Creating Attack Graphs for Adversary Emulation, Simulation and Purple Teaming in Industrial Control System (ICS) Environments

Master Thesis
zur Erlangung des akademischen Grades
M.Sc. Praktische Informatik

der Fakultät
Mathematik und Informatik
der FernUniversität
in Hagen von

Jan Hoff

Hagen 2021

Vorsitzender des Prüfungsausschusses:	Prof. Dr.-Ing. Jörg M. Haake
1. Gutachter:	PD Dr.-Ing. habil. Mario Kubek
2. Gutachter:	Prof. Dr. Lars Fischer
Eingereicht am:	12.01.2021

Acknowledgements

I want to thank my supervisors, PD Dr.-Ing. habil. Mario Kubek and Prof. Dr. Lars Fischer, for supporting me during the creation of this thesis. Their feedback has always been valuable and motivated me constantly.
This thesis would also not have been possible without researchers, practitioners and great minds who provided a foundation and know-how required to write this thesis.
Additionally, thanks go to friends and colleagues providing feedback, additional ideas and proofreading of this thesis.
I also want to thank all my interview partners, from academia and corporate organizations, for providing valuable input and constructive feedback on the implemented prototype, graph design and use cases. Despite the pandemic, everyone found time and proved that information sharing is key to tackle the challenges we face in the IT and OT security community.

Interview partners during the evaluation were (in alphabetic order):

- Christoph Alms
- Manuel Atug
- Gregor Bonney
- Christian Book
- Christian Bruns
- Prof. Dr. Christian Dietrich
- Sascha Eilers
- Sarah Fluchs
- Marco Murch
- Michael Pietsch
- Björn Siemers
- Joe Slowik
- Daniel Szameitat

Lastly, I thank everyone working relentlessly to protect critical infrastructures, sharing information on threat actors and reminding the information security industry that physical processes and safety have to be protected first and foremost.

Abstract

This work shows the viability of automatically generated attack graphs that are used for adversary behavior execution in industrial control system environments. This viability is evaluated and confirmed with expert interviews about results of this thesis, generated attack graphs and an implemented prototype.

Since existing graph-based approaches do not fulfil requirements that are set by adversary emulation or simulation activities like Purple Teaming, a new concept for attack graphs and a corresponding prototype had to be developed. For this development theoretical foundations were used to identify requirements and use cases.

The theoretical background contains information on cyberattacks, adversary emulation and simulation (e.g. Purple Teaming) as well as attack graphs. Another background section covers standard architectures and information security aspects of industrial control system environments.

Additionally to research done in this work, experts interviews allowed to derive additional room for research and additional use cases for the realized prototype. Attack graphs have shown to be a versatile means to structure risk analysis and support adversary behavior execution activities.

Zusammenfassung

Diese Masterarbeit zeigt die Umsetzbarkeit von automatisch generierten Angriffsgraphen zur Reproduktion von Angreiferverhalten in Umgebungen mit industriellen Steuerungssystemen. Diese Umsetzbarkeit wurde im Rahmen von Experteninterviews hinsichtlich der Forschungsergebnisse, der generierten Angriffsgraphen sowie dem implementierten Prototyp überprüft und validiert.

Nachdem bestehende graph-basierte Ansätze nicht die Anforderungen an Reproduktion von Angreiferverhalten (z.B. Purple Teaming) erfüllen können, wurde ein neues Konzept für Angriffsgraphen und einen dazugehörigen Prototyp zur Generierung entwickelt. Für diese Entwicklung wurden von den theoretischen Grundlagen entsprechende Anwendungsfälle abgeleitet.

Die theoretische Grundlagen beinhalten Informationen über Cyber-Angriffe, Angriffssimulation und -emulation (z.B. Purple Teaming) sowie Angriffsgraphen. Ein weiterer Hintergrundabschnitt behandelt die Architektur und Besonderheiten von Informationssicherheit in industriellen Steuerungssystemen.

Zusätzlich zu den Forschungsergebnissen dieser Arbeit konnten aus den Experteninterviews weiterer Forschungsbedarf sowie weitere Anwendungsfälle für den implementierten Prototypen abgeleitet werden. Angriffsgraphen haben sich als eine vielfältige Möglichkeit dargestellt, Risikoanalysen zu strukturieren und Angreiferverhalten im Rahmen von Übungen zu reproduzieren.

Contents

Chapter 1

Introduction and Motivation

The defense of networks and interconnected infrastructures is becoming increasingly difficult. Combined with growing capability of attackers, organizations are required and mandated to increase resilience. This does not only apply to the cyber domain, but to the whole process chain as shown e.g. in [96] and [24].

Today's attacks on control systems, especially critical infrastructures, receive more attention than they did a few years ago. Since detection of Stuxnet, theoretical and practical attacks against control infrastructure and associated networks have been observed worldwide [96]. One of the key events in the recent past was an attack on Ukrainian power infrastructures. Threat actors were able to compromise multiple Ukrainian utilities companies in 2015 and 2016. This resulted in power outages in December of each year. Those outages were traced to compromised IT assets and lateral movement towards control system infrastructure in the affected companies by an advanced adversary, classified as "Advanced Persistent Threat" [54][56][34]. Aside from technical shortcomings, Ukrainian personnel was not prepared or trained to handle such an incident.

Taking into account an increasing digitalization, defense of modern control environment is key for an effective resilience against adversaries. One defensive measure consists of properly trained personnel and associated validated technical controls. Operators of industrial control systems (ICS) have limited or no capability to train the defenders against an advanced attacker by conventional trainings or exercises. In most organizations this results in exercises repeating previous, well-known scenarios or fragments from the past and do not consider a novel end-to-end intrusion. This limitation reduces training effects and efficiency. Effective defenders must be trained with previously unknown realistic adversary profiles, otherwise tools and processes might be overfitted to known scenarios.

Many publications in academia focus on a defender's perspective. Regardless of perspective, increasing resilience of processes and technology should always be a goal. This thesis promotes an attacker's view to improve security posture of people, processes and technology of an organization. The ability to assess and mitigate risk correlates with the understanding of attacker behavior. Known adversarial behavior can be defended against, but unknown behavior is likely to bypass defenses.

Simply reproducing known attacks from the past will create scenarios that are either already known or far-fetched. The challenge of unrealistic and repetitive training scenarios can be countered creating novel, individual adversary profiles and executing their behavior in exercises and assessments. This allows to cover end-to-end intrusions even in environments that rarely encounter attacks. Using these kind of adversaries profiles in training, organizations can gain necessary capabilities to thwart advanced attackers in the future and validate protective efforts.

Today, organizations and especially operators of ICS cannot simulate or emulate aforementioned adversaries sufficiently. This is especially the case when trying to deliver proper trainings and exercises that increase resilience. Organizations usually lack knowledge and scenarios (playbooks), which are required for adversary behavior execution. Additionally, since only a few public incidents with ICS technology were documented, the number of available attack graphs is limited and those graphs are mostly static and a result of significant manual effort to create.

A major success factor in effective detection of and reaction to an attack is a defender's ability to have proven methods for incident response and analysis before the attack. It has to be expected that attacks on critical infrastructures will continue to involve a digital dimension. This means defenders also have to move into the digital domain as well. A usually complex chain of effects that is part or leads to attacks on industrial infrastructures is a challenge for defenders. In contrast to conventional IT intrusions, attacks against Operational Technology (OT) environments/ICS are not encountered or trained often.

Using attack graphs has been common in the defensive information security domain, but has not seen many references for attack emulation or simulation (see Kaynar in [44] for a taxonomy of attack graphs). Usually graphs depicting an incident are often created in hindsight after attacks come to light or incidents happen. Additionally those graphs are hardly modified, once they are created [60]. This property makes them less usable for aforementioned training scenarios.

Techniques and methods of real attackers constantly change and expand. They do not only combine existing methods and techniques, but adversaries also add new techniques to their arsenal. The combination of attacker behavior and attributes are subsumed in the NIST Standard "800-150" as Tactics, Techniques and Procedures (TTP) [42]. To emulate these TTP in trainings and simulations is task of a Red or Purple Team. Those teams require playbooks (step-by-step attack sequences) for actions they have to perform. Playbooks can be based on those aforementioned attack graphs as shown in this thesis.

For the purposes of adversary behavior execution (simulation and emulation), individual and customized attack graphs are required to represent previously unknown attacker groups. This means combining existing TTP into artificial attacker profiles. As stated in the following research question, a generation of these types of individual attack graphs is subject of this thesis. A program or tool set for generating new attack paths and attacker behavior for training purposes currently does not exist for the OT domain, but is required to improve defender's capabilities to detect and react to security anomalies.

Existing attack graph approaches, including their tools, lack a possibility to be individualized for organizations using them and cannot combine TTP from multiple threat actors. For successful use in trainings and simulations, existing attack graphs are not sufficient from a use case perspective, lack flexibility or have an inappropriate level of complexity.

1.1 Research Question

From aforementioned challenges, statements and identified shortcomings the following research question has been developed. At the end of this thesis, an answer to this question is provided and results critically reflected.

The research question of this thesis is as follows:

> Is it possible – and if yes to what extent – to algorithmically generate attack graphs that can be used for practical adversary behavior execution in ICS environments and can the process be supported by a corresponding application?

1.2 Thesis Structure and Approach

An engineering approach is taken to design and implement a prototype to answer the aforementioned research question. This thesis will start with that problem statement, cover requirements for attack graphs and a prototype, system design and implementation phases and close with a solution and evaluation of results. Following is a brief description of each chapter and their content. A high level overview of the approach is also depicted in Figure 1.1. The Background chapter as well as the chapter visualizing results has been removed from the overview to maintain readability. They are only indirectly involved in answering the research question.

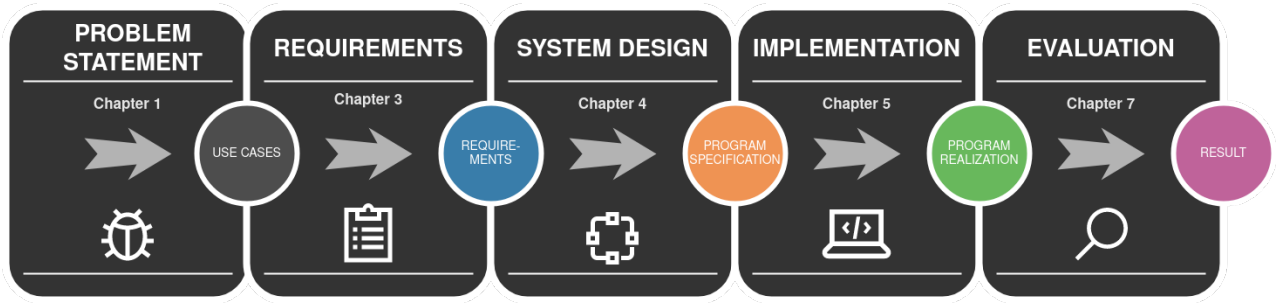


Figure 1.1: Approach and mapping to thesis' chapters

In Chapter 1 an introduction given; stating motivation and problems the IT security community faces when trying to emulate or simulate adversary behavior. It also provides an overview about this thesis' chapter structure.

Theoretical foundation for this thesis is provided in Chapter 2. Key concepts, required for an understanding of the solution, are introduced and explained. This chapter does not provide a full, in-depth, extensive or exhaustive review of those concepts. A reader is referred to referenced literature for details.

Requirements for attack graphs in emulation and simulation scenarios are analyzed in Chapter 3. Use-cases of the application as well as the functional and non-functional requirements are introduced and put into context of the foundation chapter.

Chapter 4 details the design and concepts for this thesis' solution. Individual chapters are referenced accordingly and put into context of the development steps. The design includes a graph approach as well as a prototype's architecture and database layout. For this purpose, identified requirements and approaches from related work are put into context and a decisions for a prototype implementation explained.

In Chapter 5 implementation aspects of this thesis are covered. This includes approaches for graph generation and rendering as well as a database implementation. Additionally, limitations of realized prototype and taken decisions are identified and put into context.

A visual overview of the prototype's different aspects is given in Chapter 6. User interfaces and generated graphs that are created and selected for evaluation in the subsequent chapter are depicted. For this purpose different graphs are generated with varying constraints.

In Chapter 7 the prototype implementation and solution is evaluated. To support such an evaluation, a set of expert interviews have been performed. In addition to the expert's views, this chapter also provides a reflection of results by the author and subsequently allows to answer the research question stated in Section 1.1.

This work is concluded with Chapter 8. This chapter contains a summary of results, an outlook to development of adversary behavior execution in ICS environments and possibilities to extend the solution in the future. This chapter ends with further possibilities for research.

Chapter 2

Background

Following sections provide an overview about the foundation and context that is used in the course of this thesis and the corresponding implementation of a prototype to aid in generating attack graphs. Before going into design and implementation of such a solution in the following chapters, this chapter will first provide an overview about anatomies and pattern of attacks that are part of an intrusion. Existing frameworks like MITRE ATT&CK and the Cyber Kill Chain are covered accordingly. The following section covers exercises involving an emulation or simulation of adversarial behavior. Additionally, that information serves as a foundation for the use cases discussed in Chapter 3.1. Following that, concepts and existing approaches for attack graphs are covered. Since focus of this thesis is placed on ICS environments, the last section of the Background chapter covers foundations and necessary background information for such industrial architectures.

The following questions are discussed in this Background Chapter and serve as a foundation for providing a solution to the research question stated in Section 1.1:

1. What are cyberattacks and how are they structured?
2. What are tactics, techniques and procedures (TTP) of attackers?
3. What is adversary behavior execution activities and what are their goals?
4. What are attack graphs and how can we use graphs to represent attacks?
5. How are attacks, exercises and attack graphs combined in this thesis?
6. What are ICS environments and what is their architecture and security posture?

2.1 Cyberattacks and Ontologies

As a first background topic, the following section will cover general cybersecurity related aspects relevant to this thesis. This includes a definition of cyberattacks and ontologies to describe different phases and attributes of an attack.

Information technology (IT) and information security has become an important part for many business processes in corporate environments, governments and science. Thus those parts have also gained attention from criminals and other adversaries (see Chapter 1).Recent cyberattacks have resulted in billions of damages to the global economy. Malware NotPetya alone cost logistics company Mærsk $250-$300 million [31].

Aside from financial losses, attacks on critical infrastructures have impact beyond share prices or money. In 2015 and 2016 adversaries were able to compromise the power grid in Ukraine. These incidents have been the first recorded cyberattacks to turn off power for civilians as Greenberg documented in his book Sandworm [32]. Those attacks were the second documented, notable incident on industrial controls systems of that scale after Stuxnet. The associated operations and adversarial actions were investigated publicly by researchers and also shared in the information security community.

In order to emulate an adversary that performs cyberattacks on control systems, and also to generate appropriate attack-graphs, a common understanding of a (cyber)attack has to be established. A commonly accepted definition of the term "cyberattack" does not exist though. Statistics and reporting also do not provide an answer, since presumably many attacks or adversarial actions in IT and OT environments are either not detected, not reported at all or only reported in an incomplete manner. This is especially true for industrial control systems, since IT security and cyberattacks have only recently received attention in that domain. When taking into account Turk et al. in [105], data sets are incomplete in this regard.

One approach for a definition of "attack" is available in the ISO/IEC 27000 standard [39]. This standard defines an attack as an "attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset". This definition is rather broad and leaves room for interpretation, but it also shows how diverse cyberattacks can be.

Knowledge and research about threats does not stand for itself, but is part of a bigger picture. For defenders and decision makers to properly respond to threats and attacks, information needs to continuously gathered, analyzed and shared. That information comes e.g. in the form of evidence-based knowledge, indicators and context. The collection and processing of aforementioned data is commonly referred to as TI or "threat intelligence" [65]. The following sections should all be seen in a context of cyber threat intelligence, since the description and analysis of adversarial behavior and models will aid in detection of and response to an attack as per the ISO definition in the previous paragraph.

When trying to emulate or simulate an adversary, details about a threat that should be simulated need to be available. Those details like adversary groups, behavior, indicators and methods can be derived from threat intelligence resources. Further details on threat intelligence and its role in the information security context can be found in [104],[98] and [64]. An extensive overview on sharing threat intelligence can be found in [42].

To enable threat intelligence sharing and consumption in organizations, common understanding (in form of standards and models) as well as appropriate tools need to be available. Pawlinski et al. give in [79] an overview about standards currently used for information sharing in a threat intelligence context. They show that large ecosystems exist of different standards for different purposes and that sharing information requires proper tooling. This holds true not only for consuming data, but also when producing information about adversarial actions.

Attack graphs covered in this thesis should also be associated with threat intelligence on a broader sense. They do not only consume information provided but also generate potentially new insights that aid defenders. They combine information about attacks into new adversary profiles. The following sections cover the elements that can be used to model an attacker. When describing adversaries, aside from the adversary itself, two domains are relevant. The observables on the one hand and behavior on the other hand. Figure 2.1 provides an overview.

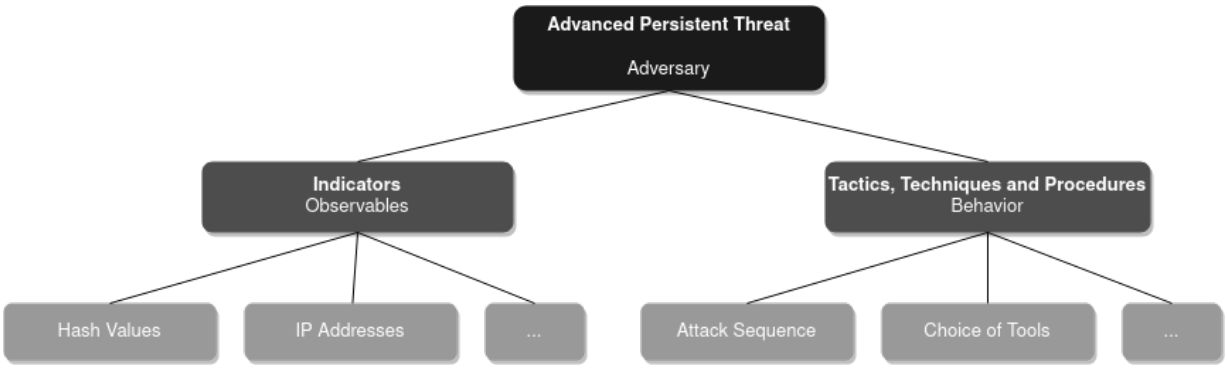


Figure 2.1: Threat intelligence information

2.1.1 Advanced Persistent Threats (APT)

Adversaries successfully breaching critical infrastructures have in common that they possess extensive expertise and resources and may retain long-time access until they achieve their goals. The NIST standard "SP 800-39" [73] defines the term Advanced Persistent Threat (APT) for the classification of these groups of adversaries. Initially it was used in a military context, but the acronym APT has now found its way also into the broader information security terminology. It is not only used to define threat actors themselves, but also actions they take. In the course of the thesis, the term APT is used as "an adversary that possesses sophisticated levels of expertise and significant resources [...]"[73].

Defenders and researchers can establish a common understanding for a certain threat actor by clustering campaigns, behavior and indicators into APT groups. This allows to assess risk associated with a threat actor for an organization and improve detection and protection mechanisms, where required. Additionally, one can try to attribute certain groups to certain interests and thus identify intent and relevance for a certain domain (e.g. critical infrastructures, finance, etc.).

Classifying a threat actor as an APT is not achieved by using simple metrics and is up to the discretion of researchers and intelligence institutions. The transition from presumed, conventional cybercrime to APT is smooth. Often reporting about incidents is delayed, e.g. due to classification of material, and only fragments of attacks are available to researchers and security companies. Thus a threat actor that might not be initially classified as an APT might be "promoted" to APT level. An overview of over 40 known threat actors and corresponding research sources is available in Lemay's paper in [57]. Of those APT groups only a small subset is related to attacks on ICS environments though. Details on ICS are depicted in Section 2.4 of this thesis.

As an example: the APT group "Sandworm" was responsible, among other incidents, for the power outages in Ukraine in 2015 [34] and 2016 [54]. Forensic investigation recovered traces that allowed reconstruction of the steps taken and the vulnerabilities the attacker exploited to execute the breaker operation leading to failures in the distribution grid. They used a known malware to gain credentials to remote access functionality. After switching breakers in multiple substations, they hampered recovery efforts by rendering the control systems useless. The adversary was present in the victim's network for multiple month (persistent), was performing complex, previously unseen actions within the target environment (advanced) and they strategically targeted the power grid (threat). All of this allows to classify Sandworm and its actions as APT.

With the ability to assign certain attributes to adversary groups comes the question of attribution. Attribution of actions, e.g. to a state or a crime organization, in the context of cyberattacks is even harder than it is in the physical domain. With computers being networked worldwide and actions performed location independent, it is hard for researchers and investigators to identify the real origin. Despite its difficulties and potential for false positives and false-flag operations[1], many threat intelligence products, e.g. reports or analysis, try to associate activity to a state or threat group.

Attribution is a highly disputed topic in academia as well as in state and corporate environments. On the one hand it is argued that due to incomplete information, the possibility for false-flags and the anonymity of cyberspace an attribution will always be flawed and potentially misleading. On the other hand attribution is deemed necessary for appropriate response and gives an actor something tangible, especially in a political environment. Rid and Buchanan cover attribution in [83]. In the context of this thesis, attribution is only done as far as reproducing existing attacks and potentially have an association with a certain actor. It does not change the way an attack graph is constructed knowing who is responsible for an attack.

2.1.2 Indicators of Compromise (IOC)

Identifying intrusions and APT in infrastructures is often difficult, since attackers can hide and act with different methods. Adversaries and attacks have specific attributes associated with them. Adversaries leave traces that are results of their actions. They can be purely technical (e.g. hash values) or behavioral (e.g. sequence of attacks). When describing attacks and artefacts commonly the term "Indicators of Compromise" or just "indicators" is used. It is often written in its abbreviated form: IOC.

Steffens defines Indicators of Compromise (IOC) in [98] as "[...] a technical artefact, which existence in a system or network indicates an attack or intrusion". The name IOC implies that they can be used to detect a compromise of an infrastructure or system and less a specific adversary itself, at least usually not by a single indicator. Comprehensive information on IOC and corresponding examples can also be found in [86].

IOC are commonly generated as a result of forensic analysis of incidents, either publicly reported or identified within an organization. Government institutions and security companies provide advisories about attacks and which IOC are associated with it. This information sharing is an important part of threat intelligence. When IOC are shared with other organizations, defenders can use that knowledge to search for those indicators within their own environment to identify an intrusion. One of the simplest methods are signature based antivirus programs, which identify malicious patterns that are provided by the vendor. ENISA published in [78] a whitepaper on how incidents with a focus on ICS can be analyzed ex-post to generate intelligence to improve security posture and better understand an incident.

