\\
&\langle Parent = svchost.exe\rangle,\\
&\langle Operation = image\_load\rangle,\\
&\langle Argument = advapi32.dll\rangle
\end{aligned}$$

$\langle Sequence = 2\rangle,$
$\langle Parent = svchost.exe\rangle,$
$\langle Operation = image\_load\rangle,$
$\langle Argument = cfgmgr32.dll\rangle$
$\ldots$
$\langle Sequence = 30\rangle,$
$\langle Parent = svchost.exe\rangle,$
$\langle Operation = registry\_modify\rangle,$
$\langle Argument = /machine/system/controlset001/ services/wbiosrvc/\ldots\rangle\rangle$

The result is a simple description of X that can be easily converted to other threat definition languages or shared directly with others. In our particular case, X represents an anomalous deviation tagged as 'CAPEC-112' by AIDIS. The link to CAPEC provides us with additional semantic information, namely that the corresponding 'Brute Force' label refers to activity where the "attacker attempts to gain access to this asset by using trial-and-error to exhaustively explore all the possible secret values in the hope of finding the secret (or a value that is functionally equivalent) that will unlock the asset." [32]

According to the meta model summarized in Fig. 7, CAPEC-112 can be an APT kill chain 'Delivery–Intrusion' as well as an 'Installation–Propagation' or 'Installation–Persistence' support action which is typically used in combination with other attack activity. Categorized as the identically named BF (Brute Force) attack action, PenQuest also defines appropriate primary controls countering the threat, namely 'Authentication protection' (AP): This controls group is primarily concerned with managing authenticators such as passwords, tokens, and biometric information. Associated defense actions include the categories 'Remote Access' (REA) and 'Authenticator Management' (AUM), with a range of controls directly out of NIST SP 800-53 [17]. Specific countermeasures therefore include remote access control and encryption, access point management, password/PKI/hardware/biometric authentication, as well as controls related to cache expiration settings.

The information gleaned from the model can now be used to plan appropriate defensive measures to prevent this particular attack. PenQuest's gamified nature also allows us to play through the attack and test various controls and systems that may reduce threat impact and probability. While the efficacy of the suggested countermeasures need to be evaluated on a case-by-case basis using real world infrastructure or larger-scale expert interviews, the interviewed security practitioners agree that PenQuest is 'applicable to real world scenarios' and that the example mapping makes sense in terms of threat–mitigation pairing.

7 Discussion

In this section, we reiterate key features and briefly discuss the implications arising from the evaluation. Future enhancements that go beyond the current implementation are highlighted. In conclusion, we discuss the drawbacks of the current variant of the game and talk about future research.

7.1 Features

One of the key aspects of the model is its support for new, hitherto neglected use cases that go beyond the hacking scenario described in this paper. The general structure of classes presented in Sect. 3.2 is fully compatible with user-side changes to their content. New actors, motivations, enablers and disablers, effects, and attack/defense classes can simply be replaced or amended with low to medium effort, and without any alterations to the core rules. Information derived from external sources (primarily attack patterns and controls) can be changed by applying the introduced data mapping mechanisms – be it semi-automated assignment using intermediate abstraction levels or a natural language approach such as the one sketched in Sect. 5.4.

The network topology introduced in this article – while based on STIX' `VictimTargetingType` TPP schema – remains flexible as well. As long as a attack vectors and dependencies are maintained, game masters/designers can add or remove assets as they see fit. In simulation scenarios it is actually encouraged to model the topology after the real-world system chosen for assessment instead of using PenQuest's default structure. While the process of creating a custom topology is not currently formalized in the rule system, it will be added in the near future to minimize human error.

Assembling new scenarios from the list of actors and assets is a matter of computing all possible actor–goal combinations. The same is significantly harder to do for the game itself due to the combinatorial challenge of pitting each actor, action, enabler, and disabler against one another. It is currently more feasible to use PenQuest to explore an exemplary infrastructure and test various likely attack/defense scenarios in the course of one or several playthroughs. Still, automatically simulating a large number of attack cases is a valid approach to deriving ideal strategies. Such a simulator is one of the major contributions planned for future research and will investigate the utilization of both model checking [7] and reinforcement learning [18].

7.2 Limitations

There are a number of limitations that need to be considered before employing the prototype version of PenQuest for attack analysis, risk assessment, or simply as an awareness game. Most of these factors also offer opportunities for future research.

Firstly, one needs to keep in mind that PenQuest currently exists as model with a physical prototype implementation only. While the zero-sum component of system compromise has been realized programmatically (see Sect. 3.3), it is not yet integrated into a fully featured app. This impacts scalability and makes it difficult to effectively reenact large APT campaigns in a sensible amount of time. Ongoing efforts to remedy this limitation include an educational two-player web application expected to be released in 2020. Despite the lack of automation, all modeling and mapping tasks are fully fleshed out and ready for use through cross-referenced tables and the physical game prototype itself. The limited automation and the length of an average game session make the current version of PenQuest best suited for moderated workshops and special lectures.