Not all attributes of an attack and an adversary can be assigned to technical information. In the Ukraine incident [56] the adversaries used a program called "KillDisk" to wipe victim's computers after the attack. That program has a certain hash, filename and antivirus signature (IOC) that can be used for detection after they have been identified once. The behavior (e.g. wiping victim's computers) is not easily expressed in technical indicators, but are also a sign that indicates an intrusion. They are commonly referenced to as "tactics, techniques and procedures" and covered in the next section.

[1]False-flag operations involve an adversary using indicators and behaviors of a different adversary group as diversion. This could e.g. be a Russian state actor using Chinese symbols to divert attention.

2.1.3 Tactics, Techniques and Procedures (TTP)

(Cyber) threat intelligence goes beyond aforementioned technical indicators that can be simple hashes, filenames or IP addresses. Attackers can easily change those from campaign to campaign or from attack to attack. Adversaries can change one thing only with great difficulty: their behaviour, their modus-operandi. Redesigning behavior is not done by simply registering a new domain, but often requires change of involved teams. This property makes TTP not only a good means to find adversary actions, but also to share information with others that might encounter the same adversary.

The documentation for the STIX information sharing standard [99] summarizes the duality of IOC and TTP in the following way: IOC are detective in nature and TTP are descriptive. TTP cannot be processed by machines in the same way as IOC can be. They require an analyst for interpretation (see [98] for details). In the context of adversary emulation and creating an executable attack graph both components of an attack are needed.

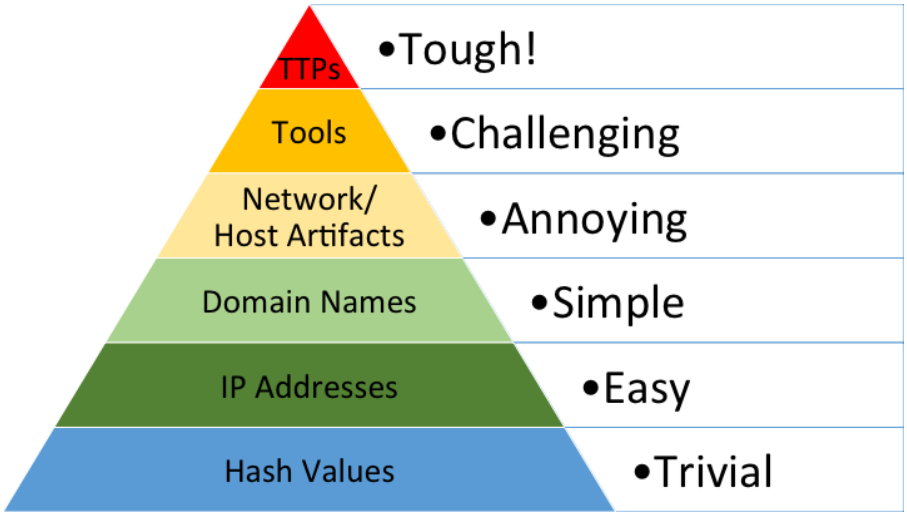


Figure 2.2: David J. Bianco's Pyramid of Pain for classifying intelligence [56]

The dilemma and value of different levels of threat information about attackers has been described by David J. Bianco in his article about what he calls the "Pyramid of Pain" [10]. This concept is commonly referenced, when discussing detection capabilities and where to focus collection of data in a target environment. It is called "Pyramid of Pain", because it originally described the difficulty an adversary has, when trying to change a certain indicator. Those indicators are part of the properties an APT would exhibit in the real world or an artificial adversary during an exercise or simulation.

NIST Standard 800-150 [42] states that "Tactics, techniques, and procedures (TTPs) describe the behavior of an actor. Tactics are high-level descriptions of behavior,techniques are detailed descriptions of behavior in the context of a tactic." Procedures are even lower-level and are often not feasible to be described to the same extent as tactics and techniques.

TTPs could describe an actor's tendency to use a specific malware variant, order of operations, attack tool, delivery mechanism (e.g. phishing or watering hole attack), or exploit. Details about tactics and techniques and their relevance for the generation of attack graphs are described in Section 2.1.5. Procedures as a lower level description of individual techniques are not directly covered in this thesis, but are represented by IOC and technique instantiations as part of Adversarial Tactics, Techniques and Common Knowledge (ATT&CK) (see Section 2.1.5).

2.1.4 Cyber Kill Chain and Other Models

Structured design of attack chains requires proper models and a common understanding. This applies not only to an attack and its components, but also to the order and goal of individual steps. Models try to describe common features and allow to abstract from technical details. When discussing adversarial behaviour and sequence of attack steps, the term "Kill Chain" provides the highest abstraction level and is commonly referred to. Figure 2.3 shows a classification and abstraction level different models have.

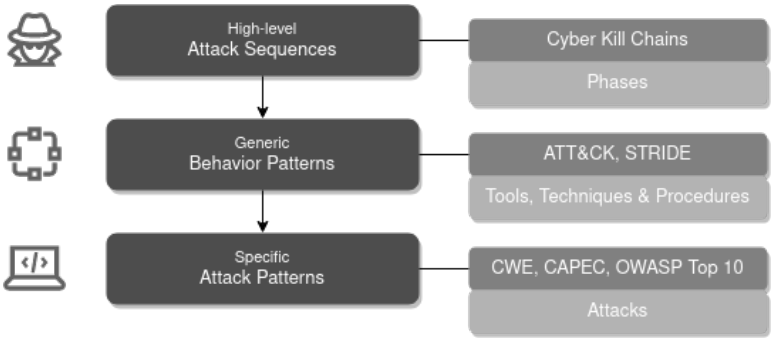


Figure 2.3: Classification of different attack models

The word "Kill Chain" has its origin in the military domain, like many other cybersecurity related terms as well. Initially used by the US navy to describe the steps of a kinetic attack, it has since been adapted into the information security domain. The first company to introduce the term "Kill Chain" in in a cyber context was Lockheed Martin in 2011 [36].

The idea of an oversimplified model like the Cyber Kill Chain has been disputed in the past and encouraged discussion among academia and industry as shown in [81]. The original Kill Chain is malware-centric and assumes a target environment with a clear perimeter. Nonetheless and potentially due to the approachability and ease-of-access, the Kill Chain became a popular model for describing digital intrusions, like the compromise of the American retail corporation "Target" [84] in 2014 or the Cybersecurity & Infrastructure Security Agency (CISA) alert TA17-293A regarding APT in critical infrastructures [20], as well an intense use by consultancies and vendors to market their products.

Figure 2.4: Cyber Kill Chain [36]

From the original Kill Chain multiple other models have been developed by researchers and security practitioners. Examples of those include Malone's Expanded Cyber Kill Chain [63] or the Unified Kill Chain by Paul Pols [81]. Each new version trying to extend or remediate shortcoming of Lockheed Martin's model.

In the context of this thesis, one extension of the Kill Chain model is especially relevant. After the Ukraine 2015 incident, Michael Assante and Robert M. Lee, who investigated that incident (see [56]), added a second stage to the original Kill Chain model. That second stage was specifically aimed towards actions impacting control system environments. A combination of Stage 1 (a conventional IT intrusion) and Stage 2 (an ICS part) was titled the "ICS Cyber Kill Chain" [7]. Adding an additional stage included special consideration for ICS environments. Where in the regular Kill

Chain the action on objectives was located in the information domain, the ICS Kill Chain does not stop with an adversary being able to act on an IT target. Having gained that capability, a second stage is available, where targeted attacks on control systems occur [7]. Detailed information on control systems can be found in Section 2.4.

One with a publication in 2006 comparatively early approach to cluster TTP and map them to a common language and model was introduced by Caltagirone, Pendergast and Best. This model was named the "Diamond Model". Later they published details in [16]. This became the basis for todays intrusion analysis and heavily influenced MITRE ATT&CK that is covered in the next Section 2.1.5.

2.1.5 Adversarial Tactics, Techniques and Common Knowledge (ATT&CK)

In this section and throughout the thesis references to MITRE ATT&CK frameworks will be used. MITRE projects and frameworks have become the de-facto standard in the cybersecurity domain when cyberattacks and attack fragments are classified and put into context. Researchers and corporations often already directly refer to MITRE materials in their work. The different ATT&CK frameworks provide a taxonomy and classifications for different areas of information security for the offensive and defensive track. Initially developed to categorize adversary behavior, it turned into one of the prime references to map and cluster attacker methodologies and the corresponding mitigative measures [101]. ATT&CK has been strongly influenced by other methodologies like the previously covered Cyber Kill Chain and the Diamond Model (see Section 2.1.4), but takes more detailed view on building blocks of an attack.

ATT&CK frameworks are intended to be used in different aspects in the cybersecurity domain. Strom et al. state in [101] as potential use cases: adversary emulation, red teaming, behavioral analytics development, defensive gap assessment, Security Operations Center (SOC) maturity assessment and cyber threat intelligence enrichment. Those use cases and the idea behind the MITRE frameworks complement the concept of generating attack graphs.

ATT&CK places itself on an abstraction level that is more granular than the Cyber Kill Chain (that does not detail beyond its seven phases. For details see Section 2.1.4) and more generic than exploits, weaknesses or attacks themselves [101]. Since it provides as appropriate level of detail and has been initially also designed for adversary emulation, it is suitable as a foundation for this thesis. Regarding the classification of ATT&CK compared to the Kill Chain or other frameworks/models, refer to Figure 2.3. Mandiant, as one of the first companies to analyze APT, even uses ATT&CK references in their yearly reports to classify adversary behavior.

The ATT&CK projects are community driven[2] and have received regular updates over the last year – first adding non-windows components and later also mobile security considerations. The MITRE ATT&CK concept was updated in 2020 to also include an own set of Techniques tailored towards ICS [69]. MITRE ICS tracks an additional 81 Techniques that have been observed to affect control systems (for details on those systems see Section 2.4). Other extensions to the original "enterprise" framework are the PRE-ATT&CK (covering tactics and steps before an actual attack) and "mobile" (covering compromises of mobile devices). Not all of these technology domains[3] have the same level of detail and maturity with regards to their Techniques and MITRE coverage.

[2]MITRE encourages information security professional to provide feedback and additions to ATT&CK. The repository is open source and incorporates many data sources/reports that are available openly.

[3]Enterprise, PRE, Mobile, ICS.

Since this thesis has an ICS focus, the corresponding ATT&CK for ICS framework will be used. MITRE ATT&CK for ICS has been the latest addition of technology domains and covers specific adversary behavior with regards to industrial control systems. This allows for specific Techniques covering, among others, Programmable Logic Controller (PLC)s, historians and industry specific protocols. For details on MITRE for ICS refer to Alexander et al. and their paper "Design and Philosophy" of ATT&CK for ICS in [2] as well as the corresponding website.

The structure of all ATT&CK matrices can be divided into three main components that are necessary to describe the TTP of an adversary. Those components do not only differ in the level of detail and abstraction level, but also in their likelihood to receive changes and updates in their future. High level components can be considered static whereas lower level components are more likely to be changed. The following listing and Figure 2.6 put aforementioned components into context.

- Tactics – the "what"
  static, contain Techniques, high level goals
- Techniques – the "how"
  rather static with minor modifications, associated to Tactics, medium level activities
- Instantiation of Techniques – the "details"
  dynamic and potentially unique for an attack, comparable to procedures in TTP

One core concept of the ATT&CK frameworks is to classify adversary behavior into Tactics and Techniques. Each Tactic has at least one Technique associated with it. ATT&CK frameworks can be expressed in so-called matrices. Figure 2.5 shows an excerpt of such a matrix. When comparing to the Cyber Kill Chain approach (see Section 2.1.4), the seven phases are comparable to the short term adversary goals in ATT&CK: the Tactics. Even though a direct mapping is not possible (e.g. Reconnaissance was initially not part of ATT&CK for Enterprise), Pols shows in [81] that similarities exist and that a "translation" can be performed.

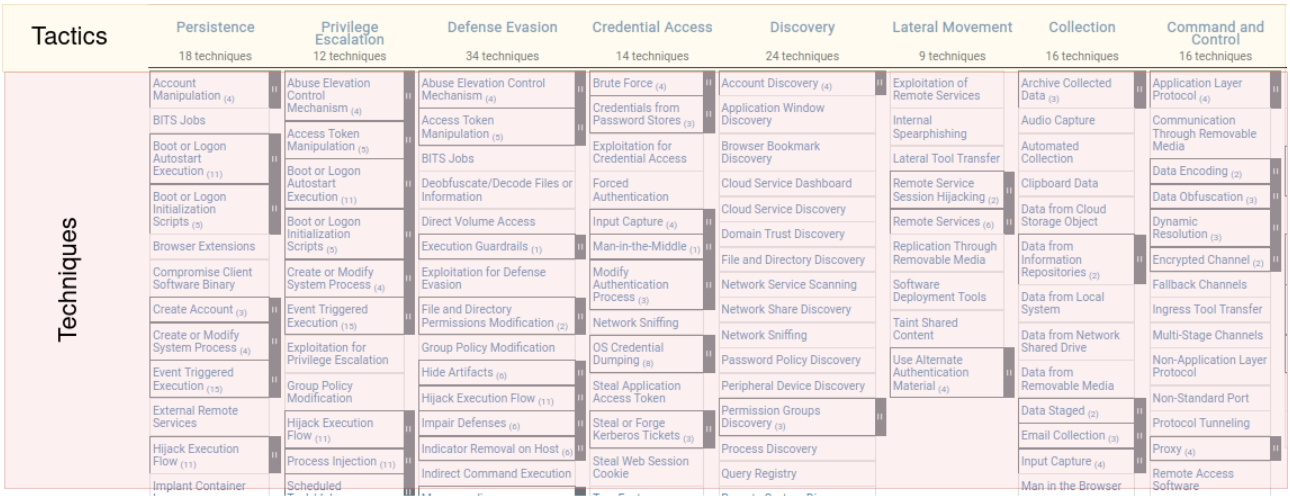


Figure 2.5: Excerpt of the ATT&CK matrix showing Tactics (headers) and Techniques (cells)

Tactics represent what an adversary does to reach their goals. Tactics are not fixed to every technology domain tough. The mobile framework differs with regards to the Tactics from the enterprise framework and the PRE framework has completely different Tactics, since it covers the time frame before an actual intrusion. The Tactics can be seen as sequential: going from outside the target environment (e.g. initial access) towards the goal (impact). An adversary

might skip certain Tactics in their attack chain. If the adversary already gained maximal privileges during the initial access, a privilege escalation is not necessary any more.

Techniques are the type of attacks an adversary might perform to reach the goal (Tactic). They can be considered a blueprint of an individual hacking activity with a certain goal. Where a multitude of different ways exist to perform certain attack tasks, they can be clustered in a manageable set of Techniques. Each Technique can be instantiated with a specific implementation. Recently rather generic Techniques like "credential dumping" are now extended. An example would be the different means adversaries might extract credentials (e.g. "/etc/shadow" file or "LSASS" memory access). Though the Sub-Techniques are not actively used in the course of this thesis[4]), they may be incorporated into the implementation of attack graphs as a later stage.

Information for Techniques' details are gathered from public reports and expert knowledge. They are an example how threat intelligence creates repositories adding value beyond individual reports themselves. Each Technique contains references to research or actual incidents that are associated with that Technique. In addition to the examples/occurrences, each Technique also contains information on how to detect an adversary trying to use that Technique and how to prevent an execution.

The different matrices of ATT&CK do not account for low level enumeration of possible vulnerabilities and weaknesses. Other frameworks like Common Attack Pattern Enumeration and Classification (CAPEC) or Common Weakness Enumeration (CWE) are better suited for this purpose and may be used to augment the Techniques of the ATT&CK model. One may also reference specific vulnerabilities according to their Common Vulnerabilities and Exposures (CVE) identifier. ATT&CK explicitly allows to include and cross reference that type of data. When trying to reproduce (instantiate) a Technique, low level data in form of specific IOC has to be available or has to be fabricated.

Figure 2.6 puts the different layers into perspective. As an example: If an attacker wants to achieve "Persistence" in a network, he may employ different techniques to maintain their access beyond a reboot or a system configuration change. Where an adversary might create their own accounts as a "Technique" to allow for persistent access, the locations and the way an account can be created often differs. Details of that account creation can be considered an instantiation of that Technique. Specific information for an instantiation, or even code itself, needs to be available or that data needs to be "filled in" by a person or machine that performs an emulation or simulation.

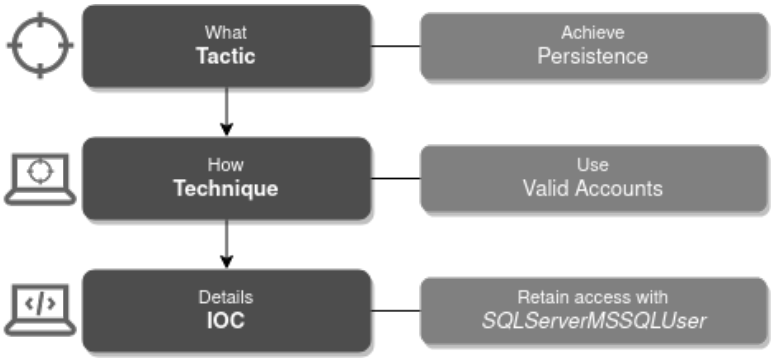


Figure 2.6: Visualization of exemplary Tactic, Technique and IOC

This idea of Tactics containing Techniques that are instantiated, will be used as the core concept

[4]The concept of Sub-Techniques has not been implemented in ATT&CK for ICS yet.

(see Section 4) of this thesis to create attack graphs. Those graphs are supposed to mimic real world attackers. Such a concept allows for enough freedom to create behavior that can be used in training and exercise scenarios.

2.2 Exercises and Adversary Behavior Execution

This section covers different types or exercises and means to reproduce adversarial behavior in a broader context of exercises. Response to todays targeted attacks is not trivial and requires targeted and appropriate actions. Organizations defending IT environments successfully have shown that proper training and exercises can make a difference. The same applies for verification of defensive measures. Trainings and tests of effectiveness can come in different forms, each having their own advantages and disadvantages. They can be targeted to individuals or are team-centric; they can focus on processes or technical skills and implementations; they can be dedicated or integrated in every day operations.

In this thesis the focus lies on an execution of adversary behavior and exercises involving emulation or simulating advanced adversaries for training and exercise purposes. Wilhelmson uses the terms Table-top Exercise (TTX) and Cyber Defense Exercises (CDX) in [111]. Since reproducing adversarial behavior is not always connected to exercise scenarios[5], a more generic term "adversary behavior execution" will be used when describing non-exercise emulation and simulation.

In cybersecurity defenders[6] are commonly called "Blue Team". A team with permission to attack an environment for training or assessment purposes with the methods of malicious hackers without causing harm is commonly called a "Red Team"[7]. The term Red Team also applies to the specific exercise involving ethical hackers attacking a target environment [75].

Exercises and simulation may contain more roles than just defenders and attackers as depicted in [77]. Among those are a few roles that are also relevant to this thesis. Depending on the type of exercise a dedicated planning team is required to decide on scenario and conditions an exercise will be executed in. During execution it can be beneficial to also include moderators and managers that can steer the adversary behavior execution, e.g. decide on when to move on with new activities. Adding dedicated observers that are not directly part of attacking or defending during an exercise allows to gather notes on actions for a debriefing phase.

Adversary behavior execution can have different high-level goals. Just reproducing adversarial behavior does not change anything in an organization or gives new insights. The goals of adversary emulation and simulation and conventional exercises are generally testing processes and technology for their resilience as well as validate existing controls (e.g. business continuity, incident response and disaster recovery). On a lower, more technical level, executing adversary behavior can also lead to identification of blind spots in monitoring, test security technology, create "malicious" events and network traffic for future use (e.g. Intrusion Detection System (IDS)) and build detection logic off attacker activity. Rosso shows in his thesis [85] how attack simulations can be used to measure performance of an SOC. Specific goals of an exercise have to be defined at a very early stage of the planning phase to match the security of an organization.

[5]Not all exercise scenarios though require an adversary to be active. Exercises can also be broader and e.g. cover pandemic effects or hardware failures and an organizations reaction to it.

[6]Usually containing cybersecurity personnel, a Computer Emergency Readiness Team (CERT), a SOC and/or forensic specialists.

[7]Ethical hackers and offensive security specialists with skills to simulate real attacker behavior and intrusive computer network operations.

In the planning phase of an exercise one of the most important decisions planners have to take, next to the objectives, is a scenario that should be executed. A scenario can also be seen as a playbook being executed over time during an exercise. This also applies for simulations without human interactions. In the context of adversary behavior execution a scenario could be an emulation of a certain adversary or a fusion of multiple adversary profiles into one scenario. Depending on the goal of an exercise, repeating the same scenario in multiple exercises limit the learnings. As an example: If defenders are always confronted with the same playbook, they will not be challenged any more and the value gained from an exercise diminishes. At the same time, it might be necessary to reproduce a certain scenario to validate improvements that were implemented as a result from a previous exercise. For details on the planning phase of exercises see the "Handbook for planning, running and evaluating information technology and cybersecurity exercises" by Wilhelmson [111]. Generating playbooks/scenarios with attack graphs is one of the key ideas of this thesis.

Organisations like ENISA[8], CCDOE[9] and NERC[10] already coordinate cyber-exercises for critical infrastructure sectors. These exercises are usually geared towards cross-industry and cross-institution collaboration, but also contain hands-on activities, if participants want to join [91]. Planning and coordination of those exercises usually take months and such preparative efforts are usually not feasible for regular organizations and companies. They can use the approaches taken in cross-national exercises and adapt the phases of an exercise to their needs.

ENISA in their "Good Practice Guide on National Exercise" [77] and Seker in [91] establish a sequence of phases during an exercise. This approach is depicted in Figure 2.7 and can also be mapped to simulations. Even though each exercise is unique, each of the phases depicted in the figure will be performed.