Another limiting factor pertains the use of external data sources and threat intelligence: Attack patterns may outdate due to a lack of database maintenance and novel threats might not be considered immediately after disclosure. For CAPEC specifically, there are patterns that lack CIA impact or purpose information, e.g. CAPEC-2: "Inducing account lockout". This might make it necessary for the player to expand the repository of attack strategies/actions with their own information.

The assignment of primary controls to defense actions (see Fig. 8) is currently done semi-manually by a group of IT security experts. Automated mechanisms have proven to be too inaccurate during initial experiments, which utilized a natural language approach using several IT security glossaries. Because of this limitation, the attack–defense mapping of data sources that are significantly larger than the NIST standard might not be feasible. Another minor drawback of the NIST control approach is the small number (19) of orphan controls, which are not related to any other countermeasures. Currently, these controls are not considered in the game, since they would have to be manually added and assigned a specific category, levering out most mechanisms of automation currently in place.

On the CAPEC side, we do not restrict attacks to a specific target type like we do for equipment, since CAPEC does not offer a clean way to assign systems (hosts, network, industrial components, etc.) to a particular pattern. That means that e.g. the BF primary attack such as CAPEC-49: "Password Brute Forcing" can, in the game, be used on an arbitrary victim without further distinction into target categories. This limitation might in some cases simplify a hostile action at the expense of realism. Countering this drawback is possible, but currently requires manual intervention: Depending on the description of the respective attack pattern, the information provided in CAPEC's database (namely the `Summary` and `Example Instances` columns) can be parsed and assigned one of the equipment type categories used for assets ($\langle EffectTarget\rangle$).

Modeling-wise, not all of the 517 CAPEC classes and only a portion of the defensive controls are currently part of PenQuest. Around 12% of the available patterns have been used to populate the model to date, whereas 'detailed' technical patterns referring to specific software attacks (as opposed to 'meta' and 'standard') are not currently included. Control-wise, we prototypically implemented 70 out of 224 controls specified by NIST. With the model itself ready for use, the remainder of the data can be added at will. However, it needs to be stated that the CAPEC repository itself is missing some of the required information needed for automated mapping, which increases the effort required to add the remaining patterns. For future iterations, we will therefore consider alternative vocabularies such as MITRE ATT&CK.[17]

Lastly, the limits of the preliminary evaluation itself need to be mentioned: With the relatively small pool of test persons, representative quantitative assessment of the game and model has proven to be difficult. Our current prototype confines experiments to on-site play-testing, and the number of security experts available for evaluating PenQuest's base, game, and rule models who were not involved in the creation of PenQuest is similarly limited – mostly due to geographic constraints and the time required to explain, understand, and assess such a complex model. For this reason, large-scale evaluation has been postponed until the full digital iteration of the game is available, which will eliminate the need for test players to be physically present.

7.3 Future work

In addition to remediating above limitations, additional future work is planned. First and foremost, a browser-based iteration of the prototype is currently in the pipeline. Such an app will automate the computation of scores and success rates, thereby eliminating much of the game's complexity.

The mapping of events captured by intrusion detection systems to the model is one of PenQuest's key features. To further automate this process, future work is planned to include the conversion of anomaly data from the basic $\langle Event\rangle$ format to established threat intelligence languages such as STIX: Individual events could be translated to STIX Cyber Observable Objects,[18] corresponding to various file system activity. As de-facto standard for describing threat intelligence [47], STIX includes a wide range of observables that are generally similar to the events defined by our system. There is still work to be done, however: PenQuest's seman-

[17] https://attack.mitre.org/.

[18] http://docs.oasis-open.org/cti/stix/v2.0/cs01/part4-cyber-observable-objects/stix-v2.0-cs01-part4-cyber-observable-objects.html.

tic annotations will require additional adaptation of existing STIX objects, likely the ones used to describe attack patterns, campaigns, and action relationships.[19] Integrating our system into the data models of intelligence languages will generally help to improve threat information sharing between organizations.

New scenarios and data providers beyond the introduced network-centric application will be added to harness the full potential of the model. Future research will investigate how PenQuest can be adapted to other domains such as physical, industrial, or public security and safety. Ultimately, it is planned to create a simulation app that will allow us to automatically compute hitherto unseen attack campaigns and identify systemic weaknesses in arbitrary infrastructures.

8 Conclusion

We have presented the design of a gamified meta model that can be used to train personnel, assess risk mitigation strategies, and compute new attacker/defender scenarios in abstracted (IT) infrastructures. While the model itself is extremely flexible and supports varying levels of granularity, the initial game prototype design based upon this model utilizes accepted taxonomies and security standards to support out-of-the-box organization-level gameplay for simulating cyber-attacks on various types of local or networked assets. Our data mapping mechanisms enable domain experts to easily extend the system with new actors, actions, and (mitigating) equipment. We also exemplified how real-world data such as OS kernel events can be linked to the model.

The development of the first game prototype based on the introduced model has been completed. Initial evaluation by security professionals has yielded that PenQuest shows significant promise for deployment in education and that it aptly captures real-world threats to IT infrastructures.

The gamified model also offers a solid foundation for the development of an ontology for targeted attacks, which can be populated with both threat intelligence sources as well as host and network monitoring data. This link to applied anomaly detection and interpretation will be further explored in conjunction with intrusion detection systems such as our previously proposed threat detection and explication system, which uses multi-class classification to label anomalous process behavior [28]. By bringing the worlds of awareness, mitigation planning, monitoring data, and threat modeling closer together, PenQuest significantly aids in understanding and closing the semantic gap – while allowing people to learn about information security in an entertaining fashion.

[19] http://docs.oasis-open.org/cti/stix/v2.0/cs01/part2-stix-objects/stix-v2.0-cs01-part2-stix-objects.html.

Acknowledgements Open access funding provided by FH St. Pölten - University of Applied Sciences. The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

Appendix

Appendix A: List of acronyms

General glossary, alphabetically sorted:

APT	Advanced Persistent Threat
C2	Command and Control
CAPEC	Common Attack Pattern Enumeration and Classification
CIA	Confidentiality, Integrity, Availability
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerabilities Scoring System
CWE	Common Weakness Enumeration
DMZ	De-Militarized Zone
ICS	Industrial Control System
ICT	Information and Communications Technology
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
LAN	Local Area Network
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OS	Operating System
RPG	Roleplaying Game
SCADA	Supervisory Control and Data Acquisition
STIX	Structured Threat Information Expression
VoIP	Voice over IP

In the context of the game model, sorted by type and occurrence:

SO	Sophistication (Actor skill level attribute)
DE	Determination (Actor motivation attribute)
WE	Wealth (Actor resources attribute)
INI	Initiative (Actor time efficiency derived attribute)
INS	Insight (Opponent knowledge derived attribute)
TH	Thief (STIX Cyber Espionage Operations actor)
EX	Explorer (STIX Hacker, White Hat actor)
RO	Rogue (STIX Hacker, Grey Hat actor)
RA	Raider (STIX Hacker, Black Hat actor)
CR	Crusader (STIX Hacktivist actor)

OP	Operative (STIX State Actor/Agency actor)
IN	Infiltrator (STIX Insider Threat actor)
PR	Protester (STIX Disgruntled Customer actor)
CP	Production Company (Primary sector actor)
CM	Manufacturing Company (Secondary sector actor)
CS	Services Company (Tertiary sector actor)
IF	Infrastructure provider (Actor)
MI	Military (Actor)
SA	State Actor/Agency (Actor)
ED	Education sector (Actor)
PI	Private Individual (Actor)
ATT	Attacker equipment (Enabler)
MPT	Multi-Purpose Tool (ATT)
Sca	System scanner (ATT)
VSc	Vulnerability Scanner (ATT)
NSc	Network Scanner (ATT)
Wir	Wireless tool (ATT)
Pwd	Password cracker (ATT)
Mal	Malware (ATT)
VUL	Vulnerability (Enabler)
PoC	Proof of Concept (Enabler Maturity)
AST	Asset (Disabler)
SEC	Security solution (Disabler)
Pre	Prevention solution (SEC)
Det	Detection solution (SEC)
Del	Delay solution (SEC)
Rec	Recovery solution (SEC)
Cnt	Countermeasures (SEC)
POL	Policy (Disabler)
FIX	Fix for VUL (Disabler)
***DC, D**	Detection Chance (Effect)
***SC, S**	Success Chance (Effect)
***CR**	Credits (Effect)
***STA**	Status (Effect)
H	Host (Effect target)
M	Mobile system (Effect target)
I	Industrial system (Effect target)
N	Network asset (Effect target)
T	Third party service (Effect target)
R.*	Reconnaissance (APT kill chain phase)
W.*	Weaponization (APT kill chain phase)
D.*	Delivery (APT kill chain phase)
E.*	Exploitation (APT kill chain phase)
I.*	Installation (APT kill chain phase)
C.*	Command and Control (APT kill chain phase)
A.*	Action on Objective (APT kill chain phase)
IG	Information Gathering (CAPEC primary attack class)
IN	Injection (CAPEC primary attack class)
SE	Social Engineering (CAPEC primary attack class)
SA	State Attack (CAPEC primary attack class)
FA	Function Abuse (CAPEC primary attack class)
BF	Brute Force (CAPEC primary attack class)
IA	Illegal Access (CAPEC primary attack class)
DM	Data Manipulation (CAPEC primary attack class)
PR	Preparation (Non-CAPEC primary attack class)
CO	Communication (Non-CAPEC primary attack class)
IL	Information Leakage protection (Primary control class)
CP	Context Protection (Primary control class)
AW	Awareness (Primary control class)
SP	State Protection (Primary control class)
FI	Function Integrity (Primary control class)
AP	Authentication Protection (Primary control class)
AC	Access Control (Primary control class)
DI	Data Integrity (Primary control class)
SI	Security Intelligence (Primary control class)
CS	Communications Security (Primary control class)
ACM	Account Management (NIST-based defense action)
ACE	Access Enforcement (NIST-based defense action)
IFE	Information Flow Enforcement (NIST-based defense action)
LEP	Least Privilege (NIST-based defense action)
REA	Remote Access (NIST-based defense action)
RST	Role-based Security Training (NIST-based defense action)
COM	Continuous Monitoring (NIST-based defense action)
CCC	Configuration Change Control (NIST-based defense action)
COS	Configuration Settings (NIST-based defense action)
COP	Contingency Plan (NIST-based defense action)
AUM	Authenticator Management (NIST-based defense action)
INH	Incident Handling (NIST-based defense action)
NOM	Nonlocal Maintenance (NIST-based defense action)
MES	Media Storage (NIST-based defense action)
SEP	Security Engineering Principles (NIST-based defense action)
BOP	Boundary Protection (NIST-based defense action)
CRP	Cryptographic Protection (NIST-based defense action)
FLR	Flaw Remediation (NIST-based defense action)
MCP	Malicious Code Protection (NIST-based defense action)
ISM	Information System Monitoring (NIST-based defense action)