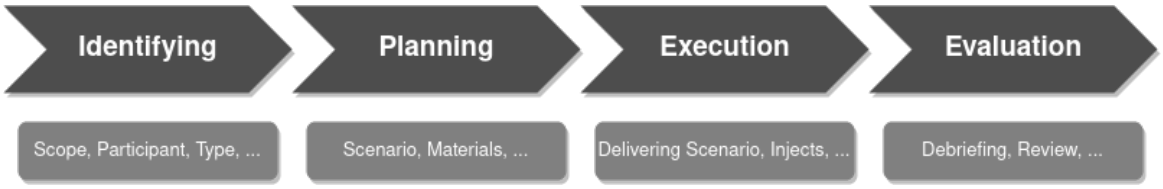


Figure 2.7: Exercise lifecycle

Before classifying different means to execute adversary behavior to improve security, the terms "simulation" and "emulation" have to be defined. Generally a clear and comprehensive distinction is difficult to make and they are often used falsely synonymous in literature. One definition is available by Malcomvetter in [62], where he states that *emulation* is "a reproduction of a function or an action [...]" and *simulation* is "imitation of a situation or a process [...]". In the context of this thesis, both imply the execution (real or virtual) of actions an adversary would take.

In an exercise situation an organization usually has to chose between simulation and emulation. A simulation of an attacker is solely virtual and is never actually carried out. A simulation is appropriate for validating and evaluating attack chains, e.g. against a model. An emulation though takes the TTP to real systems with people issuing attacks and allows for use of attack chains in trainings of defenders and attackers. In this thesis the derived attack playbooks are supposed to be suitable for simulation (e.g. in Artificial Neural Networks (ANN) or pen-and-paper exercises) and emulation (e.g. in Purple Teaming exercises) [5] [106].

[8]The European Union Agency for Cybersecurity has established "Cyber Europe" as a regular exercise.
[9]The NATO Cooperative Cyber Defence has "Locked Shields" as regular exercise.
[10]The North American Electric Reliability Corporation has "GridEx" as a regular exercise.

Emulation:

- creates real world artefacts on systems and networks
- is performed in real-time
- is limited by real world constraints
- has efforts during execution and planning; repetition requires efforts again

Simulation:

- does not create real world artefacts on systems and networks
- needs not be real-time
- does not necessarily need to fulfil all constraints of the real world
- has significant efforts during planning and can be easily repeated

Classification of exercises and emulated/simulated adversaries has been performed by Kick in [45] (for the exercise part) and by Applebaum et al. in [5] (for adversarial behavior execution). Since both approaches are covered in this thesis, the following sections will explain the different characteristics.

Besides the aforementioned distinction between emulation and simulation, a distinction of adversary behavior execution can be made whether it is performed manually (e.g. by an operator or training lead) or automatically (e.g. by a machine or algorithm). Figure 2.8 shows and classifies different means to use graphs to simulate and emulate adversarial behavior.

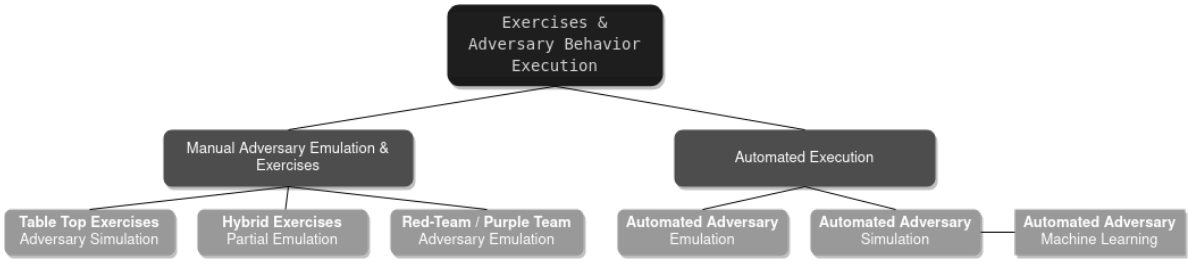


Figure 2.8: Overview Exercises and Adversary Behavior Execution

The following sections will go into detail on different types of simulations and emulations that are relevant for this thesis.

Table-Top Exercises

As the first class of adversary emulation, table-top exercises are often overlooked by more technical personnel. Table-top exercises[11] represent the "safest" way and least technical approach to simulate an attacker and try to learn from it. During table-tops participants of such an exercise[12] are presented with scenarios and have to decide on how to act in that situation. All scenarios are usually prepared upfront as injects[13] and are delivered to participants according to a playbook that has been developed by exercise planners.

Aside from little technological effort, proper planning and communication is required to perform a table-top exercise. These types of exercises are usually most suited for organizations, which have not yet established a technical capability to handle more advanced emulations. Depending on the goal of an exercise, a table-top might be sufficient to achieve them without additional technical overhead [111].

[11] Sometimes these exercises are also referred to as TTX.

[12] Participants are often management, decision makers and occasionaly also technical personnel.

[13] Injects are small parts of a scenario that are provided into an exercise to participants over time steering the course of events.

Example scenarios (not necessarily restricted to adversarial behavior though) for table-top exercises can be found in [18]. This document from the Center of Internet Security (CIS) provides an idea, how scenarios may be designed. Creating good and target-audience oriented playbooks though is a challenge for any exercise, independent whether emulated or simulated.

Advantages of a table-top approach for adversary behavior simulation are a limited preparation effort and no need to prepare technical infrastructures. Additionally, table-top exercises do not impact any systems or operations and only require corresponding participants and moderators to be held. Complex scenarios can be delivered, even if the setup in the real world is difficult to reproduce.

Disadvantages of table-tops are simulated actions and reactions as well as a limited number of participants, if an exercise is not designed to be distributed over multiple teams. A pure table-top exercise may also lack realism, if many activities have to be simulated and actions pretended. Wilhelmson and Svensson describe in [111] the crucial role of a coordinator or moderator during table-top exercises. This means that coordinative personnel needs to be available to fulfil such a role for successful exercises.

When pure table-top exercises are combined with selected live activities (e.g. analysis of an intrusion), they can be considered "hybrid" [91]. The transition between a pure table-top and a full-live Red Team (see following Section 2.2) is smooth and the amount of live activities depends on the exercise concept. In reality, especially in ICS environments, not all attack steps and activities can be realized full-live. As an example: organizations will never allow to have parts of production shut down, just because the exercise is more realistic then. Hybrid exercises allow to balance between real impacts (e.g. by shutting off a system) and simulated impact (e.g. just pretending a system is unavailable). The following section goes into detail regarding full-live exercises that also result in artefacts on target systems.

Full-Live exercises with Red and Purple Teaming

Coming from mostly simulated approaches like table-top exercises, the next step for more mature organizations are often Red Teamings. When most of the exercise parts are done in a real environment, result in real artefacts on systems and real incidents are created by defenders, then the term "full-live exercise" or "Red Teaming"[14] is used (see [49] and [91]). Sometimes, especially when a certain attacker pattern should be mirrored, adversary emulation is used as a synonym for Red Teaming [75], which is not taking into account other uses of adversary emulation.

Where in a table-top actions are simulated and theoretic responses are taken, in Red Teaming a practical aspect comes into play. Activities have to be performed (by attackers and defenders) and not solely simulated. "[Red Team] tests mimic the tactics, techniques and procedures (TTPs)of advanced threat actors who are perceived by threat intelligence as posing a genuine threat to entities." [8].

The term "Red Teaming" is derived from the team acting as attackers. That team is commonly called "Red Team" in opposition to "Blue Team" as defenders. An overview of the main teams encountered during full-live exercises (excluding planning and management team) is depicted in Figure 2.9. When boundaries between attack and defense blur and both teams work together "Purple Team" is used. The latter is described in the course of this section.

Red Teaming and specifically emulation of adversaries has close ties to military concepts as shown in Blumbergs' dissertation in [11] and by Lauder in [53]. Nicholson calls Red Team

[14] Occasionally those exercises are also called Cyber Defense Exercise (CDX).

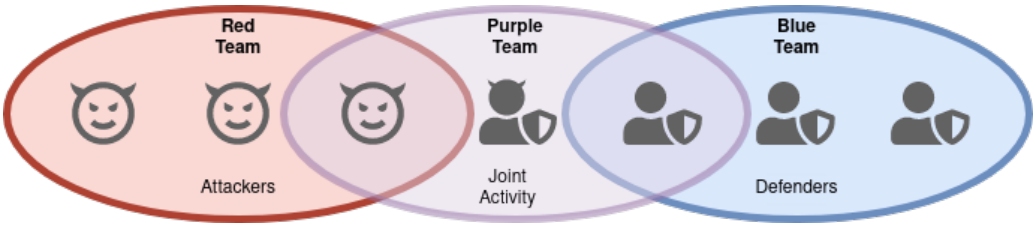


Figure 2.9: Classification of Red, Purple and Blue Team

exercises even "Wargames" in [72]. Despite its military background, the concept of "intelligence-based Red Teaming" has now even been made a part of mandatory testing required by the European Central Bank in their TIBER-EU framework for the European finance sector [8]. In addition to the finance sector other sector's companies have adopted Red Teaming as part of their overall security roadmap.

Generally, a penetration test could be considered a very limited, but thorough adversary behavior execution activity[15]. A Red Team assignment though is not a simple penetration test or a vulnerability assessment on a technical level. Such an assignment might execute a penetration test as part of adversary emulation, but the goal of a Red Team is not to find as many vulnerabilities as possible in an application or infrastructure; goal of a Red Team is to emulate adversary behavior and trying to exploit weaknesses in technology and processes to reach a target. This is often done while evading detection.

The overall goal of Red Teaming, as with any other exercise or assessment of that manner, is to improve security for an organization. In contrast to other security assessments, Red Teams are usually less restricted regarding their scope. They still have to follow a certain scenario (e.g. emulate a disgruntled employee or use techniques related to a certain adversary), but are often free to chose their way to achieve their goal[16]. Physical intrusions and social engineering may also be part of an assignment, but for the purpose of this thesis, Red Teams will be restricted to the cyber domain. At the end of a Red Team, lessons learned should be used to improve those areas, where attackers were able to circumvent protective measures.

The advantages of full-live exercises on an environment allows to validate and identify effective controls. If only table-top exercises or simulated activities are available, assumptions about system behavior and detection coverage have to be made. These assumptions can be turned into facts, when environments are attacked in reality. Red Teams also generate event data that can be used to improve detection and prevention tools. Technical details and training of defenders is difficult when activities cannot be reproduced on a system, thus a full-scale emulation of an advanced adversary can be utilized better for training low level processes and technical skills.

When comparing Red Teams to other more simulative forms of adversary behavior execution, the effort that has to be put into planning, execution and documentation is significantly higher. Higher efforts usually lead to higher costs. Another disadvantage of Red Teaming comes with the difficulty of repeatability. If an engagement does not limit the Red Team, the means to achieve their goal might differ from test to test, making comparison of to exercises difficult. Quality of adversary behavior execution correlates directly with the information available for testers and detail of techniques to be reproduced. A good Red Team requires good threat

[15]Given that penetration tests are usually scoped to analyze a certain susceptibility to a threat/adversary and tools used are often identical to an attacker's arsenal, it can be considered adversary emulation.

[16]If solely an upfront selected set of techniques should be executed with predefined adversarial behavior, freedom of the Red Team is rather limited during execution.

intelligence to thrive.

When Red Teaming is combined with direct training or integration of defenders, the term "Puple Teaming" is used [21] (see also Figure 2.9). A successful defender is often a good attacker and vice versa. This property is leveraged in a Purple Team exercise setup. Oakley also refers to a Red Team as "helpful" hackers in that context. Even though Purple Teaming is most resource intense and most demanding from a planning perspective, fusion of attack and defense allows to gain insights into flaws while training defenders and design counter measures. Scott Austin shows in [90] that a joint approach of attack and defense can be also applied in control system environments.

In [21] and [75] the concept of Red and Purple Teaming is detailed further.

In the conventional IT domain, Red and Purple Teaming has found an increased interest for organizations to determine the effective level of protection or reaction to incidents [75]. For industrial control systems the idea of simulating known attacker behavior has not been researched or implemented to the same extent yet. Existing frameworks and solutions focus on conventional IT infrastructures and do not incorporate attacks on process/OT level [46].

Performing full-live exercises and adversary emulation in industrial environments differs from conventional IT. Requirements for safety and productivity might prevent certain tests or do not allow for activities to be performed. Due to the sensitivity of those environments and sometimes the lack of a testbed, Red Team members have to closely align with operational personnel. Covert operations, where a Red Team tries to perform adversary emulation without being detected, is often not feasible in industrial environments. Additionally, often physical aspects (e.g. breaking and entering) are also included in the scope of such an engagement, since physical security controls are a significant part of security concepts in these environments. For details on adversary emulation in OT see [90] and for details on ICS see Section 2.4.

When performing security tests and exercises, either a simulative approach is chosen or simplified cyber ranges[17] have been set up for defenders and attackers to train reaction to incidents (see [114] and [59]). Only in rare cases, exercises and adversary emulation techniques should be performed against production systems in ICS environments. Some organizations even invest in full-testbeds of their OT systems, to allow attack and defense exercises. Those testbeds are also known as "Cyber Ranges" [30][114].

Even though a Red Team and a penetration tests are different from a goal and process perspective, they require very similar skills from trainers (also called operators). Staffing of a Red Team should include personnel that can not only mimic activities of an adversary, but also understand the process on how adversaries behave. Operators should be capable to "turn into" hackers with malicious intent for the duration of an exercise to achieve maximum realism and challenge for a defending team. For this purpose a Red Team should be equipped with an appropriate scenario and information about an adversary being emulated. Such an adversary for training purposes, either simulated or emulated, needs a set of techniques (the how and what) and attributes (the who and with what identifiers), which will be executed against a target. This set of techniques is also known as adversary profile or playbook.

As with table-top exercises, documentation of Red Teaming activity is not only required after adversary activity has been emulated, but also during an exercise. Most exercises have a step-by-step documentation of activities and their results that allows to reproduce and retrace activities afterwards. Aside from technical documentation, each exercise also requires appropriate after-action debriefings and lessons learned that have to be build from exercise documentation.

[17] A cyber range is a testbed or environment for security assessments, trainings and exercises [114].

Since Red/Purple Teams involve not only extensive planning, lack repeatability and incur costs for organizations, automation of Red Team activities has been researched by Applebaum et al. in [6], among others, and is covered in the next Section 2.2.1.

2.2.1 Automated Adversary Behavior Execution

This section covers means to automate activities that are covered in previous sections. Generally automated adversary behavior execution can be achieved, by removing some human element and human operators from Red Teaming approaches. Automation of table-top exercises is due to a missing moderator and potential feedback[18] not feasible. An automated Red Team with at least partially scripted attack steps and automated simulation approaches, with a potential for machine learning, are part of automated adversary behavior execution (see Figure 2.8).

Removing a human element from the equation allows for better reproducibility and better scalability, but no system can replace the human thought process (yet) when it comes to identifying and exploiting deficiencies in systems and processes. For an overview on advantages and disadvantages of automation see Oakley in [75]. Most adversary emulation automation approaches are based on expert systems with necessity to fine-tune and human, at least initial, input or feedback.

In practice full automation of adversary emulation is often not possible. Adversaries are human and to replicate human behavior by automation is still challenging. A transition from manual to automated execution is not a clear distinction. Depending on the engagement some parts may be automated, others may still require human interaction [6]. Applebaum et al. discuss approaches to even automate the planning aspect, e.g. which attack to perform next, in [6]. Their work has heavily been integrated into CALDERA as a tool for Red Team automation.

As mentioned earlier, automatic adversary emulation has seen tool support, e.g. with CALDERA[19] and Uber's Metta[20]. CALDERA is also featured in Miller's paper "Automated adversary emulation: A case for planning and acting with unknowns" [68] blurring the line between conventional Red Teaming and adversary emulation. Both tools allow to recreate common adversarial patterns, e.g. MITRE ATT&CK techniques. After defining a playbook, those tools will execute chosen activities automatically without human interaction. A planning and documentation phase though remains fully manual. Additionally, if new behavior should be implemented, it needs to be implemented in a tool of choice. For an extensive overview and taxonomy of automated adversary emulation tools, called threat emulators, see Zilberman et al. in their paper [12].

As another example: FlightSim from AlphaSOC[21] allows to generate network traffic that would be considered malicious and mimics that behavior. Since only packets are generated and no systems compromised, it would be more considered a simple simulation of an adversary than an emulation. Generated data may then be used by defenders to validate detection rules.

Input for automated adversary behavior execution is previously identified behavior incorporated in datasets like MITRE ATT&CK [12]. From those datasets scenarios are derived, which should be reproduced automatically. Those scenarios can come in different representations; one of those may be attack graphs, as they are featured in this thesis.

[18] It may be possible to use an expert system to automatically process reactions from participants to injects.
[19] https://github.com/mitre/caldera
[20] https://github.com/uber-common/metta
[21] https://github.com/alphasoc/flightsim

Hoffmann discusses in [35], how an automated simulative approach to penetration testing and subsequently also adversary emulation is possible. Hoffmann discusses two concepts: on the one hand using Markov Decision Process (MDP) and on the other hand using attack graphs. Details on attack graphs are provided in the next Section 2.3.

Machine Learning in Simulations

Using machine learning is a logical extension of attack simulation. If one is capable of properly simulating an adversary and has a properly modeled environment (or is capable to automatically derive one), multiple executions of a simulation with slight modifications while analyzing outputs and results is possible. Machine learning can use effects of scale to test the outcome of different attack and defense models. Appropriate scenarios and proper planning of experiments/optimizations are required for machine learning to provide a benefit. In the context of attack graphs, machine learning can be used to explore different graphs and attack scenarios and their corresponding outcome.

Nguyen and Reddi show in their survey [71] where deep reinforcement learning may be used in cybersecurity. One use case is in context of intrusion detection and prevention and another one for defending cyber-physical systems. Machine learning approaches from emulations are possible in theory, but individual attacks need to be countered by automated defender. Most discussed machine learning aspects are ranking challenges mostly from a defensive point, but may also be transferred to the offensive track. They try to have automatic manipulation of parameters to find the "best" results. Every organization has to decide what a best solution in their context is, e.g. best attack path, best security-cost-benefit, highest resilience, etc..

With neural networks and Adversarial Resilience Learning it is possible to optimize and learn from simulation as Fischer et al. show in [26] and Veith et al. in [107]. In [66] the idea of ranking attack graphs with Graph Neural Network (GNN) or the Page Rank algorithm from Google is covered. Lisỳ and Píbil discuss in [61] how attack graphs can be leveraged in computing the optimal attack strategy. Other works utilize game theoretic approaches in automated analyses of simulations. Since performing machine learning is not directly in scope of this thesis, the reader is referred to aforementioned cited work.

Automation and the field of machine learning receives much attention. Especially in combination with attack emulation, conventional exercises as described in previous sections might benefit from advances in those domains. Attack graphs covered in the following section are one option to formalize attacks and attack chains for machine processing and automation.

2.3 Attack Graphs

In this section, the third aspect for the background of this thesis is covered. After graphs and specifically attack graphs are explained, a selection of existing approaches are described as well as means to automate graph generation.

Graphs are a flexible tool to model structures like computer networks, state diagrams or relations and may be e.g. used to solve problems and to optimize. From path-finding algorithms over social networks to the spanning tree protocol are just a few applications which utilize the idea of directed and undirected graphs. In a survey on directed acyclic graphs for attack modeling, attack graphs are described as "combining user friendly, intuitive, visual features with formal semantics and algorithms that allow for qualitative and quantitative analysis" [48].

Graphs can be in general defined as a tuple $G = (V, E)$, where V (vertices) is a non-empty set of nodes[22] and E (edges) is a set of pairs $\{(u, v)\}$ with $u \neq v$ from the vertices. If the edges have a direction, the graph is considered a directed graph or digraph. Directed graphs allow modeling of sequences and transitions between nodes . For further details on graphs and their properties see [14]. Graph terminology used in this thesis is detailed in [48].

In [40] different attack graph use cases are mentioned: detection, defense and forensics. Even though more most applications focus on these three domains, graphs may also be used for offensive purposes like planning or modeling an attack. In the security domain (attack) graphs are use to model, represent and analyze security states and attack vectors in an intuitive way. They allow, among others, to model critical attack paths, vulnerabilities and necessary counter measures. In [76] attack graphs are used for an intrusion detection use case. In [74] and [3] attack graphs also are employed for risk assessments. [37] covers the use of attack graphs for a defensive use case.

Every intrusion/attack can be expressed as a combination of steps: a set of nodes with transitions between them. Depending on the scope and level of abstraction, these might include high-level targets like a compromise of a substation or may even go as far as a single protocol vulnerabilities or a bit-flip. In real world scenarios the combination of potential attack techniques and potential attack targets is extensive, if not unlimited, considering the number of "entities". A simplified, non-exhaustive high-level attack graph (without special semantics or syntax) for a power outage scenario is depicted in Figure 8.1 in this thesis' addendum as an example.

The idea to use graphs and trees for simulation of failures or hazards has its roots not in the information technology but in the engineering, reliability and safety domain. The concept dates back to the 60s - before cybersecurity became relevant [47]. Engineers today still use graph-based analysis to identify weak-points and possibility for failure. Concepts like the bowtie method (also graph-based) e.g. are used by engineers to analyze safety aspects and aid risk management [22]. Modelling (cyber) threats in industrial environment by using graph-based approaches are already known and increase acceptability.

Attack graphs are part of attack modeling techniques as described by Lallie et al. in their review [51]. Attack graph may be used for attack modeling, defense modeling or both [60]. The use of directed graphs to model attack chains has been a thoroughly researched approach to analyze behavior of defenders and attackers as shown in [51] and [87]. Modeling attacks as graphs and trees has been researched by Bruce Schneier in [89] and Chris Salter et al. in [88] and since then seen multiple extensions and modifications. Swiler et al. were one of the first to propose a tool to use as a graph generator and implemented a prototype in [103]. From initial attack graph research as different approaches have developed of which a small set will be covered in the course of this section. Research is still ongoing and approaches will be refined further.

Attack graphs can among other attributes be distinguished into full and partial attack graphs (see [44]). As per [16] and [76] a "[full] attack graph is the enumeration of all possible attack paths". Full attack graphs contain every single path an adversary can take, whereas partial attack graphs only represent a subset or only a single path. For the purpose of adversary emulation it is often sufficient to just use a subgraph and thus reproducing all possible attack vectors is solely a potential use case for simulations, but will not specifically be covered in this thesis.

Attack graphs are not standardized, which leads to many different solutions how they can

[22] Node and vertex can be used interchangeably. In the course of this thesis "node" is used.

be built and represented. Lallie et al. identified in their review of attack graphs [51] more than 70 different syntax and visual representations. Each of them with different characteristics and different designs. A missing standardization makes it difficult to select a graph approach suitable for this thesis.

With that understanding, the Cyber Kill Chain (see Section 2.1.4) can already be considered a generic attack graph with sequential attack modeling properties. Each phase is a node and moving from one phase of an attack to the next is a transition when displayed as a graph.

Following in Section 2.3.1 contains a taxonomy of attack graph approaches. Those approaches provide an overview off different applications and implementations. They differ in their use case, their graph model, their generation concept and more. This shows how diverse attack graphs are and how they can be used in different information security and engineering domains.

2.3.1 Selected Attack Graph Approaches

Attack graphs may be classified based on different characteristics. Kaynar proposes a taxonomy in [44] based on reachability, graph model and graph building. Reachability classification distinguishes based on how individual nodes are potentially reached. Graph model distinction classifies graphs based on how a graph is designed in its edges and nodes. Graph building considers how individual attack paths are identified and how a graph may be pruned if necessary.

A comprehensive overview of different attack graph concepts and approaches can be found in Kaynar's work [44]. A different approach to classify different graph-based attack and defense modeling can be found in [48] as a survey with focus on Directed Acyclic Graph (DAG)s.

Attack graphs in this thesis are mainly distinguished based on the graph model - when creating playbooks for adversary behavior execution reachability or graph pruning is only of secondary relevance. A classification is depicted in Figure 2.10. Each characteristic is not exclusive and many models combine e.g. hosts and vulnerabilities in their approach.

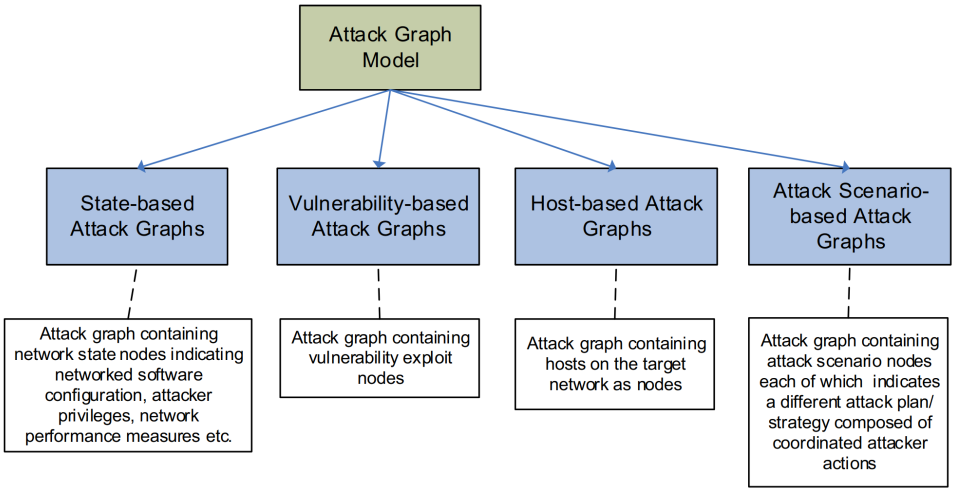


Figure 2.10: Attack graph model classification in Kaynar's taxonomy [44]

Following are three approaches to model, generate and use attack graphs. The full number of different attack graph concepts in literature cannot be covered here in its entirety and thus only a subset with relevance to this thesis are described. For comprehensive and further information refer to Kaynar and his taxonomy in [44].

Attack-Defense Trees (ADTrees) (Kordy et al.)

Extending the idea of the initial attack trees from Schneier and Salter, the Attack-Defense Trees (ADTree) were proposed by Kordy et al. in [47]. In Kaynar's taxonomy defensive nodes are not considered, but the offensive nodes create a vulnerability-based attack graph model that are based on DAG. These ADTrees do not only consider the adversarial actions, as the initial concept of attack trees did, in form of attack nodes, but also allow for countermeasures in form of defensive nodes. Defensive nodes are relevant to model success of actions and to identify room for improvement from a detection perspective and potential roadblocks for attackers towards a goal. The graphs have structural similarities to fault-trees and the initial work by Schneier on attack trees in [89].

ADTrees are built from the top with the main protection goal and branch out with protective measures and vulnerabilities/attacks. Red ellipses depict offensive nodes and green rectangles symbolize protective elements. Figure 2.11 shows how with each level the detail increases and previously detected attack techniques are specified further. Against each attack one or many defensive elements can be placed. Edges may be conjunctive or disjunctive, by default they are disjunctive. This means transitions are logically "or" connected.

ADTrees (and ADTerms as the corresponding algebra) provide a language and a hierarchic and static model that develop from a root node [47]. It fulfils the tree properties and is thus a specialization of graphs. ADTrees do not provide dependencies of attacks or a sequence of events, but traversing the tree allows to refine attack and defense approaches. By combining attack and defense in one approach, it is possible to show evolutions of security mechanisms and vulnerabilities.

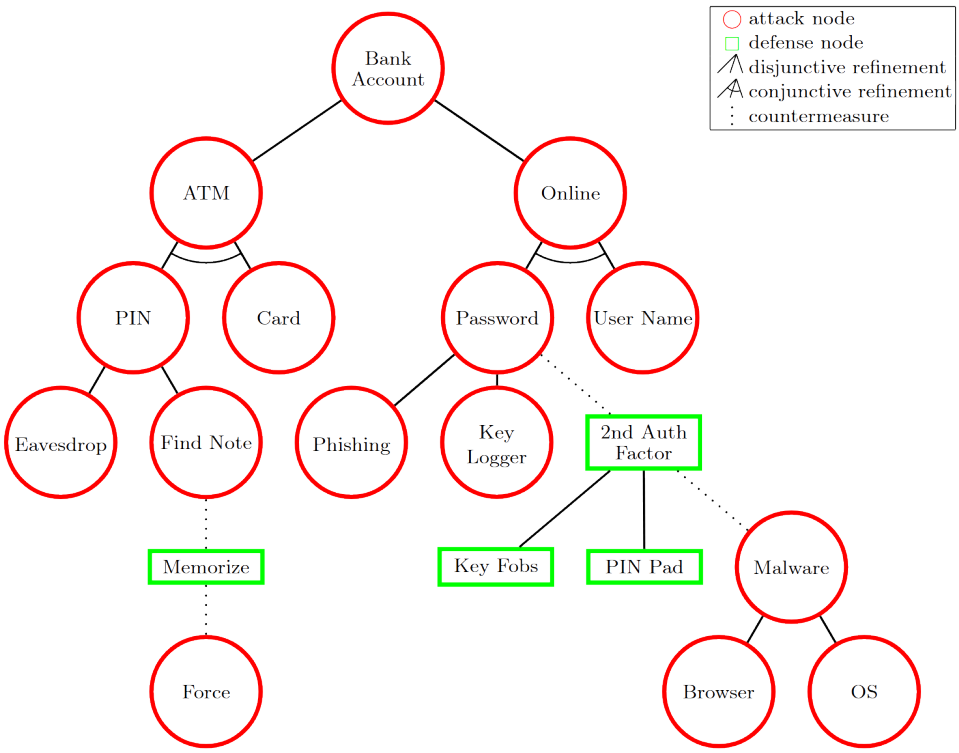


Figure 2.11: Example of an ADTree [47]

Attack Execution Graphs (LeMay et al.)

Attack Execution Graphs (AEG) from LeMay et al. are introduced in [58] and can be classified as state-based attack graphs. They take into account, how an adversary is likely to move in a target environment and uses probabilistic methods to calculate next steps and success rates. For an attack emulation in a training scenario as part of a Red Team, the Attack Execution Graphs are comparably hard to utilize. They require detailed knowledge about a target and that is usually scarce and might be changing over time.

Attack execution graphs do not consist of only a single type of node, but are built on five different elements. The top of the graph acts as the entry points and goal are at the bottom of the graph. Paths from top to bottom depict successful attack vectors. Each attack step is shown as a yellow rectangle. For each attack steps an adversary might have a certain access (red square), a certain knowledge (green circle) and skill (blue triangle). After each attack step new access, knowledge or skills might come available. An example Attack Execution Graph is depicted in Figure 2.12.

A comparably high complexity of Attack Execution Graph (AEG) allows to model advanced and detailed attack steps. Knowledge about networks and systems (and their vulnerabilities) as well as capabilities and know-how an adversary might have or gain can be incorporated into a graph-based visualization. LeMay et al. use AEG in their tool to create "Model-based Security Metrics using ADversary VIew Security Evaluation (ADVISE)" in their corresponding paper [58]. From generated graphs executable models are generated. With these models different adversary types can be simulated and the outcome calculated. As a proof-of-concept the authors simulate attackers in an ICS environment.

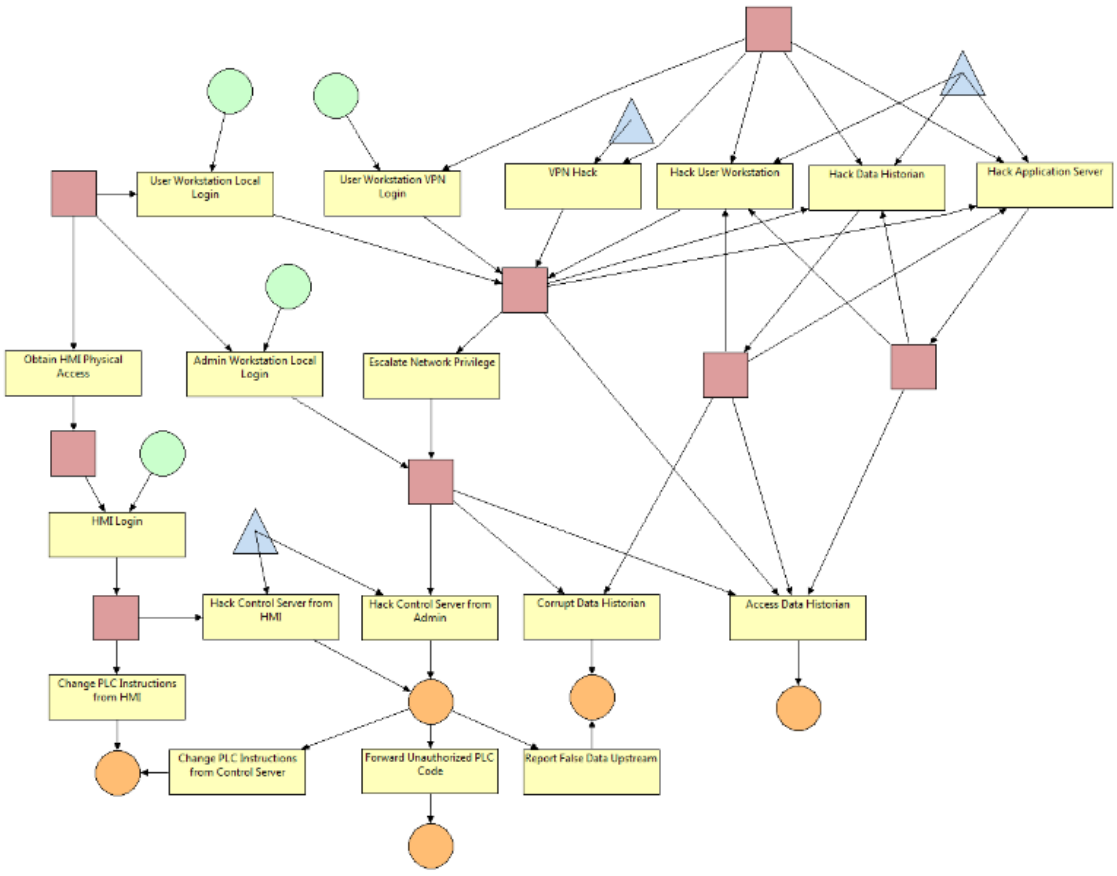


Figure 2.12: Example of an AEG [58]

Attack Graphs (Wang et al.)

Another approach to use and build vulnerability-based attack graphs is proposed by Wang et al. in [108] in order to collect security metrics based on probabilities of attacks. Their attack graphs are based on the work of Ammann et al. on graph-based vulnerability analysis in [4] and has similarities to attack graphs by Phillips and Swiler in [80]. These graphs focus on individual vulnerabilities of exploitable systems in order to achieve a goal or assess the security state of a network. That network needs to be already well-known and vulnerabilities have to be already identified in order to build a graph.

Attack graphs by Wang et al. are less diverse on elements used to create graphs, compared the previous two examples, but as in the other examples an acyclic directed graph is used. Two elements are required for attack graphs by Wang et al.: ovals are used to model exploits and plain text information to show conditions. Transitions from one exploit to the next may be limited by a condition (e.g. certain access rights required) that is indicated in clear text. In parentheses it is possible to indicate from which system to which system an exploit is acting. Attack paths are developed top to bottom, where at the top the initial access is placed and at the bottom the final target state. Next to ovals, Wang et al. propose adding probabilities that cascade down to a target node indicating an overall probability of an attacker reaching that node in that network [109]. Figure 2.13 shows an excerpt of such an attack graph.

Wang et al. proposed their attack graphs to establish a security metric that does not only provide a secure vs. insecure binary view, but allows to calculate the probability of a successful intrusion. This metric is called Attack Resistance in [109]. They make use of a bayesian networks combined with a traditional sequential attack graph. A security metric derived from attack graphs may be used to aid decision makers as well as evaluate and compare security measures. An example for such a probabilistic approach can also be seen by Cerotti et al. in [19] and their combination of frequencies of attacks based on MITRE data with an attack graph.

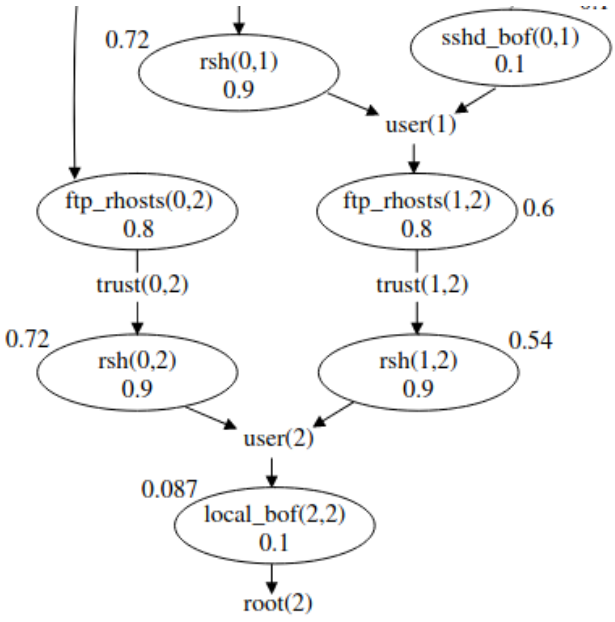


Figure 2.13: Example of an Attack Graph (Wang et al.)[108]

2.3.2 Automation of Attack Graph Generation

After covering different approaches and applications of attack graphs in the previous Section 2.3.1, following are aspects of automation, when dealing with graph creating. Creating attack graphs manually is a tedious and error prone process [92]. This is especially relevant, when dealing with complex environments, many vulnerabilities and/or complex chain of events. Another issue with attack graphs is also scalability and visual representation of bigger graphs.

Automation and optimization the process of graph generation for full attack graphs has been the focus of researchers. Depending on the kind of graph, an automation can be implemented on modeling target environment(s), producing actual attack graph(s) and analysis of resulting graph(s).

As for this thesis, an automation of generation of attack graphs is within scope. Depending on the type of graph a generation can be extremely complex and requires additional input from other sources, e.g. vulnerabilities, hosts, attack steps, states, etc.. Many automated approaches generate all possible attack paths and use custom algorithms for their graph syntax accordingly. Often those algorithms use model checking techniques as shown in [92]. Swiler et al.provide an automatic approach to generate attack graphs in [103].

Target modeling, e.g. building a dataset from a network security scan, and drawing conclusions from results, e.g. where security controls need to be placed, does not directly impact Purple Teaming and adversary behavior execution. When it comes to simulations and machine learning automation is a key component and has to be investigated further [93].

For further details refer to Sheyner et al. in [92] and [93]. When automating graph generation it is often beneficial to have a formal modeling language to describe graphs and target environments. Those languages are addressed in the next section.

Meta Attack Language and Cybersecurity Modelling Language (Ekstedt et al.)

Not all attack graph descriptions are a graphical representation. Following are two examples of modeling languages,that are per se not graphs, but can be transformed into such a representation. For description of attacks and modeling of attack graphs for defensive purposes previous research has been performed by Sommestad et al. with the Cybersecurity Modelling Language (CySeMoL) [97] and Johnson et al. with the Meta Attack Language (MAL) [43].

Both languages focus on modeling of the attacks on a high level and have a foundation in aforementioned attack graphs. This research resulted in a tool (securiCAD) to manage risks and visualize attacks from a defensive perspective [25]. This tool allows to model IT environments and then generate attack graphs based on that model. The language used by securiCAD is specified in MAL.

Specifically CySeMoL and MAL are not attack graphs syntax or approaches, but languages to describe models that can be represented as a graph. In the context of this thesis, such languages may be used to transform attack graphs for emulation into machine-readable and standardized formats.

Practical examples of securiCAD have been shown in case studies [33], where graph-based security analyses modeling attacks can be use to identify weak points. They did that, among others, by analyzing an ICS network and simulating an attacker similar to the adversary that was responsible for the Ukraine outages. Details about Supervisory Control and Data Acquisition (SCADA), ICS and aspects of industrial cybersecurity are covered in the next Section 2.4.

2.4 Industrial Control Systems (ICS)

This section covers requirements industrial control systems have, how they are engineered and how they fit into the information security domain. With an ongoing digitization of processes in production and industrial processes, operational technology OT has replaced analog sensors and controls. That created Cyber Physical Systems (CPS) and ICS.[23] An increasing number of OT components are based on technology, which has been used in the IT domain for a long time [41]. This shift also resulted in a comparable threat landscape, which affect confidentiality, availability and integrity of these systems.

Definitions of industrial control systems (ICS), cyber-physical systems (CPS), operational technology (OT) and supervisory control and data acquisition (SCADA) are not standardized and often differ, whether looked at from an engineering or information technology focus. One approach, though not covering all domains, can be found in [82]. The German Federal Office for Information Security defines ICS as a "generic term for automation solutions used to control technical processes." In the context of this thesis, OT and CPS consist of ICS and a subgroup of that are SCADA systems. The same definition is also used by Murray et al. regarding convergence of IT and OT in [70].

As an example from the electricity sector: all components involved in physical processes of electricity would be OT or CPS, intelligent devices in the field can be considered ICS and a system collecting data and allowing control is SCADA. Examples for components in ICS environments are control systems (e.g. controlling many field devices), human machine interfaces (e.g. a touch-panel at a machine) and programmable logical controller (e.g. reading values and setting set-points).

Ralph Langner, after he analyzed Stuxnet, stated in his book about "Robust Control System Networks" [52] that complex ICS network no longer have a single person that understands it fully and that many of those system cannot have accurate documentation any more. This fact shows the complexity of modern industrial and automation systems. Where in the past, relay-based closed-loop controllers managed input and outputs, programmable and computerized systems have mostly replaced dedicated wiring by now [94].

One important aspect of physical processes, like manufacturing, in comparison to mainly digital processes, like banking, is the aspect of safety. In contrast to the conventional IT world, attacks on the OT domain may result in loss of safety or impaired physical processes, like distribution of power. Organizations operating such processes need to defend against these threats. A deliberately manipulated CPS device (e.g. an industrial controller) can lead to environmental harm or even to loss of life.[27]

Even though many OT systems contain IT parts, aforementioned characteristics of ICS environments can make it difficult to directly map IT concepts to their OT counterparts. The following section will detail those aspects of ICS, with relevance to this thesis.

2.4.1 ICS Architecture

Before addressing security aspects of ICS it is important to understand the general architecture and concept in many industrial control environments. Architectures do not only influence security aspects, but also behavior of adversaries and subsequently emulation of that behavior. When modeling attacks and trying to reproduce behavior, a certain abstraction level has to be

[23] The term CPS is mostly used in Europe, whereas OT is more common in the US, though distinction blurs.

taken. A commonly accepted, though generic, approach can help in creating detailed models (see Section 2.1.5 for ATT&CK as an example).

Many different models for architectures of industrial systems exist, but they all have one goal in common: a physical process that is being serviced in the end. An overview of different reference architectures (automation pyramids) can be found in [67] and shows different approaches. In Figure 2.14 the individual layers that would be encountered in ICS are depicted as part of a PhD thesis on Industry 4.0. The closer technology is to an actual physical process, the higher timing requirements are. Additionally, the number of systems increases the closer one gets towards the process layer. When modeling target environments for generating attack graphs, this has to be considered.

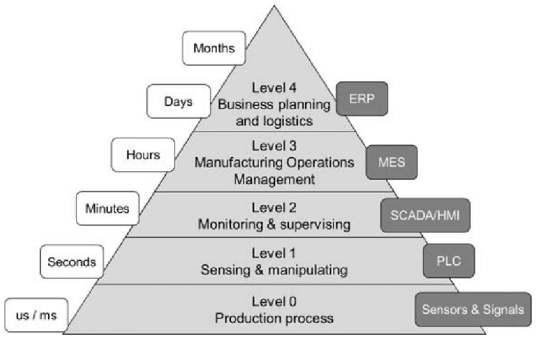


Figure 2.14: Automation Pyramid [1]

Different definitions and approaches to ICS architecture are also discussed in [82]. A clear distinction between conventional information technology and operational technology is difficult to make. Due to an ongoing convergence (see also [70]) and an increased use of information in operational processes, this thesis will continue to use the term OT for technology, which is used close to the physical process and usually segregated from the regular office network. All components related to industrial control systems will be referred to as OT. The same approach take Cerotti et al. in [19].

One of the first unified models for security in process automation have been discussed with the Purdue model in the ISA95 standard [112] that has evolved from the ISA88 standard. The current iteration, the ISO/IEC 62443 [38] series, was initially referenced as ISA99, but subsequently renumbered to the now often referenced IEC numbering. Though all models simplify and may not be fully applicable to all environments, they nonetheless allow development of best-practices and common understanding. The model used in the ISO/IEC 62443 standard is depicted in 2.15 and shows a layered approach with zones. Each zone acts as a protective element to shield assets of underlying layers from compromise from above.

Especially with today's high degree of interconnectivity, the cloud and ubiquitous presence of computing, hierarchical models as proposed in the ISA standard have to be used flexibly to be still relevant. In Figure 2.15 an exemplary setup is depicted. This setup is also commonly referenced, when discussing automation assets and when describing adversarial behavior in ICS environments.