Appendix B: Action sub-classes

Actions

The definition of action X has been introduced in Sect. 3.3. In this appendix, we specify the sub-classes and provide a detailed view at their data representation.

Actors

The $\langle AttackActor\rangle$ classes currently modeled in PenQuest include: Cyber Espionage Operations (TH), Hacker (white (EX), gray (RO), black hat (RA)), Hacktivist (CR), State Actor/Agency (OP), Insider (IN), and Disgruntled Customer (PR). See Sect. 4.1 for more detailed information. Attacker motivation includes ideological goals ($id.*$), as well as ego-centered (eg), financial (fi), military (mi), opportunistic (op), and political (po) motivations.

$AttackActor = \langle$
$\langle Class\{TH, EX, RO, RA, CR, OP, IN, PR\}\rangle,$
$\langle Motivation\{id.*, eg, fi, mi, op, po\}\rangle,$
$\langle Attributes\langle SO\{1..n\}, DE\{1..n\}, WE\{1..n\}\rangle\rangle,$
$\langle Resources\langle INI\{1..n\}, INS\{1..n\}, \langle Enabler\rangle\rangle\rangle\rangle$

Following the same formula, $\langle DefenseActor\rangle$ classes include the three primary sectors (CP, CM, CS), infrastructure (IF), military (MI), state actors/agencies (SA), the education sector (ED), and private individuals (PI).

$DefenseActor = \langle$
$\langle Class\{CP, CM, CS, IF, MI, SA, ED, PI\}\rangle,$
$\langle Attributes\langle SO\{1..n\}, DE\{1..n\}, WE\{1..n\}\rangle,$
$\langle Resources\langle INI\{1..n\}, INS\{1..n\}, \langle Disabler\rangle\rangle\rangle\rangle$

Equipment

Attacker equipment ($ATT.*$) subsumes mostly attack tools (MPT, Pwd), OS, application, and (wireless) network scanners (Sca), various types of malware (Mal), as well as vulnerabilities (VUL) that modify the chance of success against specific assets.

$Enabler = \langle,$
$\langle Type\{ATT.*, VUL.*\}\rangle,$
$\langle Effect\langle Type\{incSC, decDC\}, EffectValue\{1..n\},$
$EffectTarget\{H, N, I, M, T\}\rangle\rangle,$
$\langle Attributes\langle Sophistication\{1..n\}, Level\{1, 2\}\rangle,$
$\langle Properties\langle Privileges\{high, low\},$
$UserInteraction\{none, required\},$
$\langle Impact\langle C\{low, med, high\},$
$I\{low, med, high\}, A\{low, med, high\}\rangle,$
$Maturity\{unproven, PoC, functional, high\}\rangle\rangle\rangle,$
$\langle Name\{\_ToolName\_\}\rangle$
$\rangle$

Defender equipment encompasses assets to be protected (AST), security policies (POL), fixes (FIX) that counter vulnerabilities, as well as tools that increase security ($SEC.*$), e.g. through prevention (Pre), detection (Det), delay (Del), or by generally hindering the attacker (Cnt).

$Disabler = \langle,$
$\langle Type\{AST.*, SEC.*, POL.*, FIX.*\}\rangle,$
$\langle Effect\langle Type\{decSC, decSC\}, EffectValue\{1..n\},$
$EffectTarget\{H, N, I, M, T\}\rangle\rangle,$
$\langle Attributes\langle Sophistication\{1..n\}, Level\{1, 2\}\rangle\rangle,$
$\langle Properties\langle Maturity\{official, temporary,$
$workaround\}\rangle\rangle,$
$\langle Name\{\_SystemName\_\}\rangle\rangle$

Some enablers and disablers only target or apply to specific equipment categories. The general $\langle Effect\rangle$ of using a piece of equipment depends on its type and is exemplified on a per-item basis in the game rules. Typically, various resources, attributes, and detection/success ($*DC/*SC$) probabilities are either increased ($inc*$) or decreased ($dec*$) upon use, which further impacts future events and the overall course of the game. For the $\langle EffectTarget\rangle$, we generally differentiate host-based (H), network-based (N), industrial (I), mobile (M), and third-party (T) equipment. Enablers generally have an adverse $\langle Impact\rangle$ on the CIA status of the victim (more below).