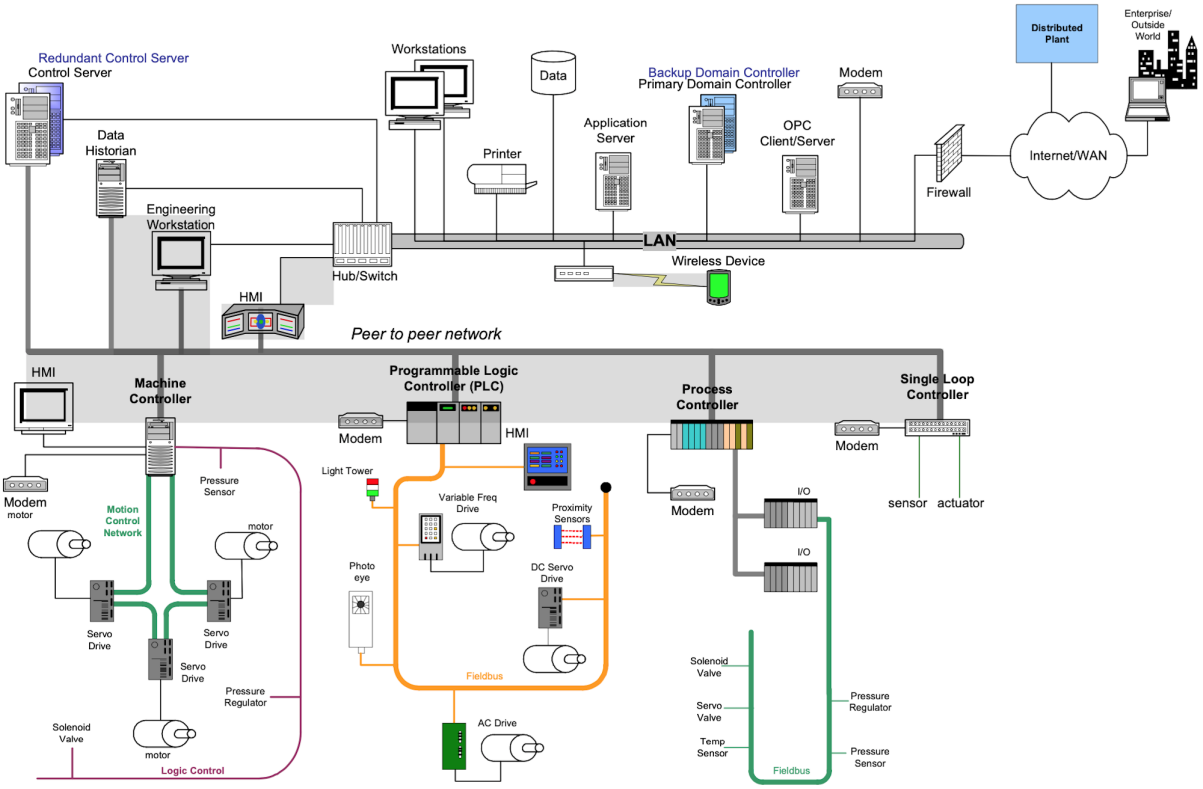


Figure 2.15: Generic ICS network layout as depicted in ISO/IEC 62443 [38]

During this thesis the reference architecture depicted in Figure 2.15 and ISO/IEC 62443 [38] will be used to describe target environments against which attacks should be simulated or emulated. Each industry and each plant is individual and thus attack graphs generated need to be adapted to a target environment. Generic adversarial actions executed against targets stay the same though.

2.4.2 ICS-Security

As stated before, ICS environments are digitalized more and more. This section covers cyber-security implications of this digitalization.

When discussing security in an industrial context many people would first consider physical security with guards, fences and locks. But with digitalization, more components are equipped with computerized functionality. IT security of those industrial computerized components has received attention increased attention and moved into the focus of defenders, researchers and adversaries.

As one example of early research (2007) in the control system's domain, the Aurora Generator Test from the Department of Homeland Security (DHS) and Idaho National Labs (INL) showed the potential for effects in the physical world by manipulation of digital controls. It abused circuit breaker to affect synchronization of frequencies to physically damage a generator with out-of-phase effects.[24][110] A generator was irreparably damaged after targeted on-off-events originating from a cyberattack.

Turk provided with [105] in 2015 already an overview about cyber incidents affecting ICS.

[24]Generators need to be synchronized, before being connected to a power grid. Failure to do so, will result in physical forces acting on the generator.

Nicholson showed in [72] "SCADA security in the light of Cyber-Warfare" how security of industrial control systems may be abused by nation states in a cyber-war context. (Cyber) Incidents received global attention were e.g. Stuxnet (disruption of nuclear centrifuges), winnti (disruption of German steel mill), Sandworm (Ukraine power outages 2015 and 2016) and Trisis (manipulation of safety controller). The Trisis incident in 2017 [23] was the first one to be publicly documented targeting the last line of process safety and may have led to aforementioned consequences. All these incidents show why organizations with industrial systems also need to protect their OT assets from cyberattacks.

From a cybersecurity perspective ICS and OT can be protected and analyzed with similar approaches. Information security focuses mostly on confidentiality, integrity and availability (C-I-A). These security objectives still apply in the OT domain, but availability usually takes precedence over the other two. In [82] Radvanovsky proposes that security objectives in industrial environments should align with safety, reliability and productivity (S-R-P). Independent of objectives, security controls need to implemented also in ICS environments to prevent compromises. Security controls include technical measures, like network segmentation, and organizational measures, like awareness and training programs. Many concepts already implemented in the conventional IT domain, have to be slightly modified and transferred to the OT domain. With increasing convergence measures can be implemented in both domains simultaneously. But not only defensive measures are becoming more and more similar, attackers and their behavior can be analyzed and reproduced in the same way.

In 2014 a compromise of a German steel mill was taken to create a defensive use case and was one of the first industrial incidents to be analyzed for defense improvement [55]. Using threat intelligence to learn from incidents and implement controls to prevent them is also covered in ENISA's paper [78].

Eric Byres states in [15] that a long time protection concept in industrial control environments was and sometimes still is an air-gap[25]. Securing by isolation and security by obscurity is still considered appropriate in some environments. But since Stuxnet (using USB for proliferation of malware) an air-gap (effective or not) is not the only security control in place any more.

Organizations today implement e.g. full-scale security monitoring solutions and require secure architectures from vendors. The BDEW Whitepaper [9] and standards like NERC CIP [26] and ISO/IEC 62443 [38] provide guidance on security controls with need to be implemented.

For further details on concepts and mechanisms in OT environments see Stouffer in [100] NIST 800-82 for a "Guide to industrial control systems (ICS) security" and Zhu's "Taxonomy of cyberattacks on SCADA systems"[115].

Due to the complex nature of cyberattacks on CPS, corresponding threat actors are usually associated with APT groups[27] (see also Section 2.1). A common cyber criminal or hacktivist usually lacks capacity and capability to perform targeted attacks on critical infrastructures. Less-targeted attacks like ransomware or collateral damage from cybercrime operations are known to have had impact on ICS systems though. Actors and adversaries that are known to operate in this domain, are either nation-state or state-sponsored groups. Attribution to certain groups, though flawed, has allowed to cluster behavior. Each group is associated with a set of TTPs typical for each threat actor (see Section 2.1.3). Among those TTPs can be a use of certain malware, prominent language fragments or IP addresses [17][72].

[25]The term "air-gap" symbolizes an approach to sever all communications with other systems, effectively isolating a system or infrastructure.

[26]https://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx

[27]Not taking into account an opportunistic threat actor not directly targeting the process, but creates disruptions e.g. by ransomware or similar.

In the conventional IT domain, networks are ideally defended by IT security professionals and incidents analyzed by forensic experts. Operators and defenders in industrial environment onsite often are engineers and not IT specialists. This makes reaction and response to cyber incidents more difficult and requires trainings and exercises not only for IT personnel, but also for engineers responsible for the OT components. One means to train personnel from both domains is adversary emulation. Technology can be tested e.g. with simulation of an adversary. At this point attack graphs generated in this thesis can improve cybersecurity in industrial control systems.

Since production environments are potentially very sensitive, they do not provide a suitable environment for testing security, emulating or simulating an adversary. Either trainings and exercises are performed against a duplicated environment or within dedicated training instances. These instances are commonly referred to as cyber ranges. Details on such testbeds for research can be found in [30]. Attack graphs can also be used in such testbeds, to aid in research to improve security posture.

The Cyber Kill Chain covered in Section 2.1.4 is itself not fully applicable for the OT domain. Since the model itself did not cover specifics of industrial control systems, Assante and Lee [7] added a second stage, and thus covering the OT side of an attack. The general concept of sequential attack techniques being executed has also been picked up in the MITRE ICS ATT&CK framework (see also Section 2.1.5). Tactics, Techniques and analyzed incidents have been collected in a community effort from many ICS professionals and resulted in a repository that can be used to improve and analyze security posture as well as emulate previously identified attacks [2]. Despite the requirement to adapt the Kill Chain approach for ICS environment, Figure 8.2 in this thesis' addendum shows, how it may be used to describe intrusions targeting industrial systems.

Approaches and considerations for cybersecurity in control environments depend strongly on the domain or sector in question. Sun et al. have analyzed cybersecurity of the power grid recently in [102]. Many of their findings can also be transferred to other sectors. MITRE ATT&CK ICS uses that property to build its repository with sector independent data [2]. This makes such a repository a good foundation for generation of attack graphs and subsequent adversary behavior execution, especially if it is combined with detection and mitigation information to increase cybersecurity in a target domain.

2.5 Background Summary and Context

Previous sections have covered multiple aspects from the information security domain. Goal of this thesis, is to generate attack graphs that aid adversary behavior execution in industrial control environments (see Section 1.1). Thus information from this Background Chapter will be used in the course of the following chapters. Where necessary, appropriate references to this chapter will be provided.

The concepts covered in Section 2.1 around cyberattacks and frameworks describing the course of intrusions (tools, techniques and procedures), are the prototype's database foundation. From these concepts required data for attack graphs is derived. Additionally, this section also aids in design and implementation of a function to generate graphs that resemble an adversary as close as possible.

Section 2.2 covers adversary behavior execution and exercises, especially Red and Purple Teaming. This includes use cases, input and output data as well as a general exercise lifecycle and workflow. This section also clarifies terminology for emulation and simulation and how those

terms can be summarized as “adversary behavior execution”.

Design and implementation of attack graphs relies on Section 2.3. It describes foundations and general concepts of graphs and specifically of attack graphs. Graphs in general are not only required for designing data structures in a prototype, but are also needed to create appropriate functions when generating attack graphs.

The last Section 2.4 on ICS helps to understand the environments this thesis’ attack graphs target. This includes challenges encountered when creating attack chains in those target environments. Additionally, specific challenges when addressing security topics in industrial environments are described.

Chapter 3

Requirements Analysis

This section on requirements begins by defining application scenarios, also called use cases, where attack graphs and a corresponding application could be used in. From identified use cases in Section 3.1 and considering theoretical foundations from Chapter 2, a set of functional and non-functional requirements are derived for the design and implementation of a prototype to answer the research question. Those requirements impact e.g. the choice of system architecture, features and other implementation aspects (detailed in Chapter 4 and in Figure 4.7). In addition to requirements on what a system should provide functionally, "soft" requirements on system behaviour and design are also covered.

3.1 Application Scenarios/Use Cases

As shown in the problem statement in Section 1.1 and the Background Chapter 2 the field of adversary emulation and simulation has yet to be put to broader use beyond the academic domain. When looking at the challenges related to implementation of those type of exercises and simulations, they are similar and support by tools is limited, especially when looking at the ICS domain.

Adversary emulation needs proper planning und support with proper tools: on the one hand to handle the complexity and on the other hand to challenge defenders with new adversary behavior (TTP) without manually creating attack chains each time. An operator can use attack graphs with corresponding adversary behavior as a planning instrument and utilize the possibility to track activities during attack execution. In the beginning is a user with a need for a playbook. Usually a user already has a general idea of what and how should be emulated or simulated. These ideas can be considered constraints for graph generation. If no constraints are given, results should still be consistent with general adversary behavior. A certain level of randomness is desireable though. Once a user has created a chain of attacks, the application allows progress tracking of execution and gives detailed information about the single steps to a user. A simplified overview on different phases use cases follow through is depicted in Figure 3.1. Independent of the specific use case, each scenario has the same basic logical flow: generating the graphs during preparation, executing the individual steps according to playbook and finally evaluating gathered results.

The program developed as part of this thesis should generate attack graphs from known TTP to be used for real scenarios (red teaming) or for simulation purposes in the ICS domain. The program should enable a "build your own advanced attacker" approach to be used by defenders and white-hat attackers. It will use input data in form of TTPs and optional constraints. From those constraints the program generates an attack graph of, at least partially sequential,

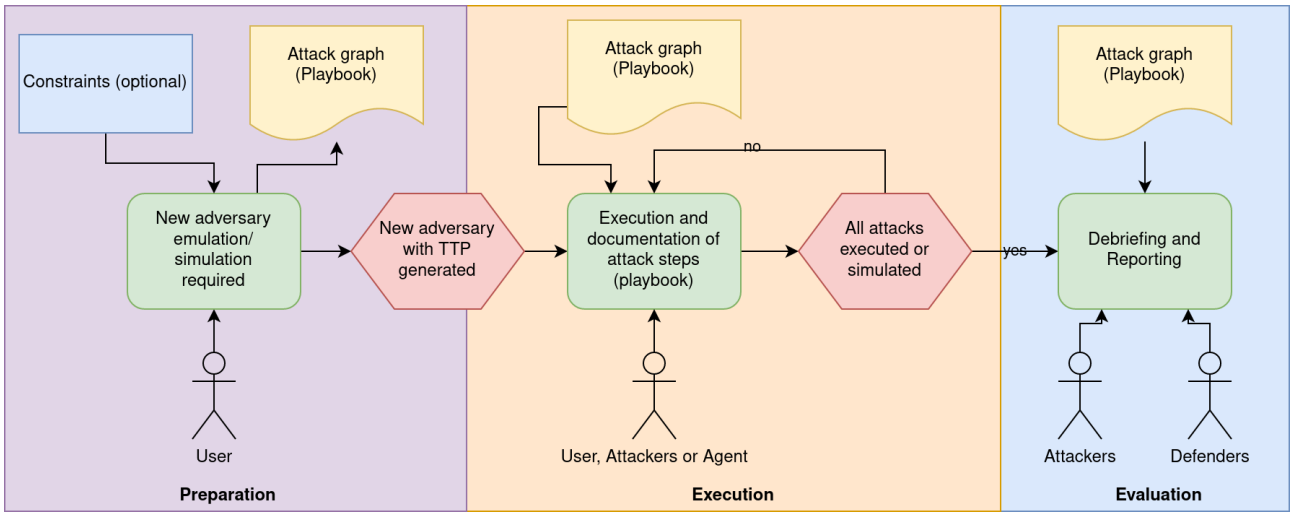


Figure 3.1: High level overview of exercise/simulation steps

techniques with corresponding attributes. This attack graph has to be an appropriate playbook for adversary emulation or simulation purposes. Resulting attack graphs represent an artificial adversary that might not have been observed in the real world yet and can be used in full simulations in the context of machine learning or as a training mechanism for real world defenders of control environments.

The overall goal of the application is to provide a basis for validating and improving security measures by supporting white-hat hackers to perform adversary emulation and simulation with attack graphs. Those use cases can be divided into two categories: the first two activities are exercise-like (emulation) and the latter two involve a primarily machine-based execution (simulation) and evaluation of results. Background information regarding exercises and the concepts around adversary emulation and simulation have been laid out in the background section in Sections 2.2 and 2.2.1.

As the five key use cases for attack graphs the following scenarios are identified. They should be supported by a tool to be able to answer the research question. Those five use cases can be summarized as adversary behavior execution (see Section 2.2).

1. Table-top exercises
2. Red/Purple teaming
3. Hybrid exercises
4. Simulation
5. Machine learning

In general all scenarios can be considered an exercise or a training: An infrastructure or a team of defenders is subject to attacks with the goal to improve security controls and/or processes.

In the beginning of every exercise is a scenario that should be trained (see Section 2.2). The same applies for Purple Teaming and table-top exercises.

Table-top exercises: "Pure" table-top exercises as described in Section 2.2 usually do not execute any actual attack steps and thus the importance of specific IOC is less. During technical table-tops the focus lies on steps an attacker would take and which targets would be affected.

Based on that information defenders can judge, whether defensive measures would be effective and whether system are properly covered. When performing a table-top exercise against an attack graph with a high number of attack paths, potential gaps can be identified. If a table-top exercise is designed as a management exercise, structured graphs may be used to identify deficiencies in incident management and escalation at a certain point in an attack chain.

Red/Purple teaming: As with table-top exercises, a trainer or operator of an exercise has to prepare and document activities. The application should support such an individual in creating realistic chain of events mimicking an adversary. For Purple Teaming (see Section 2.2) and exercise injects (events/fragments that are played into the exercise) the application has to not only provide an attack graph that models the steps an artificial attacker would take, but also has to provide details on fragments that should be created (IOC). A user requests a new attack graph, an adversary profile, in the application and based on choices and constraints a graph is created.

Hybrid exercises: Most technical exercises will involve a combination of executed attack steps with corresponding reactions and a paper-based walk-through of processes and procedures where real attacks are not feasible. This type of hybrid approach is especially relevant for the ICS domain where manipulation of process control may lead to unintended consequences and disruptions. Any person executing the steps of an attack graph should consider the effects of each action. If an attack cannot be performed it should be marked as such and where possible manually place IOC into the network or system in question. The attack graph generator does not only help as a generation utility, but also functions as a documentation tool and status indicator of individual attack exercises.

Simulation: For simulation (see Section 2.2.1) a program, called agent, has to interact with the application to generate an attack graph that should be used during the emulation. Individual attack steps need to be machine-readable and unambiguous to be used as an input for a program performing a simulation. An attack graph needs to contain enough information for an agent, a component simulating the attack, to perform the steps with no or minimal interaction by an operator. An agent can use an API program to generate and manipulate graphs. One of the critical success factors for simulation is a mapping of the target environment to a machine model. Before any simulation is attempted, the target environment should be modeled in the attack graph generator, so attacks are suitable for the simulation scope.

Machine learning: The machine learning (see Section 2.2.1) use case is similar to simulation, since machine learning may use many iterations of simulation or data from example sets. It is unlikely though for regular trainings and exercises to create enough data for algorithms to reliably infer results. One approach to learn from iterations of attack graph simulation could be, when the same graph is executed against a model of an infrastructure and each time either the model or the graph is modified, depending on the outcome of the previous iteration. An organization might be able to use that information to evaluate existing security postures and expected efficiency of protective measures. Such an approach is “adversarial resilience learning” as introduced in [26]. Generated attack graphs can be used an input parameter (and feedback) for ANN utilizing adversarial resilience learning in the cybersecurity domain.

3.1.1 User Groups

Knowledge of potential user groups of an application or a solution is key to designing appropriate interaction mechanisms and the overall functionality that has to be developed. Use cases provide a basis to identify potential types of users. When considering use cases from the previous section, potential users interacting with the application can be clustered into groups. The

following listing is not exhaustive and a user might be located in more than one group (e.g. a researcher with a background in Purple Teaming).

1. Adversary emulation operators and ethical hackers (managing and tracking Red and Purple Teaming)
2. Security analysts/defenders (simulating attacks and their responses)
3. Exercise managers and trainers (generating scenarios for exercises)
4. Information security specialists (analyzing potential new attack vectors)
5. Researchers (analyzing attacker and defender behaviour with artificial intelligence)

Based on user groups above, requirements are defined and an application has been subsequently implemented.

3.2 Requirements Specification

In the following two sections requirements are identified for an application that will aid in adversary simulation and emulation. Those requirements are divided into functional und non-functional aspects. All requirements depicted in Figure 3.2 should be addressed in the Design (Chapter 4) and Implementation phase (Chapter 5).

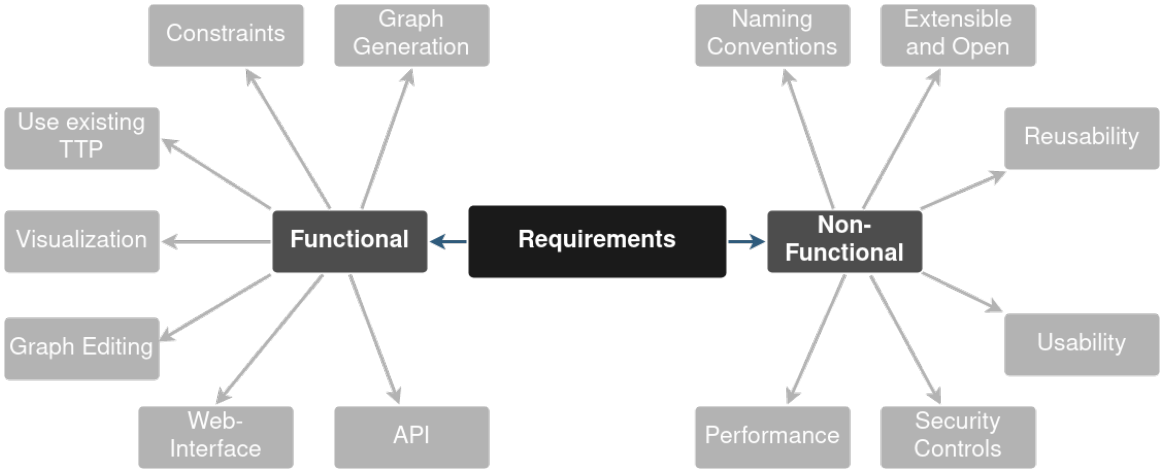


Figure 3.2: Requirements overview

3.2.1 Functional Requirements

Functional requirements describe features and functionality the application should provide. Below mentioned requirements are derived from use cases mentioned in Section 3.1.

After previous short enumeration in Figure 3.2 of key functional requirements, each aspect is detailed out in the course of this section.

Attack Graph Creation

As shown in the use cases from the previous section, the main functional requirement is to create attack graphs and corresponding attack playbooks, the sequence and details of actions that are part of an attack. Each attack graph should consist of a set of steps (nodes) an adversary has to take to compromise an infrastructure (for details on attack graphs see section 2.3). Generated graphs should be applicable to real world environments without being too rigid to allow for new adversary behavior. That way an input for proper planning and execution of emulation and simulation is provided and the overall process of emulation and simulation supported.