In each scenario, the attacker attempts to achieve his or her goal by compromising a *Victim* (target) in a specific fashion. This translates to the attempted compromise of one the defender's assets by changing its $\langle Victim\langle Integrity\rangle\rangle$ to 'compromised' or its $\langle Victim\langle Status\rangle\rangle$ to 'stopped'.

$Victim = \langle$
$\langle Type\{Asset, SecuritySolution\}\rangle,$
$\langle Name\{\_SystemName\_\}\rangle,$
$\langle Exposure\{internal, exposed\}\rangle,$
$\langle Parent\{\langle Victim\rangle, \langle Disabler\rangle\}\rangle,$
$\langle VectorParent\{\langle Victim\rangle\}\rangle,$
$\langle Configuration\{tech, org\}\rangle,$
$\langle Knowledge\{Information, \langle Configuration\rangle\}\rangle,$
$\langle Status\langle OperationStatus\{running, affected, stopped\}\rangle,$
$\langle Integrity\{nominal, affected, compromised\}\rangle\rangle\rangle$

Meta Information

Part of the *AttackClass*, $\langle Mode \rangle$ identifies the optional CIA goal [52] and impact rating of the action, which, if successful, is reflected in the victim service's $\langle Status \rangle$ information. *AttackClass* information helps to abstract specific attack actions and provides a means to link adversary behavior to possible defense measures.

$AttackClass = \langle$
$\langle Stage\{R.*, W.*, D.*, E.*, I.*, C.*, A.*\}\rangle,$
$\langle PatternClass\{IG, IN, SE, SA, FA, BF, IA, DM, PR, CO\}\rangle,$
$\langle Mode\langle Goal\{C, I, A\}, Rating\{low, med, high\}\rangle\rangle\rangle$

$DefenseClass = \langle$
$\langle Category\{organization, information\_system\}\rangle$
$\langle ControlClass\{IL, CP, AW, SP, FI, AP, AC, DI, SI, CS\}\rangle$
$\langle ActionClass\{ACM, ACE, IFE, LEP, REA, AMO, RST,$
$COM, CCC, COS, COP, COP, AUM, INH, NOM, MES,$
$PAC, SEP, SCP, BOP, CRP, FLR, MCP, ISM\}\rangle\rangle$

Requirements are optional and depend on the complexity of the respective attack, which is defined through its *Properties*, namely minimum required actor skill (Sophistication SO) and the chance of success (SC) as well as detection (DC).

$Requirements = \langle$
$\langle *Actor\langle Attr.\langle SO\{1..n\}, DE\{0..n\}, WE\{0..n\}\rangle\rangle\rangle,$
$\langle *Actor\langle Resources\langle INI\{1..n\}, INS\{1..n\}\rangle\rangle\rangle,$
$\langle Victim\langle Exposure\{internal, exposed\}\rangle\rangle,$
$\langle Victim\langle Integrity\{nominal, affected, compromised\}\rangle\rangle$
$\rangle$
$Properties = \langle$
$\langle Sophistication\{1..n\}\rangle, \langle SC\{0..100\%\}\rangle, \langle DC\{0..100\%\}\rangle\rangle$

Each action corresponds to an observable *AttackPattern*, which is identified by its ID and its impact on the CIA triad. Attack patterns link the modeled action to specific hostile activity as described in the CAPEC schema.

$AttackPattern = \langle$
$\langle Purpose\langle Exploitation\{T, F\}, Obfuscation\{T, F\},$
$Penetration\{T, F\}, Recon\{T, F\}\rangle\rangle,$
$\langle Impact\langle C\{low, medium, high\}, I\{low, medium, high\},$
$A\{low, medium, high\}\rangle\rangle,$
$\langle ID\{n\}\rangle\rangle$

Events contain information about basic (on-system) operations and arguments, triggers (parents), timestamps, anomaly scores, and sequence numbers. The 'pattern' type describes event sequences that directly represent the action, while an 'anomaly' describe the behavioral deviation from a baseline.

$Event = \langle$
$\langle Type\{Pattern, Anomaly\}\rangle$
$\langle Time\langle Start\{\_timestamp\_\}, End\{\_timestamp\_\}\rangle\rangle,$
$\langle Score\langle Deviation\{0..n\}, Threshold\{0..n\}\rangle,$
$\langle Sequence\{n\}\rangle,$
$\langle Parent\{\_ParentName\_\}\rangle,$
$\langle Operation\{\_OperationName\_\}\rangle,$
$\langle Argument\{\_OperationArgument\_\}\rangle\rangle$

Appendix C: Defense action mappings

Find below the mapping of primary controls to defense actions, as discussed in Sect. 4.5.3. The first defense action exemplarily includes related controls as well as control enhancements. A full list of control enhancements can be retrieved from NIST SP 800-53, Appendix F.

- **Account Management *ACM*** (AC-2 → AC): Establishes conditions for group and role membership, specifies authorized users and access authorizations (i.e., privileges), creates, enables, modifies, disables, and removes information system accounts, and monitors their use.
  - Related (non-primary) information system controls: Concurrent session control (AC-10), Protection of audit information (AU-9), Identification and authentication for organizational (IA-2) and non-organizational users (IA-8).
  - Control enhancements: Automated system account management, Removal of temporary/emergency accounts, Disable inactive accounts, Automated audit actions, Inactivity logout, Dynamic privilege management, Role-based schemes, Dynamic account creation, Restrictions on use of shared/group accounts, Shared/group account credential termination, Usage conditions, Account monitoring/atypical use, Disable accounts for high-risk individuals.
- **Access Enforcement *ACE*** (AC-3* → AC): Enforces approved authorizations for logical access to information and system resources.
- **Information Flow Enforcement *IFE*** (AC-4* → DI, CS): Enforces approved authorizations for controlling the flow of information within and between interconnected systems.
- **Least Privilege *LEP*** (AC-6 → AC): Employs the principle of least privilege, permitting only authorized accesses for users which are vital to accomplishing assigned tasks.

- **Remote Access *REA*** (AC-17 → AC, AP): Establishes, authorizes, and documents usage restrictions, configuration/connection requirements, as well as implementation guidance for each type of remote access allowed.
- **Wireless access *WIA*** (AC-18): Establishes usage restrictions, configuration/connection requirements, and implementation guidance for wireless access and authorizes wireless access to the information system. This action category is not part of the initial version of PenQuest.
- **Access control for mobile devices *AMO*** (AC-19 → AC): Establishes usage restrictions, configuration requirements, connection requirements, and implementation guidance for organization-controlled mobile devices and authorizes the connection of mobile devices to organizational information systems.
- **Role-based Security Training *RST*** (AT-3 → AW): Provides security training to personnel with specific security roles and responsibilities.
- **Continuous Monitoring *COM*** (CA-7 → AC, CS): Establishes and monitors metrics and frequency related to ongoing status monitoring of controls and information systems.
- **Configuration Change Control *CCC*** (CM-3 → SP, FI): Reviews proposed configuration-controlled changes to the information system and approves or disapproves such changes while also documenting and implementing the change itself.
- **Configuration Settings *COS*** (CM-6 → IL, SP): Establishes and documents configuration settings for information technology products employed. Implements, checks, and monitors said settings.
- **Contingency Plan *COP*** (CP-2 → SP, DI): Develops a contingency plan that provides recovery objectives, restoration priorities, and metrics for maintaining essential missions and business functions despite information system disruption, compromise, or failure.
- **Authenticator Management *AUM*** (IA-5 → AC, AP): Manages information system authenticators (tokens, passwords, etc.) by verifying that authenticators have sufficient strength of mechanism for their intended use. Manages the changing of default content and establishes minimum and maximum authenticator lifetime restrictions.
- **Incident Handling *INH*** (IR-4 → SP, DI): Implements security incident handling procedures that include preparation, detection, analysis, containment, eradication, and recovery. Also coordinates incident handling activities with contingency planning, and incorporates lessons learned from ongoing incident handling activities into incident response procedures, training, and testing.
- **Nonlocal Maintenance *NOM*** (MA-4 → SP): Approves, performs, and monitors nonlocal (remote) maintenance and diagnostic activities with permitted tools.
- **Media Storage *MES*** (MP-4 → DI): Physically controls and securely stores types of digital and/or non-digital media within controlled areas and protects information system media until it is destroyed or sanitized.
- **Physical access control *PAC*** (PE-3 → IL, AC): Enforces physical access authorizations at entry/exit points to the facility where the information system resides by verifying individual access authorizations, maintaining access audit logs for entry/exit points and by providing security safeguards to control access to facility. Not currently modeled.
- **Security Engineering Principles *SEP*** (SA-8 → CP, FI): Applies information system security engineering principles in the specification, design, development, implementation, and modification of an information system.
- **Supply chain protection *SCP*** (SA-12 → AP): Protects against supply chain threats to the information system, system component, or information system service by employing security safeguards as part of a comprehensive, defense-in-breadth information security strategy. Not currently part of PenQuest.
- **Boundary Protection *BOP*** (SC-7* → IL, CS): Monitors and controls communications at the external boundary of the system as well as at key internal boundaries through e.g. network segmentation.
- **Cryptographic Protection *CRP*** (SC-13* → IL, DI, CS): Implements cryptographic protection in various contexts.
- **Flaw Remediation *FLR*** (SI-2 → FI): Identifies, reports, and corrects information system flaws. Installs security-relevant software and firmware updates.
- **Malicious Code Protection *MCP*** (SI-3 → CP, FI, DI): Employs malicious code protection mechanisms at information system entry and exit points to detect and eradicate malware.
- **Information System Monitoring *ISM*** (SI-4 → CP, SP, FI, DI, CS): Monitors information systems to detect attacks and indicators of potential attacks in accordance with monitoring objectives such as unauthorized local, network, and remote connections.