Support for Constraints

Data and general constraints for graph generation should come from a knowledge base that puts different steps into order and associates available targets with individual hacking techniques. Resulting attack graphs should resemble the steps a real adversary might take to compromise an ICS environment. Mapping steps to proven and accepted activities is key for valid attack chains. Thus initial data used to populate the knowledge base should be based on observations in the real world and from scenarios that have been analyzed thoroughly. For this purpose the attack graph generator should make use of existing collections of TTP (see also Section 2.1.3). A de-facto industry standard in this regard are the MITRE ATT&CK frameworks [2] (see Section 2.1.5).

As a user should be supported in generating previously unknown adversary behaviour (one of the planning steps before an exercise), the attack steps will generally be generated pseudo-randomly making use of the knowledge base. A user might specify additional constraints based on a target environment and setup of an exercise of simulation. Constraints can be for example exclusion of certain steps, targets or techniques, that are not allowed. Other types of constraints may be put on the outdegree of the graph (number of edges leaving a node) or the number of nodes per attack phase (an attack phase groups similar types of attacks). In practice this is necessary, since during an emulation only a limited time is available and the number of nodes need to be reduced to a manageable level for operators. The aforementioned constraints should be provided by a user prior to generation and should be stored for future reference and reproducibility.

Use of Existing TTP

Since many organizations already make use of those frameworks for defensive purposes, the solution should also support the ICS framework and include the publicly available dataset from MITRE. Using an industry standard collection set and naming scheme also creates a common understanding for users and other involved parties. The use of MITRE naming schema is also listed under the non-functional requirements, since it is not only a functional topic to use existing documents and TTP. Attack steps covered by ATT&CK for ICS are widely accepted and using those will reduce discussions on feasibility of certain attack steps. Despite a focus on existing collections of TTP, a user should be able to extend the existing knowledge base, e.g. when new TTP are published or when an organization identifies new behavior after investigating an incident. This function is also covered by the goal of extensibility in the following section of non-functional requirements.

Documentation and Tracking

Additionally to storing parameters of the graphs themselves, the nodes and sequence of those nodes has to be stored persistently. Adversary emulations usually take place over a period of time (see also Chapter 2.2) and over that time frame a user needs to be able to interact with graphs associated with a certain emulation. Since progress and status of exercises has to be documented, the graph, the nodes and individual results have to be stored accordingly. Stored graphs can then later be exported for reporting, copied for modifications or used in a completely different setting. For a use in simulation and machine learning use cases, graphs need to be stored such that other programs can ingest data accordingly. This might also be achieved by an Application Programming Interface (API) or e.g. by exporting data to a serializable format.

In addition to just generating attack graphs and playbooks for adversary emulation or simulation use cases, a user (in case of emulation) or an agent (in case of simulation) should be able to track the status and outcome of each attack step. If an attack step has been completed, it may be marked as such in the solution and results and comments can be added. At any time of an exercise it has to be clear, which steps have been executed and which are work in progress or have not started yet. In a post-action debriefing, the part of an exercise where attackers and defenders come together in the end, or in a reporting phase the tracking information is an important input. The application should support the processes associated with use cases from the previous Chapter 3.1: planning (generating playbooks), execution (performing and tracking the attacks), documentation (debriefing and reporting).

Visualization of Graphs

As shown in in the background Chapter 2 a Red or Purple Team operator, someone controlling an emulation or a simulation, does not only need to coordinate attacks, but also needs to provide team members (usually such exercises require a team of specialists) with an overview about the progress and the sequence of events. For this purpose attack graphs need to be available as a visual representation as well as a a list of attack steps the operator can follow. After an exercise it might also be required to perform a reporting or a debriefing, where a graphical representation may be used to show the sequence of attacks. A list of attack steps may be used to structure reporting and discuss results, mitigative and detective measures with the defenders. The debriefing and reporting phase of a Purple or Red Team exercise often requires explanation that can be easily structured along the techniques, where each technique already should contain the mitigation and detection activities required. An overall graph view allows management of the target organization to identify weak spots along a Kill Chain (see also Section 2.1.4 and Figure 2.1.4).

Graph Editing

Pseudo-randomly generated graphs as described previously, even with provided constraints, might not initially fulfil all needs of users in a given exercise environment. An emulation or a simulation might require constraints that are not included in the knowledge base or can not be modeled in such way. For this purpose a functionality to manually edit graphs is required. Individual graphs, nodes and edges should be editable without interacting with the database directly. Before or during an exercise it might be required to remove certain attacks (e.g. if they are excluded late in or after the planning phase) or change targets (e.g. when they are not available in a target organization any more). Whereas for exercise scenarios (Red and Purple teaming) a user interface is required, machine-based use cases (simulation and machine

learning) do not necessarily need a graphical interface.

Such an "edit" functionality is also beneficial for simulation scenarios, where only small modification to an existing graph are required with each iteration (e.g. to run simulations on the effects, when a component is removed or added to the target environment). A program performing the simulation can simply generate an attack graph and modify it as long as it needs. For a machine learning scenario an agent may perform modifications and simulations until the desired outcome is achieved.

User Interface

An end-user interaction and a visualization needs to have an appropriate frontend to allow interaction by operators. Especially when tracking progress, documenting results and configuring constraints, a user should be guided through the workflows. The application (primarily the front-end aspect) should be realized as a web application, so access is independent from system or device type and universally accessible by the people involved in an adversary emulation or a simulation. In addition to an ease-of-access, this allows for the option of local instances, if no collaboration is necessary.

Import/Export Functionality

A user must be able to import and export data from the knowledge base in a machine readable data format. Data then might be used in other applications or manipulated and reimported accordingly. Bulk editing might be necessary, if new environments are supposed to be integrated before a series of exercises or a set of new attacks have been collected and should be entered into the application. Aside from backups via database and application backups, an export functionality may also be used as a backup mechanism.

Application Programming Interface (API)

A user has to be able to interact with the application and data also via an API. Activities like generation of attack graphs, querying and manipulating data as well as interaction with the knowledge base should be possible without using the application's user interface. Machine based interaction is a key part for simulation use cases and thus needs to be implemented for that use case. An API is also the component that allows an integration with other programs and databases.

3.2.2 Non-Functional Requirements

Not every requirement for a tool or a solution is solely represented by functional aspects. Important aspects like usability or performance do not directly add to the visible functionality. They also need to be addressed during design and development to create a high-quality product. As shown in [29], definitions of non-functional requirements in literature are not unanimous and subjective, but the following aspects are considered non-functional in this thesis' context. In the right part of Figure 3.2 are identified non-functional requirements depicted. Each aspect is detailed out in course of this section.

Naming Conventions

As one of the key non-functional requirement is use of industry-standard language and wording throughout the application and within the graphs and constraints. Over the last years the vocabulary of cybersecurity and ICS has developed and terms regarding attacks, TTP and attack emulation have been defined. Information security practitioners use those terms to talk about offensive and defensive actions in environments. Thus it is required for a tool like the attack graph generator to utilize an existing standard vocabulary to allow a quick learning curve for a user as well as successful adversary emulation and simulation by personnel using attack graphs. As potential standards for definitions and wording the NIST standard [73], ISO 27000 norm series [39] and MITRE ATT&CK [2] can be considered among others (see Chapter 2). Initial data of the knowledge base should also follow those naming conventions to allow for an easy use, even when no custom data has been entered.

Extensible and Open

The overall application design and a prototype implementation should follow an open architecture and an extensible system design. This extensibility and openness creates trust and transparency about the application. Since adversary emulations in ICS environments are usually sensitive an operator might want to verify application code before generating scenarios for emulations and simulations. Additionally, every infrastructure and target environment is individual. Users of the program might need to tailor certain functions or procedures to suit their needs. This is also facilitated by an open design and open source.

Reusability

IOCs, Techniques and targets should be reusable. Any data entered into the application will increase the knowledge base of the attack graph generator. Therefore the number of possible attack vectors and combinations with IOC should increase. Any subsequent graph generation can choose from a better knowledge base and thus allow for better training and simulation scenarios. Especially with the current focus on information sharing in the cybersecurity community, the aspect of reusability has a strong focus. Data need not only be able to be imported from third party sources, but must also be able to be exported to a machine readable format for sharing. Previous requirement of extensibility and open architecture should also add to the reusability of functions of the application and backend.

Correctness and Error-Tolerance

Generated attack graphs must be syntactically correct and should be, as far as with the knowledge base possible, also be semantically correct. Results of the attack graph generator should resemble as close as possible a real-world adversary. If a user provided constraints for sequence and content of single attack steps, they must be adhered to. The application must always keep the integrity of data, even if errors occur or multiple users are handling the same database set. Erroneous input and deliberate attempts to abuse the application should be handled gracefully. Errors in the application should be prevented as far as possible and caught during operation by appropriate exceptions.

Usability

From a usability perspective users should be guided through process of graph creation and utilizing attack graphs. All user interfaces should be as simple as possible with an option to show advanced features on demand. An interactive help is is not included in the scope of this thesis though. Expert systems are aimed and designed to be used by experts. An operator should encounter familiar interfaces and understand functionalities available. Where possible invalid inputs should be flagged as such and do not result in unexpected behaviour and errors. Error messages in general should be understandable and not break the application's flow, e.g. with an ability to turn off stack traces on application error. An administrative interface may be more complex and less guided than main application interfaces, since backend modifications are usually performed by experts that know application and backend.

Performance

Another very common non-functional requirement is performance. In this thesis performance is only of secondary interest. The application only facilitates emulation and simulation by providing attack graphs and playbooks and thus does not need to fulfil real-time timing constraints or allow for thousands of users with simultaneous access. The performance aspect can be reduced to a usability point-of-view. Operators should not be impaired in planning and tracking of their work by long wait-times. Complex graphs may take a longer time to render, but excessive parameters for graph generation should be prevented by the application.

Security Controls and Authorization

The non-functional aspect of appropriate security controls is relevant for an application used in attacker emulation. Handling sensitive information and storing data about exercises makes a tool itself a potential target for attacks. Security itself though is a property that is hard to verify. One aspect that can be expected by a user are effectiveness of security controls.

Independent on how the tool is deployed, it must not be possible to compromise the system or the data by exploiting weaknesses in code or application logic. Safeguards should be in place to filter potentially harmful inputs, out-of-range values and attempts to bypass authorization and authentication. A user should not be provided with detailed stack traces containing potentially sensitive data, but only with error messages detailing the type of violation, if appropriate. This is also supported by a good error handling as required earlier. Data should not only be sanitized upon entry, but also when provided back to a user. A defense-in-depth approach allows to securely operate the application, even when single safeguards fail. Despite filtering malicious inputs, an attack graph generator should allow entry and storage of potentially malicious files and documents that are used as indicator of compromise and attack fragments.

A subset of security is authorization and authentication. The application does not need to realize a fine-grained and complex authorization, authentication and permission scheme in the prototyping state. Users have to log in, but every user has access a the full set of features and all attack graphs. During adversary emulation only a small set of ethical hackers is active at the same time. Though more granular permission management may be required at a later version, the application of this thesis will not implement such features. Any authentication mechanism implemented must prevent an anonymous user access to data and attack graphs, since they might contain confidential data about attacks and weaknesses. General aspects of the application may be accessed by an anonymous user, but once interaction with graphs or graph generation is attempted a valid login is required.

Requirements, functional and non-functional, from this section are basis for design decisions that are covered in Chapter 4 below.

After identifying requirements in previous sections, a prototype for a graph generator can be designed. This design is covered in the following Chapter 4.

Chapter 4

Concept and Design

This section covers taken design decisions to create a graph design and a prototype for answering the research question from Section 1.1. To positively answer the question, attack graphs have to be designed in a fashion that they support adversary simulation and emulation. A prototype application has to be created supporting this activity.

Before implementing a prototype to support adversary behavior execution, a concept and design has to be established for the aforementioned aspects that specifies how requirements from the previous Chapter 3 are met. The following sections cover concept and design decisions.

Because ATT&CK is a well known and widely accepted framework, this thesis is highly aligned with data formats and other resources supplied by MITRE (see Section 2.1.5) and ontologies like Structured Threat Information Expression (STIX). They can be combined to a concept of Tactics containing Techniques that are instantiated. As a core concept of this thesis, such an approach allows to create attack graphs mimicking real world attackers and maintaining common nomenclature. This concept closely aligns with already proven concepts in the information security domain. It allows enough freedom to create adversary profiles that can be used in training and exercise scenarios. That way existing collections of intelligence about attacks on ICS can be used, while still allowing enough flexibility to create "unique" attack chains.

Attack graphs, the prototype and the underlying database are designed to be part of an expert system that is maintained and operated by experts. An overview about potential users (experts) have been identified in Section 3.1.1. The application's design is not aimed at someone without any previous knowledge on adversary behavior execution or TTPs, but should support individuals that are already active in those domains.

The process and anatomy of attacks have been researched extensively and different models and ontologies have been developed (see Section 2.1). Many of the approaches focus on an retrospective analysis of attacks to derive protective measures. For the purpose of this thesis, this point of view is changed. Core idea is to use existing knowledge of attacks and their properties to develop new potential adversary profiles that can be used to improve detection and response capabilities. Attack graphs that are used to allow and support adversary behavior execution, enable such a change in point of view.

Generation of attack graphs has been shown to be (semi-)automatable (see Section 2.3.2), but approaches are primarily designed to create graphs that cover all potential attack vectors and require detailed knowledge about target environments. Early research on graph generators is available in [103]. Swiler et al. show how an attack graph can be created by analyzing network topology and available exploits. They take a similar approach as this thesis, but utilize a different detail level (specific exploits) and consider different data sources (see Section 4.3.4).

Lippmann et. al introduce "NetSPA" as a tool in [60]. It can be considered similar to Swiler et al.

from a goal and level of modeling point of view. It assumes comprehensive knowledge about target infrastructures and generates graphs based on network topology, system states and known vulnerabilities.

Wynn et al. describe in [113] their tool "CICAT" that allows to model and simulate how an adversary can perform cyberattacks on a target environment. CICAT was originally designed to evaluate and improve computer security in nuclear facilities and can be ported to other ICS environments. This project has show to successfully automate production of cyberattack scenarios.

Existing solutions do not fulfil the requirements given by adversary behavior execution use cases from Section 3.1. Complex, all-encompassing graphs and thorough knowledge of a target environment is likely not present and an exercise is will not to cover all potential attack paths. This is valid though, since an adversary also would not fully explore all paths. Thus a design of graph and prototype needs to compensate aforementioned issues.

Existing approaches to graph generation as summarized in Section 2.3 fail to address the specific needs of adversary emulation and simulation. Depicted approaches are retroactively analyzing incidents or are meant to model protective measures. Where applicable, different aspects of existing approaches are incorporated in the solution of this thesis. The approach and application should not only generate attack graphs, but also support an operators in tracking and reporting of attack graph execution. Details about differentiation to existing graph approaches are available in Table 4.2 in the next section.

The design of attack graphs should generally be easily understood and its visual representation should support previously identified use cases. Highly complex graphs increase the effort to perform adversary emulation. In a simulation context, this property is negligible. Details on graph design decisions are provided in Section 4.1.

Since adversary profiles should be represented and subsequently emulated or simulated, full attack graphs (indicating all possible paths and possibilities) are not feasible. This thesis' graph design focuses on partial attack graphs that only represent a subset of attack combinations. This selection though is generated from the knowledge base. That way a realistic adversary profile is available without being overwhelming for exercise scenarios at the same time. In theory, it is also possible to generate full attack graphs. Due to complexity and computing requirements, this is not recommended.

To create graphs fulfilling desired properties, a forward generation approach (top to bottom) is chosen. During generation nodes at the top are generated and populated first and further steps in the attack chain are added towards final activities (impact on objectives). This approach allows to easier control size of a graph and dependencies. Additionally, it follows a common sequence when discussion intrusions (see Cyber Kill Chain in Section 2.1.4).

Not only attack graphs should support adversary behavior execution activities, but a prototype supporting graph generation should also follow the lifecycle of of an exercise or simulation (see also Section 2.7). It should guide an operator through the planning phase (selection of attributes and constraints), aid in the execution phase (tracking results and progress) as well as provide an after-action overview about final results. The overall prototype design does not encourage complex graphs or all-encompassing attack modeling, but generates actionable graphs to help a Red Team operator or scenario planner to create and implement scenarios and playbooks.

In Table 4.1 a mapping of previously identified requirements and design aspects of a solution is available. Many requirements are covered by more than one design and implementation aspect. In the course of this chapter key design decisions, as they have been mapped in Table 4.1 are discussed in detail.

Requirement	Addressed by
Attack Graph Creation	Graph design (syntax & semantics), Generation algorithm
Support for Constraints	Database design, Graph generation, Data sources
Use of Existing TTP	Database design
Documentation and Tracking	Application design, Database design
Visualization of Graphs	Graph design (syntax & semantics), Application design
Graph Editing	Application design, Graph design
User Interface	Application design, Graph design
Import/Export Functionality	Application design, Data sources
API	System architecture, Application design
Naming Conventions	Data sources, Database design
Extensible and Open	System architecture, Data sources
Reusability	System architecture, Data sources
Correctness and Error-Tolerance	Application design
Usability	Application design
Performance	Application design
Security Controls	System architecture, Application design
Authorization and Authentication	System architecture, Application design

Table 4.1: Mapping requirements to design decisions

4.1 Graph Design

This section covers the graph design used in the prototype. This includes syntax and semantics of generated attack graphs and a comparison to existing graph approaches. Addressed requirements from Chapter 3 with those decisions are a graph generation approach, a possibility to edit generated graphs and means to visualize results. Individual elements of attack graphs (e.g. nodes, edges, etc.) for a prototype need to be defined before they can be implemented. Generated graphs have to be suitable to support adversary emulation and simulation; this includes planning of exercises/trainings, tracking the execution and perform after-action-briefings.

Before developing a function for generating graphs, semantics and syntax have to be defined. Approaches of different attack graph solutions described in Chapter 2 each have their own ways to formalize and build their graphs. As one example, Attack-Defense-Trees even go so far as to enforce a requirement that tree properties have to be fulfilled as shown in [48]. Such a restriction is not feasible for the implementation in this thesis, since proposed graphs will potentially split and join and thus violate the tree property. Graphs generated by the tool of this thesis have to primarily fulfil the graph-property and limitations regarding syntax and semantics.

Parts of previously covered attack graph syntax and semantics have similarities to Petri nets. This is especially visible considering concepts of Meta-Techniques (see Section4.1.4) that allow conjunction and disjunction of attack paths. Techniques represent places in Petri nets, results represent transitions, and Meta-Techniques represent special transitions that require more than one condition to be fulfilled. Considering previous mapping, similarities between Petri nets and attack graphs can be established. A “result” of an attack graph’s final state is only required during debriefing and complex states are not needed. Petri nets have thus not been chosen for the implementation in this thesis.

When considering Kaynar’s taxonomy in [44], attack graphs and a prototype to answer the research question currently do not exist in a level of detail that is suitable or fulfils requirements

from previous chapters. The next Section compares graph approaches and which attributes can be incorporated into attack graphs for adversary behavior execution. To maintain manageable and visualizable graphs an adequate abstraction level has to be chosen. Since MITRE ATT&CK is the foundation of the knowledge base, components of attack graphs utilize that. Figure 4.1 depicts how a graph might look like. Each node is an instantiated ATT&CK Technique and edges represent results of those techniques. This can be considered a translation of MITRE frameworks into graph-based representation.

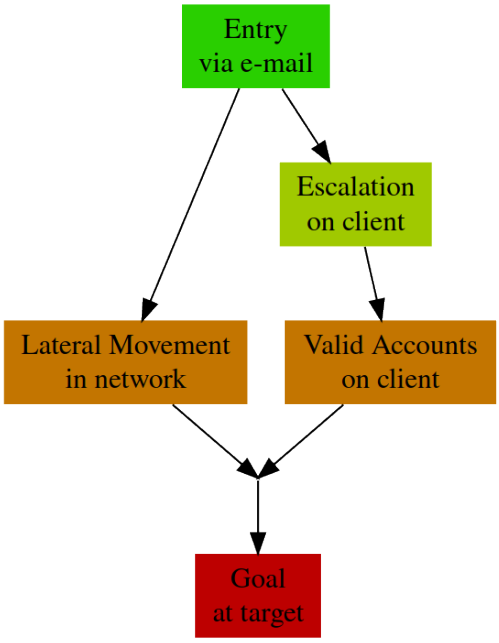


Figure 4.1: Graph design example – from initial access to final action on objectives

4.1.1 Distinction to Existing Graph Approaches

As shown in Section 2.3, using graphs to model attacks has already been covered extensively in research and previous work. Kaynar [44] and Lallie et. al [51] have shown existence of many different approaches for attack graph syntax and representation. Lallie points out a missing standardization, which is why so many different approaches have developed. Even though other graph approaches are inspiration for the attack graphs in this thesis, they either lack key details and attributes or have a completely different goal. As stated earlier, none of the attack graphs covered in the Background Section 2.3, can fully fulfil requirements stated in previous chapters.

A key concept for attack graphs in this thesis is an assumption that no or limited information is available about vulnerabilities present in a target environment during the graph creation process. Other attack graph concepts, like ADTrees (see Section 2.3.1), calculate possible attack paths and countermeasures based on available targets and their associated vulnerabilities, while often exploring all paths and options. Since attack graphs related to this thesis should be used in adversary behavior emulation and simulation, this includes validation of existing security measures, generating graphs does not need to account for existing state of weaknesses and protections, but needs to focus on generally valid chains of Techniques. Generated graphs only represent a small subset of an exhaustive exploration of all attack possibilities against targets.

Attack graphs of this thesis do not need to maintain a full state of individual nodes (e.g.

compromised or credentials harvested). Where other attack graph implementations (see Section 2.3) require a state to be tracked during execution, the implementation used in this thesis instantiates new nodes without considering success or failure of previous attack steps. This allows to construct graphs solely based on a knowledge base while excluding variables that are only available during or after graph execution.

Table 4.2 shows a comparison of this thesis' approach with three attack graph concepts covered also in Section 2.3. Where other solutions require detailed information to build full attack chains, this thesis' solution focuses on generic domain knowledge to combine just a subset of possible attacks into an actionable graph for adversary behavior execution.

	AEG (LeMay)	**ADTrees (Kordy)**	**Attack Graphs (Wang)**	**Emulation Attack Graphs**
Source	[58]	[47]	[108]	*This thesis*
Goal	Attack model solving	Attack and defense	Security metrics	*Behavior execution*
Knowledge	Full domain knowledge	Full domain knowledge	Full domain knowledge	*Minimal knowledge*
Scope	All paths	All paths	All paths	*Subset of paths*
Sorting	Top to bottom[1]	Bottom to top[2]	Top to bottom[1]	*Top to bottom[1]*
Centricity	Exploits & conditions	Asset/protective goal	Attack paths	*Attack techniques*
Type of graph	Graph[3]	Tree	Graph[3]	*Graph[3]*
Nodes	Multiple[4]	Attack/Defense	Exploits/Systems	*Attack step (Technique)*
Edge	Possible Transition	Relationship	Transition next exploit	*Result of node*
Generation	Deterministic	Deterministic	Deterministic	*Pseudo-random*
Complexity	High	Medium	Low	*Low*

Table 4.2: Comparison of different attack graph implementation and this thesis' solution

4.1.2 Graph Syntax

This section will cover the syntax attack graphs in the prototype have to follow. To keep graphs manageable an approach with only two building blocks is chosen. Derived from general considerations from Section 2.3, attack graphs in this thesis have to be *directed* (attack chains follow a sequence), *acyclic* (prevention of loops/recursion) and *without multi-edges* (performing the same attack/transition against a target twice is not realistic). *Weighted* edges can be used, when expressing probabilities or cost (e.g. in simulations). The prototype does not utilize any additional data mapped to edges, but in Section 8 extending the graphs to Bayesian networks is added to future work considerations. This would require adding additional data to a graph's edges.

As shown in Section 2.3 graphs consist of nodes and edges. Syntactically, each node may have *one or many inbound edges* (transition/attack result for the node) and *one or many outbound edges* (transition/attack result for the succeeding node). It is also possible to have no outgoing edge, when the node can be considered to be a dead end or activities end with such a node. Loops and self-references (strictly also loops) may not occur. A primary constraint on graph syntax is that edges have follow into nodes and that edges follow from nodes. A differentiation between different node types is not done – by association with a Tactic a hierarchy can be established. In the prototype's implementation a Technique (node) instantiated for a certain Tactic may not link to an earlier Tactic (effectively creating a loop in this case).

Existing approaches use nodes to represent systems or assets that are acted upon. A formal algebra, like other graph approaches (e.g. AEG) and languages (e.g. MAL) include, is not within

[1]Directed/Chronological/Sequential
[2]Static/Hierarchical
[3]Acyclic directed graph
[4]Knowledge, Access, Skill, Goal

scope of this thesis. This also includes maintaining state of an adversary during intrusions. Though this allows to infer success possibilities or even probabilities of execution, complexity implications of such an approach would go beyond set thesis' scope.

A syntactically simple approach for graph visualization with single type nodes and one type of edges, is sufficient to fulfil requirements of a playbook-like attack graph. Previous knowledge about vulnerabilities or detailed knowledge about targets is only required for simulation scenarios. Interpretation of graphs for emulations is up to a Red Team operator or trainer.

Aforementioned syntax needs to be represented by a datatype in the prototype implementation. Details on this representation are provided in the next section.

4.1.3 Graph Data Type

Given the structure of attack graphs, a set of nodes with few edges and sequential layout, the tool of this thesis will utilize double-linked lists to store graph information and relationships in a database. Each node contains information about successors and predecessors. Another very common data structure is an adjacency matrix, but since only few edges are required, such a matrix does not fulfil requirements best. Graphs loaded from a database are transferred into NetworkX data structures (see also Section 4) for easier interaction with nodes and edges as well as a possibility to perform actions on graphs.

Storing graphs directly, as graph structures, in a database (e.g. as Javascript Object Notation (JSON), Graph Modelling Language (GML) or GraphML) is generally possible, but not preferred for the prototype. The main use case is utilizing graphs in adversary emulation and simulation (including result documentation and tracking) and single NetworkX graphs are not designed to contain such additional information. The attack graph tool does also not utilize a graph database. A dedicated database meant to store large graph datasets, like Neo4j or OrientDB, is suitable for storing graphs, but does not fulfil additional requirements for a database and results in overhead. In addition, attack graphs are comparably small and dedicated graph databases are optimized for large graphs. Thus, storing a graph as a double linked list in a conventional database is chosen and represents a versatile approach for this thesis. Due to comparably small graphs, a conversion to other data structures or even a duplication in a dedicated graph database can be achieved with little overhead, if necessary.

4.1.4 Graph Semantics

When using attack graphs for adversary behavior execution, focus should be placed on individual attacks to be emulated not on assets being attacked. Due to this, nodes in attack graphs of this thesis represent specific attack steps (Techniques). Usually attacks follow a sequential pattern and each attack is followed by another attack of the next Tactic. Once access to a system is gained, further penetration into a target network is attempted. This aligns with the Cyber Kill Chain and other models described in Section 2.1. Dead ends or dangling nodes in generated graphs may occur though and are not flaws in graph generation itself, but rather result from an imperfect knowledge base and pseudo-randomness during instantiation. Techniques detached from a main graph also occur with real adversaries that might perform an attack and do not follow up on it or previous activity leading to a Technique is not known. Though it is possible that an attacker will reattempt the same Technique on the same Target more than once, it does not need to be represented in an attack graph.

Semantics of edges have to be differentiated, because emulation has a different result set and goal

than simulation. During emulation, edges represent results of performed actions. Effectively this is realized as an attribute for each Technique. Per default, attack graphs contain all edges that have been initially generated for an adversary profile. This can be considered an artificial APT's profile. During graph execution, edges might be converted into dotted or dashed styles. This means a successful attack is depicted as a solid edge, a partially successful attack is indicated by a dashed edge and if an attack has failed an edge will be dotted. For simulation such a representation is only partially necessary. For certain types of simulations though, it could be required to work with probabilities, indicating success rate of a certain attack and likelihood of transitioning to another Technique. A "likelihood metric" that can be used to describe those properties has yet to be developed. In this case the result of each simulated Technique might be augmented by a success likelihood.

As with other attack graphs discussed in this thesis, a sequence of events or attack steps must not be arbitrary, but has to fulfil certain constraints given by the nature of an attack, e.g. credentials need to be available to log on as a valid user. Another potential constraint is enforced by the logical course of events. These constraints are maintained in the knowledge base. Such restrictions, implicitly enforced by the data model, can be simple rules on sequence of hacking steps or can be as complex as multiple dependencies acting as prerequisite for a Technique. Well modeled and realistic dependencies with potential predecessors and successors for certain Techniques are key for attack graphs that mimic real world adversaries. Details about adversary behavior execution are depicted in Section 2.2. Extensive review and dependency modeling is beyond scope of this thesis, but exemplary modification to an initial dataset with corresponding constraints will be performed in the prototype implementation. Prototype constraints are mainly based on sequences of Tactics.

During execution of emulation or simulation, usually all Techniques of an attack graph are executed. This allows to also test for cases, which are known to not succeed. For an evaluation of an emulation it can be important to see though, whether prerequisites were completed successfully. This would then show room for improvement or critical attack paths that are broken. Edges indicate potential transitions an artificial adversary might take, but do not necessarily mean they have to be successful each time during simulation.

The easiest form of subgraph is a simple sequence of Techniques without additional branches and without any dependencies. In contrast to graphs with dependencies, sequences do not require a previous execution of a certain Technique. Each Technique has a set of possible Successor-Techniques in the knowledge base assigned to it. The following graph in Figure 4.2 shows such a simple sequence of Techniques. The following figure 4.3 shows a more complex sequence with branching. When expressed in logic notation it can be written as $T_1 \rightarrow T_2$ respectively $T_1 \vee T_2 \rightarrow T_3 \vee T_4$. This construct may be used to model a course of attacks where predecessors and successors to a Technique are not hard requirements and during graphs construction not each path needs to be present in contrast to the following conjunction and disjunction constructs.

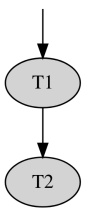


Figure 4.2: Simple sequence

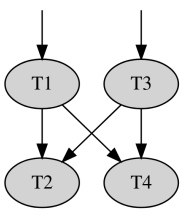


Figure 4.3: Composition/Interconnection of Sequences

Techniques with more than one dependency or more than one effective result can be represented in the knowledge base with a Meta-Technique. Those do not represent an actual action an attacker has to perform (or simulate), but act as a pivot point to merge or split to attack path as dependencies. As early as Schneier's attack graph concepts [89], conjunctions and disjunctions have been introduced. Attack-Defense-Trees [48] also contain special notations to indicate mandatory prerequisites for nodes. These ideas have been used in this thesis to establish Meta-Techniques.

A Meta-Technique is not part of the MITRE ATT&CK data model[5] (see Section 2.1.5), but is solely created to allow for modeling of complex dependencies and prerequisites of Techniques. They allow to maintain the general structure of the MITRE framework and still be able to model attack chains with constraints from the knowledge base. For this generation algorithm a Meta-Technique can be handled similar to a conventional Technique. During graph rendering and evaluation Meta-Techniques are analyzed and used as a means to join or split existing attack graph paths.

Those Meta-Techniques are effectively sub-graphs with a diameter[6] of two and multiple Meta-Techniques may be combined into a static subgraph that can be added to an attack graph for adversary emulation and simulation. Using Meta-Techniques reduces flexibility to arbitrarily combine Technique and fulfils the requirement to build graphs that follow real-life examples (see also Section 3).

Use of Meta-Techniques can be distinguished into two case: one case, where Techniques have to be fulfilled to allow for a Technique to be performed (*conjunction*); and a second case, where one Technique always branches into two or more following Techniques (*disjunction*).

The first cases, *Conjunction*, is displayed in Figure 4.4 and can also be expressed as $T_1 \wedge T_2 \rightarrow T_3$. One example taken from Stuxnet for conjunction could be an adversary profile with a supply chain compromise and usb-media proliferation, before command line interface activities can begin.

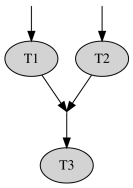


Figure 4.4: Conjunction of Techniques

The second case where Meta-Techniques are used, is a *disjunction* of a path after a Technique. If such a Meta-Technique is instantiated during graph generation, all defined successors have to be included.

Aforementioned *Disjunction* is displayed in Figure 4.5 and can also be expressed as $T_1 \rightarrow T_2 \wedge T_3$. One example for disjunction could be an adversary profile with persistence using two different Techniques (e.g. scheduled tasks and a backdoor user), once valid credentials have been obtained.

By chaining Meta-Techniques containing conjunctions and disjunctions, it is possible to model complex attack paths. If one wanted to model the following chain $(T_1 \wedge T_2) \vee (T_3 \wedge T_4) \rightarrow T_5 \vee T_6$, it can be expressed as $MTC_1 \vee MTC_2 \rightarrow MTD_1$ resp. $MTD_{MTC1 \vee MTC2} \rightarrow MTD_1$ (with MTC

[5] ATT&CK does not account for dependencies of individual techniques or sequence of events beyond the Tactic level.

[6] The longest shortest path in a graph between two nodes.

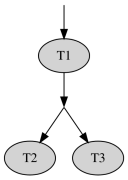


Figure 4.5: Disjunction of Techniques

and *MTD* being a conjunction/disjunction Meta-Technique). Such constructs are discouraged though for adversary emulation, since complex chains of Techniques are increasingly difficult during debriefing. Additionally, the visualization suffers with a high number of edges and Meta-Techniques. Using many Meta-Techniques in the knowledge base creates potentially duplicate activities and a high data overhead without providing much benefit for the attack graph creation purpose.

This section on graph semantics has shown that existing attack graph concepts can be extended to suit the needs of this thesis. It is recommended to use conjunctions and disjunctions only when strictly necessary, since they limit flexibility of attack graphs. Generated attack graphs should represent playbooks that are not identical copies of already existing adversaries. In the next Section 4.2 the design of the knowledge base to achieves this goal is discussed.

4.2 Database Design

A key component of the attack graph generator is a database. The system architecture is also depicted in Figure 4.7. The strongly simplified database is placed in the upper right corner. A database for the application of this thesis does not only contain resulting graphs, but also provides the knowledge base for generation and required dependencies for attack graphs that represent possible real-world attack chains.

As with the prototype itself, the database is also designed to support an expert system. Entries in the database must have enough context to be usable by experts. This means references to sources need to be included, but basic concepts do not need to be explained. Since the initial dataset for the prototype is derived from MITRE ATT&CK for ICS, not only naming schemata are replicated, but also the level of detail from aforementioned framework.

The database backend can be reduced to an SQLite file, though many other databases are possible. This decision eliminates the need for a full database, especially in smaller or development environments. It is often infeasible to setup systems solely for training or exercise purposes and thus an attacker-/trainer-team might need to bring their own systems to an exercise or rely on previously generated graphs and playbooks, where networks might not be available. Sharing data or complete knowledge bases is easily realized with SQLite files, when compared to conventional databases. The prototype can utilize arbitrary databases that are supported by the Django framework and does not result in limitations.

Figure 4.6 shows the high-level database design in an Entity-Relationship (ER) notation. Similarities to MITRE ATT&CK as well as STIX are intended for the knowledge base and allow for an easy knowledge transfer (see also Section 5). Entities solely supporting administrative functions are not included in the diagram since they are not relevant for the design itself.

Figure 4.6 can be divided into two parts indicated by color coding: one part is attack graphs and their nodes as instantiated data and another part is classes based on MITRE terminology. The knowledge base of the graph generator is independent from instantiated data. Instantiated

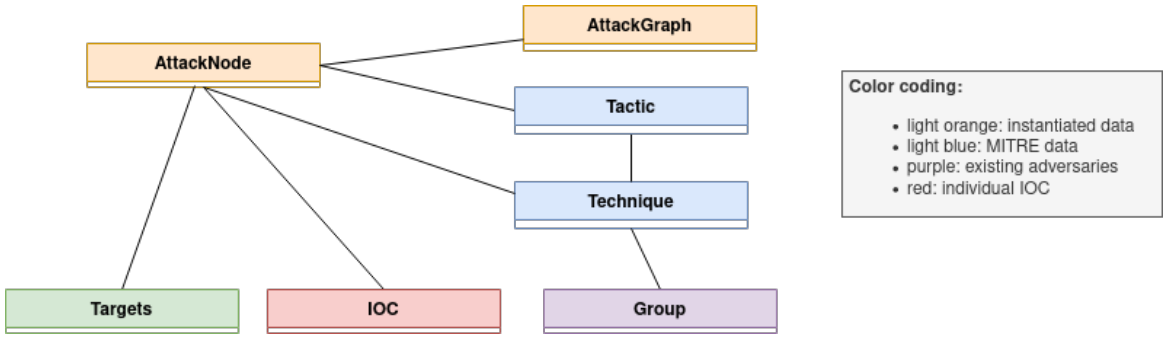


Figure 4.6: High-level database design

nodes are linked to knowledge base data (e.g. MITRE ATT&CK) and thus can be augmented with additional information. This high-level structure is depicted in Figure 4.6 and detailed in Chapter 5.

A semi-formal ontology for cyberattacks can be derived from MITRE frameworks (see Section 2.1.5) and the general idea of attack graphs is described in Section 2.3. Since those frameworks and the nomenclature have itself established as a de-facto standard[7] in the information security domain, the database design will use those concepts as well. A formal ontology like STIX may be mapped as well to the MITRE framework's naming scheme [13].

For the knowledge base existing TTP, derived from sources like ATT&CK or similar repositories, can be mapped to aforementioned database structure. Users can follow familiar naming schemes and relationships between elements, when using the attack graph generator. Figure 4.6 shows light orange elements representing instantiated data (AttackGraph entity and corresponding nodes) and other colors indicating the knowledge base part.

Instantiated data (attack steps derived from the knowledge base) is organized within the same database as the knowledge base. This eliminates a necessity to link multiple databases. Attack graphs consist out of multiple attack nodes that are instantiated techniques. Each node represents a step to be performed during adversary emulation. Linking multiple attack nodes and creating a sequence results in an attack graphs that can be considered a playbook (see Section 2.2) or adversary profile (see Section 2.1).

Requirements as stated in Chapter 3 and the fact that new incidents are regularly analyzed lead to a knowledge base that is not static. In general, a knowledge base of an expert system should be growing over time. This means once new IOCs are identified, new techniques are established and more APT groups are discovered, new entries are added to the database. If those are merged with existing data, possibilities for attack graphs become more diverse and the number of possibilities to permute instantiated techniques increase.

The level of detail for entries in the knowledge base depends on the organization, where the attack graph generator will be used. For a prototype, as it is depicted in Chapter 5, a generic set of techniques and targets suffices and fulfils the general architecture covered in ISO/IEC 62443 (see Section 2.4.1). Such a dataset is provided by MITRE ATT&CK for ICS. That data needs to be augmented by dependencies and graph generator specific attributes though. When using ATT&CK as a dataset, appropriate attack graphs should be possible to generate. Though someone using the graph generator would have to interpret results to make them fit for specific purposes.

With a database design, like it is depicted in this section, constraints for graph generation,

[7]Many security products and advisories already at least reference ATT&CK.

TTP for behavior description and naming conventions are set for a prototype implementation. Each entity, their fields and implementation are detailed further in Chapter 5.

4.3 Application Design

General design decisions affect many functional and non-functional requirement aspects. In the following sections the application flow and its characteristics will be covered. Additionally, the prototype's architecture, functionalities and data sources are described.

The prototype application should support an operator in planning and executing adversary behavior in exercise-like scenarios (see Chapter 3). For this purpose the exercise lifecycle depicted in Figure 2.7 needs to be mapped to the program flow of the prototype. To allow a seamless usability, all non-administrative operations are performed in one-application.

The attack graph generator prototype is realized in Python3. In the following all references are related to the currently supported version. Python is also used in many hacker tools and exploits, so a transfer of data can be easily facilitated. Same is true for the defenders viewpoint - many threat hunting tools[8] (e.g. the jupyter Threathunter Playbook) and information exchange about adversaries between different organizations rely on Python. The pyMISP library is a good example on how Python is used as an API for information sharing in a threat intelligence context. Besides use in the information security context, many machine learning frameworks with neural networks (ANN) today (e.g. PyTorch or tensorflow) are also based on Python.

By using Python, interoperability from outside the application is achieved and users may extend functionality easily. Aside from Python potentially other candidates for the implementation could have been golang (popular in the hacking community) or PHP (popular in web development), since they have similar attributes and a relatively high popularity among the information security community. Those alternatives do not fulfil requirements to the same extent as Python, which is why they have finally not been chosen for an implementation.

The prototype is realized as a web application with a database backend (see Section 4.3.1). This allows system-independent access and flexibility, when it comes to the development. As already stated in Section 3.2.1, a web application can usually be run independent from a client as long as a browser is available. The server component is capable of running locally on a system.

Developing a web application can be very time consuming and error-prone. To speed up development and reliability, different frameworks are available. One commonly used Python web development framework is Django allowing for rapid prototyping. This and the fact that Django provides functions per default that cover already functional requirements (e.g. authorization, database connectivity and such). This makes it suitable for the implementation aspect of this thesis. The builtin features include a system-level Python API, database back-ends, authorization/authentication and many other builtin functionalities that are required for an attack graph generator (See the Django documentation[9] for details.).

From a high-level design perspective two main libraries are used in the prototype. Reimplementing graphical visual representation of graphs or efficient data structures with graph operations can and should be avoided. Utilizing standard libraries supports extensibility and interoperability and reduces errors.

The first notable software component and corresponding Python library used in the prototype

[8]The search for malicious behaviour in existing data.

[9]https://docs.djangoproject.com/en/3.1/

is Graphviz. Graphviz allows to automatically draw and layout graphs with an own algorithm and is available for a range of architectures/systems. Graphviz has been researched extensively and algorithms are already optimized. This and the feature set make it a suitable integration into the prototype. Output formats can be any common image type and thus allows for flexibility. Layout and formatting functions can be used to fulfil the visual aspects from previous chapters regarding graph design (see Section 4.1). Other graphing solutions allowing for direct manipulation of nodes may be added in the future. The latter is not supported directly with Graphviz.

The second notable software library used in the prototype is NetworkX. Instead of reimplementing graph algorithms and data structures a proven and tested library is employed. NetworkX supports a wide range of input formats. NetworkX graphs can also be stored as serialized data or exported as JSON data structures. This facilitates import/export functions and interoperability with other tools, e.g. machine learning frameworks. NetworkX graphs are used as an intermediary format before graphs are visualized with Graphviz. Many features provided by the library are not yet used in this thesis' prototype. In the future shortest path calculations and similar calculations might be incorporated into the graph generator.

Considering the phases of an exercise or simulation (see Figure 2.7), the prototype has to guide a user towards generation of a suitable graph that is used on a corresponding use case. An operator would start at a front page with an overview about ongoing playbooks and operations. From this front page an interaction with existing playbooks (detailed view) or creation of new playbooks has to be available. The workflow to create a new graphs requires a user to enter or choose a name and add description as well as constraints in the web application. After a graph has been generated, an automatic forwarding to a details page is performed. To fulfil the requirement for documentation and visualization of attack steps, a detail page of each graph has to be implemented for associated graphs and nodes. An operator needs to be able to add results and status of each node. For each node a description about attack step has to be available. Additionally, to allow for knowledge transfer in a Purple Team scenario, mitigation and detection measures show be accessible. If parameters of a graph change or results are added, a graph has to be rendered again to visualize changes.

The application design pattern follows the idea of separating visualization from business logic and database with a knowledge base. This design pattern is commonly known as model-view-controller [50] and separates logic from presentation and database operation. With the Django framework used during implementation in Chapter 5, the model-view-controller pattern is slightly changed into a model-template-view paradigm [28]. This does not impact the design perspective, since those concepts are very similar.

In a model-template-view approach a model contains database classes and database definition. Templates are basis for webpages being shown to a user. Views in Django contain logic for views. The business logic can be placed in views or implemented standalone. In the case of the attack graph generator, logic for graph generation is a function in a dedicated backend module. Any logic directly linked to displaying data and website content is implemented in a view-module. By following this design pattern, complexity during development is reduced (each aspect is clearly segregated with well defined interfaces), different components (e.g. database or backend) can be replaced without modifying other components and functionality can be added more easily [50]. Such design pattern support extensibility and code readability.

Performance, usability and error-tolerance are implicit results from using existing and established frameworks like Django, Graphviz and NetworkX. For the prototype implementation of this thesis no additional performance optimizations have been performed. Usability is primarily achieved by an intuitive flow of interfaces and adequate guidance for data entry. Security

controls (e.g. authentication) is provided by built-in Django functionality. For the prototype, access control is limited to registered users. A fine-grained rights-managed is beyond the initial demonstrator.