References

1. Barnum, S.: Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX^{TM}). MITRE Corp. **11**, 1–22 (2012)
2. Beckers, K., Pape, S.: A serious game for eliciting social engineering security requirements. In: 2016 IEEE 24th International Requirements Engineering Conference (RE), pp. 16–25. IEEE (2016)
3. Benoit, K., Watanabe, K., Nutly, P., Obeng, A., Wang, H., Lauderdale, B., Lowe, W.: Quanteda: Quantitative Analysis of Textual Data. http://quanteda.io. R package version 0.99 (2017)
4. Blakley, B., McDermott, E., Geer, D.: Information security is information risk management. In: Proceedings of the 2001 Workshop on New Security Paradigms, pp. 97–104. ACM (2001)

5. BSI: Durchführung von Planspielen zur Informationssicherheit. Technical report (2014)
6. Caltagirone, S., Pendergast, A., Betz, C.: The Diamond Model of Intrusion Analysis. Technical report, Center for Cyber Intelligence Analysis and Threat Research, Hanover (2013)
7. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
8. Cook, A., Smith, R., Maglaras, L., Janicke, H.: Measuring the risk of cyber attack in industrial control systems. In: Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016, ICS-CSR'16, pp. 1–11. BCS Learning & Development Ltd., Belfast, United Kingdom (2016). https://doi.org/10.14236/ewic/ICS2016.12
9. Dicheva, D., Dichev, C., Agre, G., Angelova, G.: Gamification in education: a systematic mapping study. J. Educ. Technol. Soc. **18**(3), 75 (2015)
10. Engebretson, P.H., Pauli, J.J., Streff, K.: Abstracting parent mitigations from the CAPEC attack pattern dictionary. In: Security and Management, pp. 245–250 (2008)
11. Esparza, J., Leucker, M., Schlund, M.: Learning workflow petri nets. In: International Conference on Applications and Theory of Petri Nets, pp. 206–225. Springer, New York (2010)
12. Fenz, S., Ekelhart, A.: Formalizing information security knowledge. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, pp. 183–194. ACM (2009)
13. Freytag, T.: Woped–workflow petri net designer. In: University of Cooperative Education, pp. 279–282 (2005)
14. Hanus, M.D., Fox, J.: Assessing the effects of gamification in the classroom: a longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. Comput. Educ. **80**, 152–161 (2015)
15. Holmgren, A.J., Jenelius, E., Westin, J.: Evaluating strategies for defending electric power networks against antagonistic attacks. IEEE Trans. Power Syst. **22**(1), 76–84 (2007)
16. Hutchins, E.M., Cloppert, M.J., Amin, R.M.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Lead. Issues Inf. Warf. Secur. Res. **1**, 80 (2011)
17. Joint Task Force Transformation Initiative: SP 800-53 rev. 4. Recommended Security Controls for Federal Information Systems and Organizations. Technical report, Gaithersburg, MD (2015)
18. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
19. Kissel, R.: Glossary of key information security terms. NIST Interagency Reports NIST IR 7298(3) (2013)
20. Kohnfelder, L., Garg, P.: The threats to our products. Microsoft Corporation (1999). https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx
21. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Foundations of attack–defense trees. In: International Workshop on Formal Aspects in Security and Trust, pp. 80–95. Springer, Berlin (2010)
22. Kuhn, H.W.: Lectures on the Theory of Games. Princeton University Press, Princeton (2009)
23. Landoll, D.J., Landoll, D.: The Security Risk Assessment Handbook: A Complete Guide for Performing Security Risk Assessments. CRC Press, Boca Raton (2005)
24. Lewis, T.G.: Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation. Wiley, Berlin (2014)
25. Locke, J.: Some Thoughts Concerning Education. Cambridge University Press, Cambridge (1895)
26. Luh, R., Marschalek, S., Kaiser, M., Janicke, H., Schrittwieser, S.: Semantics-aware detection of targeted attacks: a survey. J. Comput. Virol. Hacking Tech. **13**, 47–85 (2016)
27. Luh, R., Schrittwieser, S., Marschalek, S.: TAON: an ontology-based approach to mitigating targeted attacks. In: Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services. ACM (2016)
28. Luh, R., Janicke, H., Schrittwieser, S.: AIDIS: detecting and classifying anomalous behavior in ubiquitous kernel processes. Comput. Secur. **84**, 120–147 (2019)
29. Marczewski, A.: Even Ninja Monkeys Like to Play: Gamification. CreateSpace Independent Publishing Platform, Game Thinking and Motivational Design (2015). ISBN 9781514745663
30. Maslow, A.H.: A theory of human motivation. Psychol. Rev. **50**(4), 370 (1943)
31. Mavroeidis, V., Bromander, S.: Cyber threat intelligence model: an evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In: 2017 European Intelligence and Security Informatics Conference (EISIC), pp. 91–98. IEEE (2017)
32. MITRE Corporation: CAPEC—Common Attack Pattern Enumeration and Classification (CAPEC). https://capec.mitre.org/. Accessed 22 Sept. 2015
33. MITRE Corporation: CVE—Common Vulnerabilities and Exposures (CVE). https://cve.mitre.org/. Accessed 22 Sept. 2015
34. MITRE Corporation: CWE—Common Weakness Enumeration. https://cwe.mitre.org/. Accessed 22 Sept. 2015
35. MITRE Corporation: STIX—Structured Threat Information Expression | STIX Project Documentation. https://stixproject.github.io/. Accessed 22 Sept. 2015
36. Miura-Ko, R.A., Yolken, B., Bambos, N., Mitchell, J.: Security investment games of interdependent organizations. In: 2008 46th Annual Allerton Conference on Communication, Control, and Computing, pp. 252–260. IEEE (2008)
37. Mobasher, B., Burke, R., Sandvig, J.J.: Model-based collaborative filtering as a defense against profile injection attacks. In: AAAI, volume 6, p. 1388 (2006)
38. Morgan, S.: 2017 Cybercrime Report. Technical report, Cybersecurity Ventures (2017)
39. Munsey, C.: Economic Espionage: Competing for Trade by Stealing Industrial Secrets. https://leb.fbi.gov/2013/october-november/economic-espionage-competing-for-trade-by-stealing-industrial-secrets. Accessed 15 Sept. 2015 (2013)
40. Myerson, R.B.: Game theory: analysis of conflict. Harvard University Press (1991). ISBN 9780674341159. https://books.google.at/books?id=1w5PAAAAMAAJ
41. Nguyen, K.C., Alpcan, T., Basar, T.: Security games with incomplete information. In: International Conference on Communications, 2009. IEEE ICC'09. pp. 1–6. IEEE (2009)
42. Ponemon Institute: Cost of cyber crime study: Insights on the security investments that make a difference, Accenture (2017)
43. Rieb, A., Lechner, U.: Operation digital chameleon: towards an open cybersecurity method. In: Proceedings of the 12th International Symposium on Open Collaboration, p. 7. ACM (2016)
44. Rieb, A.J., Hofmann, M., Laux, A., Rudel, S., Lechner, U.: Wie IT-Security Matchplays als Awarenessmaßnahme die IT-Sicherheit verbessern können. In: 13. Internationale Tagung Wirtschaftsinformatik, pp. 867–881 (2017)
45. Roy, A., Kim, D.S., Trivedi, K.S.: Attack countermeasure trees (act): towards unifying the constructs of attack and defense trees. Secur. Commun. Netw. **5**(8), 929–943 (2012)
46. Roy, S., Ellis, C., Shiva, S., Dasgupta, D., Shandilya, V., Wu, Q.: A survey of game theory as applied to network security. In: 2010 43rd Hawaii International Conference on System Sciences (HICSS) pp. 1–10. IEEE (2010)
47. Sauerwein, C., Sillaber, C., Mussmann, A., Breu, R.: Threat intelligence sharing platforms: an exploratory study of software vendors and research perspectives. In: Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik (2017)
48. Shang, Y.: Optimal attack strategies in a dynamic botnet defense model. Appl. Math. Inf. Sci **6**, 29–33 (2012)

49. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., Ideker, T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. **13**(11), 2498–2504 (2003)
50. Shaw, R.S., Chen, C.C., Harris, A.L., Huang, H.-J.: The impact of information richness on information security awareness training effectiveness. Comput. Educ. **52**(1), 92–100 (2009)
51. Shostack, A.: Elevation of privilege: drawing developers into threat modeling. In: 3GSE (2014)
52. Stoneburner, G., Goguen, A.Y., Feringa, A.: SP 800–30. Risk Management Guide for Information Technology Systems, Technical report (2002)
53. Syed, Z., Padia, A., Finin, T., Mathews, M.L., Joshi, A.: UCO: a unified cybersecurity ontology. In: Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security. AAAI Press (2016)
54. Symantec: Internet Security Threat Report Volume 20. Symantec (2015)
55. Symantec: Internet Security Threat Report Volume 23. Symantec (2018)
56. Thomson, M.E., von Solms, R.: Information security awareness: educating your users effectively. Inf. Manag. Comput. Secur. **6**(4), 167–173 (1998)
57. Undercoffer, J., Pinkston, J., Joshi, A., Finin, T.: A target-centric ontology for intrusion detection. In: 18th International Joint Conference on Artificial Intelligence, pp. 9–15 (2004)
58. Wang, J.A., Guo, M.: OVM: an ontology for vulnerability management. In: Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, p. 34. ACM (2009)
59. Wen, Z.A., Li, Y., Wade, R., Huang, J., Wang, A.: What.hack: Learn phishing email defence the fun way. In: Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA'17, pp. 234–237, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4656-6. https://doi.org/10.1145/3027063.3048412
60. You, X.Z., Shiyong, Z.: A kind of network security behavior model based on game theory. In: Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. pp. 950–954. IEEE (2003)