The overall application design, including design pattern and individual components, is depicted in the following section. Figure 4.7 visualizes, how database, backend and user interaction are separated into individual components. Graphs are generated and visualized by the backend component and stored in the central database.

4.3.1 System Design and Architecture

An appropriate system design covers multiple requirements. It directly addresses functional requirements, like an API. Other aspects, like openness, reusability and security requirements are also taken into consideration. Only if the chosen architecture is open and has well-defined interfaces, other applications can utilize and interact with the tool.

The system design with each system component is displayed in Figure 4.7. It consists of multiple elements that fulfil a certain function in the application. The system design consists of the following components to allow use cases identified in Chapter 3. The database and graph related design decisions have been covered in the previous sections.

1. Knowledge Base / Database
2. Backend with Graph Generation and Graph Rendering
3. User Interface / Administrative Interface / Frontend
4. API / Export

Aforementioned components are depicted in Figure 4.7. Even though two users have been included in this diagram, it is very likely that administrator and operator are identical, at least for the prototype phase. Red boxes represent use cases already identified in Section 3.1.

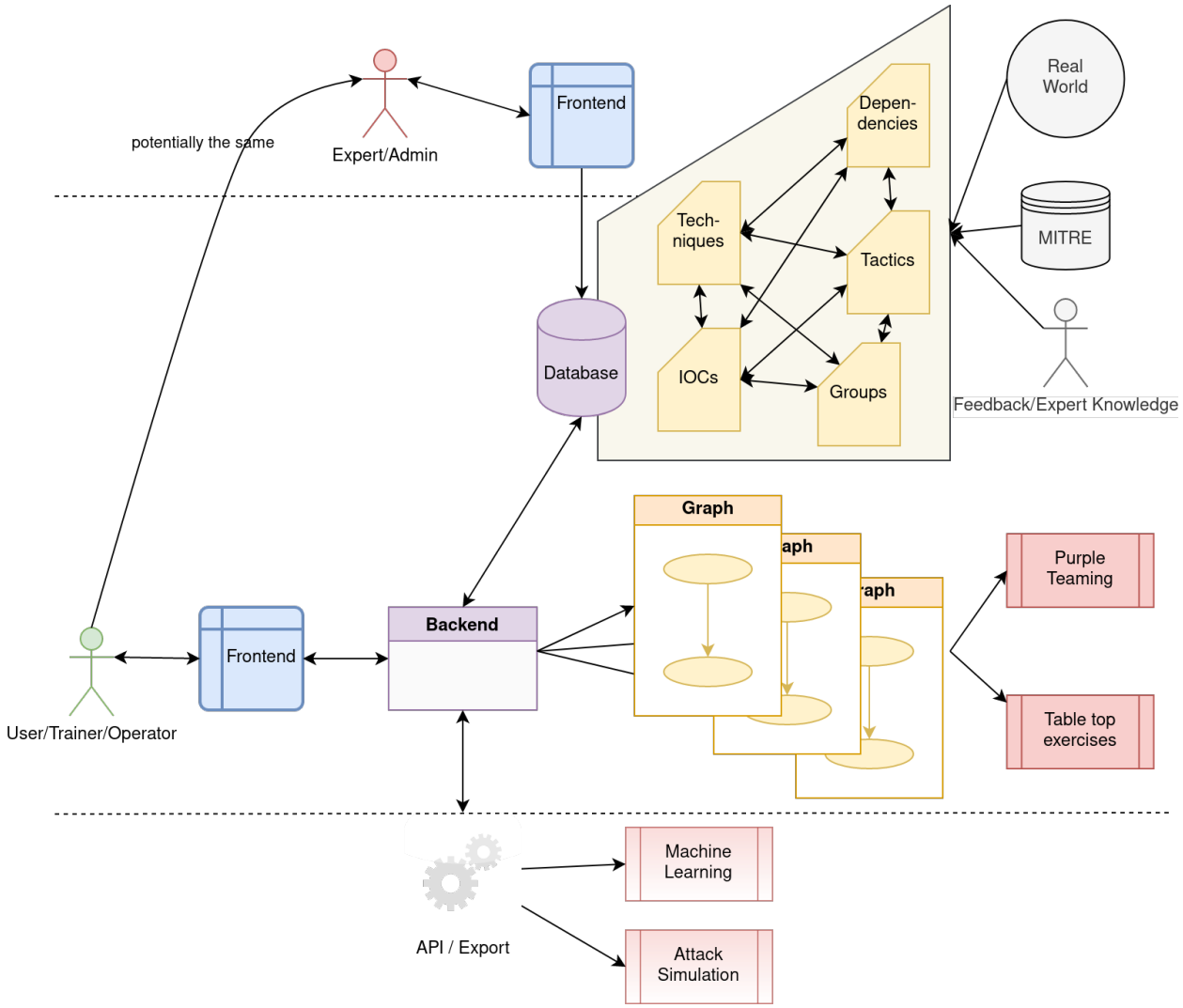


Figure 4.7: System Overview Diagram

4.3.2 Frontend & User Interface / Administrative Interface

Figure 4.7 depicts two primary means of interaction with the graph generator. In this section frontend design decisions are covered; the next section covers the API component. Interaction from users of the web application can be divided into a regular user interface, of which also parts are accessible for a non-authenticated user, and an administrative interface. Since a majority of interaction will occur in the regular user interface, the focus is placed is on an operator generating and interacting with graphs and less on maintenance of the knowledge base.

The regular user interface is designed to support attack graph generation, the attack emulation process and after-action reporting. Each component of the web application follows the same layout and has headers and footers allowing for quick navigation to each function and providing a common look-and-feel. The Graph Generation workflow directly leads to a detailed view that shows a generated graph for interaction. Details, e.g. mitigation and detection steps, are only shown on demand. This retains readability of graphs and listings of instantiated Techniques and nodes. Where appropriate, additional user interface functionality needs to be implemented to improve usability. As one mechanism, graphs can be zoomed and panned in a detail view, so even complex renderings are still readable. Functionalities not accessible without login are hidden from users accordingly.

To allow an independence from devices (e.g. high-res screens, mobile devices, etc.) different layouts and interfaces are required. To achieve this, the Bootstrap framework is used for frontend-layout and -design. This CSS framework provides a responsive design that adapts layouts to different screen sizes/orientations. Bootstrap also includes JavaScript libraries improving data input and output rendering. All layout and forms are implemented with Bootstrap, such that a modern and consistent look-and-feel is present. Implementation for frontend components is primarily realized with built-in Django and Bootstrap functions and creates a unified Graphical User Interface (GUI).

The admin interface is realized using standard Django administration components that are part of the Django framework. Viewing and editing functions with documentation for each class are automatically generated from the database model specified during implementation. To facilitate an easier knowledge base maintenance, default sorting and filtering has been changed to better represent data entry. One example is the limitation of successor and predecessor nodes to nodes of the same graph. References to other graph's nodes may not occur, so they do not need to be displayed when maintaining node data entries. Another example is an automatic sorting of techniques according to their tactic. Those optimizations to the standard administrative interface allow for easier modifications of knowledge base data and instantiated graphs.

Since user groups for the attack graph generator are information security professionals and white hat hackers, the overall user interaction does not need and does not cover many help texts and explanations. Where possible, help and validation texts are considered, when an action fails or an invalid entry is given.

4.3.3 API / Import and Export

Graphs and adversary emulations must not be considered as isolated elements, but as something that allows for interaction by experts and integration with other domains (e.g. machine learning). As depicted in Section 2.2 and in previously identified use cases (see Section 3.1) added value happens, when information is shared among participants of exercises and other programs and use cases can access and manipulate generated attack graphs. To facilitate this, the application provides an API and import and export functionality. Both options are explained

in the next paragraphs.

A user can interact with the application in an API-like fashion either via the web-frontend's import and export function with user or administration level access or directly on a Python API. As a third, but discouraged option is direct manipulation of the database. Directly interacting with the database poses a risk to introduce inconsistencies, since all application-level checks are bypassed by design.

An API is part of the Django framework and allows, once the application's modules have been imported, to instantiate objects, use methods and make use of all features the web application itself uses. If one wanted to integrate another program, e.g. for a simulation or a machine learning use case, it either needs to natively have a Python dialect or a wrapper/library has to be created. A dedicated API does not need to be implemented, but may be part of future work.

For bulk import and export to and from the database a table-level feature is available in the administrative section of the web application. This allows data download of Excel and JSON (and many more formats) as well as an import of such files. Upon importing duplicates and modified fields are detected and merged with existing data. This does not only provide a possibility for portability of data, but also to change data text-based or in a spreadsheet and afterwards importing it again. If larger amounts of data should be added to the knowledge base, it can either be done via an API or via a bulk-import from structured data (e.g. CSV).

Attack graphs can be exported in multiple forms. In the web interface graphs are shown as vector graphics (SVG). Since SVG is not a common format as e.g. PNG, each graph can also be downloaded as a PNG file. Other formats for rendered graphs can be integrated by using functionalities of the Graphviz library. Displayed rendered graphs in the web application will be vector graphics. This allows for scaling without quality-loss.

When graphs are required in a machine-readable and not in a visual representation, the application allows exporting data structures of graphs. The NetworkX library has built-in serialization and de-serialization functions. By using the right parameters in requests, graphs are exported as JSON (e.g. node-link-format) or as a pickle-byte-stream[10]. Aforementioned serialized data can also be used in a simulation or machine learning context.

Importing graphs from aforementioned data structures may be achieved with the Python API (e.g. after receiving results from a simulation), but is not implemented in the web interface for a regular user. The use cases featured in this thesis do not require such a feature; they might be implemented as part of future work though.

4.3.4 Data Sources

This section covers how data source fulfil requirements from Section 3. This allows to address constraints on generation, to establish uniform naming conventions, to allow for extensibility and reusability as well as a defined import and export functions.

An overview of input and output data used for graph generation for different use cases as depicted in Section 3.1 is available in Figure 4.8. Data consumed or produced by the application can be divided into three categories:

1. input before any graphs are generated,
2. input during execution of adversary behavior and
3. output of the application.

[10]The pickle Python module is used commonly to create a byte-stream from an object and vice-versa.

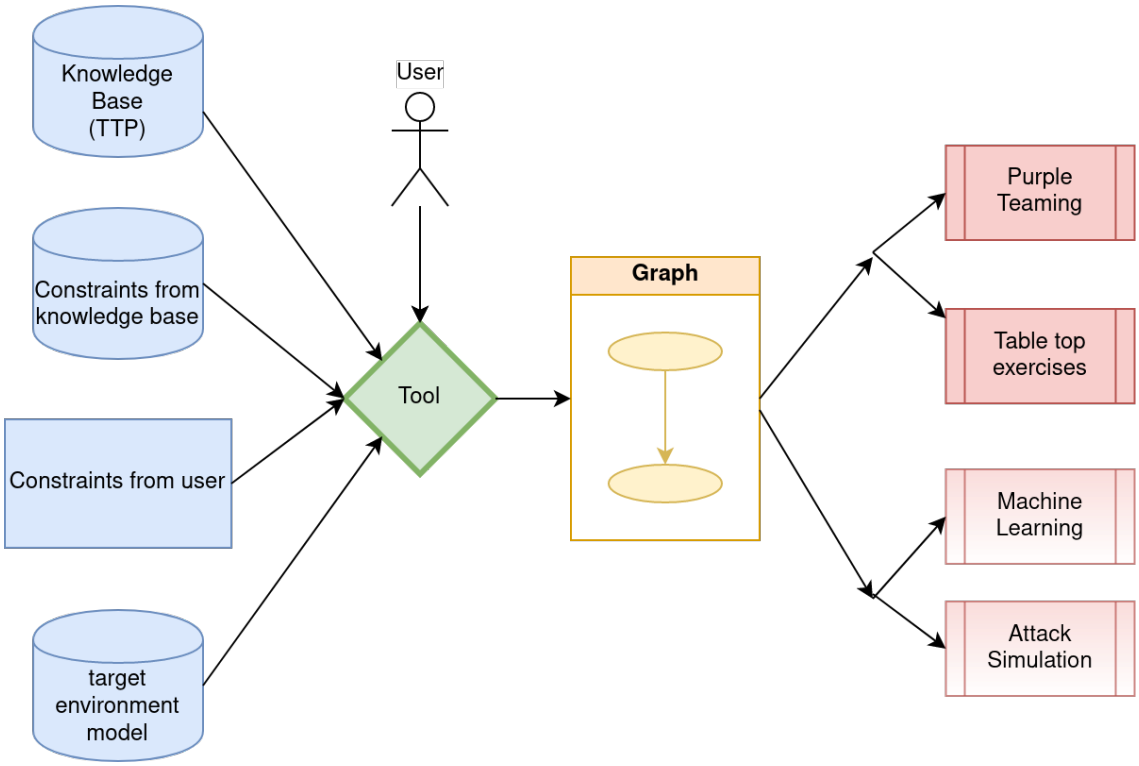


Figure 4.8: High-level overview of input parameters and output/use cases

Input to Graph Generation

The first data source category covers inputs before an attack graph is generated. In the following input data depicted in Figure 4.8 are described.

Each graph of this thesis represents one or many adversary behavior execution playbooks. Since each graph is unique, a user should specify a name for that particular, artificial adversary and a description. This allows to distinguish different graphs. The application has a name generator that creates unique names resembling common APT naming schemes, in case a user does not want to specify an own name.

As one of the key elements providing data for the knowledge base MITRE ICS ATT&CK framework is used. MITRE ATT&CK data has been peer reviewed and already contains a significant amount of data required for implementation of the knowledge base (see Section 2.1.5). If updates or additions to datasets occur, updates can be reflected in the database of the attack graph generator. Other sources of data include expert knowledge, e.g. from results of Purple Teaming, or reports of attacks not yet incorporated in the database. Since Enterprise ATT&CK already has a machine readable API, it is likely that the ICS framework will also be provided that way in the near future. This allows for automatic sync and updates of external data.

In addition to data from a knowledge base, a user has to have a possibility to enter constraints that do not result from existing data. During graph generation a user can select or remove attributes (e.g. certain Techniques) to limit available graph options.

After graph generation, manual modifications might be necessary or desired to have an appropriate playbook. A user can then utilize the application's backend or the database directly to change nodes, attributes and parameters. Even though modifications of a graph occur after the generation algorithm, they still can be considered input data.

If the modeled environment in the knowledge base is too generic or does not fit a particular

use case, it might be necessary to modify or specify a target environment model. This can be achieved in the database backend.

After a playbook or attack graph has been created, a user interacts with the tool and executes Techniques from that playbook. During that phase, further input is given into the application that is covered in the next section.

Input to Adversary Behavior Execution

During adversary behavior execution an attack graph and its corresponding activities impact how an emulation or a simulation will be executed. Those generated graphs can also be seen as the first/intermediary output of the application.

Primary input to the application after a graph has been generated, is documentation of individual attack steps. A user performs adversary behavior execution or has it performed. Ultimately, those activities should be used to improve security posture or deduct a result from emulation or simulation.

Documentation may contain individual steps or even low level descriptions of activities performed by ethical hackers or a computer program. Ideally that documentation contains information for reproducing and tracing TTP and IOC in a target environment.

To be able to find valid and unmitigated attack paths in a graph and to identify room for improvement, results of each attack step (e.g. successful, partially successful, etc.) need to be entered into the system.

Graph as well as results and documentation are finally combined into an output described in the following category, once an adversary behavior execution is finished. Output of the attack graph generator is covered in the next section.

Output of Attack Graph Generator Application

Just as the extent and detail level of input vary based on a use case, the application's output also depends on whether an adversary emulation, simulation and/or automation is employed. If e.g. machine learning is used, a full debriefing with documentation of each step is not required, whereas a table-top exercise will require detailed write-ups to allow lessons learned.

As a first output (not only for performing behavior execution) each artificial adversary/iteration of graph generation, creates a playbook for execution. Each graph represents one playbook with a set of TTP to be emulated. A user or program interacts with graphs during execution and modifies the output accordingly. The current graph (including result and progress indicators) is updated when changes occur. Graph visualizations act as a high-level overview and playbooks with step-by-step attack information provide details. For details on graphs and their semantics, see Section 4.1.4.

For each Technique the application provides context and information like detection and mitigation measures. This information can be accessed by a user and is referenced from the knowledge base (see Section 5.2).

A user might export single playbooks or graphs to be used in debriefing, other post-action activities or even other applications. During debriefing of an exercise results are discussed between attackers and a defending team, to jointly identify what went well and where room for improvement has surfaced.

Chapter 5

Attack Graph Generator

This chapter focuses on implementation aspects of the prototype of this thesis. The implemented prototype will focus primarily on exercise use cases, but the application will also provide necessary features for simulation where slight adaptations are necessary. The implemented prototype will generate graphs and has an API allowing interaction with the database and graphs. A simulation tool needs to read data with an API or directly from the database and use that data for automated execution of adversary behavior.

5.1 Backend

Two main backend functions (as depicted in Figure 4.7) are Graph Generation and Graph Rendering. Both are explained in detail in following sections. Other implementation aspects are available in the source code of the prototype of the graph generator. Describing those aspects in detail is beyond the scope of this thesis though. Since those details are not specifically related to the research question from Section 1.1, they can be omitted in this section without impacting the ability to provide answers.

5.1.1 Generation Algorithm

This graph generation algorithm follows a sequential approach. A high-level overview in an event process chain format is depicted in Figure 5.1. The graph generation function tries to fulfill constraints provided by a user when requesting a new attack graph. If constraints are too limiting and cannot be fulfilled with data from knowledge base, graph generation fails and a user has to either add new techniques to the database or loosen constraints accordingly. Additional to user constraints, the knowledge base itself also contains constraints for graph generation. These constraints either simply limit predecessors and successors of a Technique or represent more complex dependencies by using Meta-Techniques as described earlier. All knowledge base constraints are explicitly specified by an expert providing input or by analyzing reports about attacks from the past. It is not possible to override these constraints during graph generation, since that would result in inconsistencies. If such a manipulation should be required, it has to be performed directly in the database or via the administrative backend.

As depicted in Figure 5.1, initially constraints are evaluated and available Techniques from the knowledge base reduced. From that set of Techniques a number of nodes are generated. This is done for each Tactic within scope of generation. Once all Tactics are completed, generation is finished, the resulting graph is saved to the database.

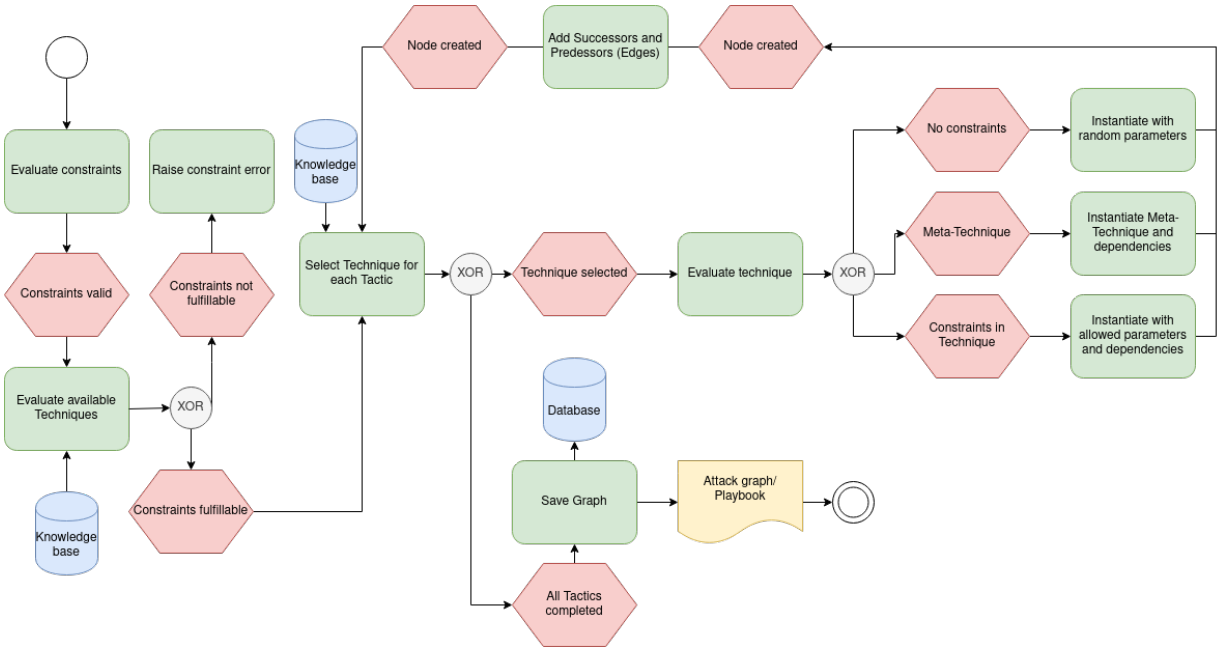


Figure 5.1: Event process chain for graph generation workflow

The first step of user constraints impacts further selection of Techniques and Targets. At this point, knowledge base constraints are not considered yet, since they are evaluated upon instantiation. If user constraints limit the available dataset too much, graph generation will fail, since not enough Techniques/Targets are available to build a consistent graph.

After a node has been instantiated, edges are generated between new nodes and already existing nodes. Though it is possible for an adversary to explore every single attack path, it is very unlikely. Analyzed attacks from the past, show that real adversaries do not exploit every vector on their way to the final target. A fully meshed graph is thus infeasible. The limited set of edges that attack graphs of real adversaries have needs to be represented in the generated attack graphs as well. This is achieved by not generating every single possible edge, but stopping edge generation after a number of transitions. The edges are saved as predecessors/successors in instantiated Techniques. Conjunctions/disjunctions can be realized by the same concept using Meta-Techniques and their predecessors/successors.

As stated, generation of edges is not performed exhaustive. Otherwise every potential transition would occur. If edges were generated completely random, smaller graphs might not have enough connectivity. To prevent this behavior, a probability function based on power-law distribution is included in the graph generation function: no edges are least likely, followed by maximum number of edges and a single edge is most likely. During testing, this provided the most reliable results and also represents adversarial behavior: an attacker usually does not exhaust his options during an intrusion.

If Techniques are associated with constraints, e.g. due to restrictions on targets or required predecessors, this has to be considered during instantiation. For the generation algorithms, three cases can be distinguished: no constraints (random choice of properties), constraints in Technique (instantiation based on constraints) and Meta-Techniques. Those Meta-Techniques and their properties are described in detail in Section 4.1.4.

As described previously, recursion and loops are not allowed and thus this algorithm does not need to account for those cases. If such a feature should be necessary, aforementioned graph generation algorithm needs to be changed and the knowledge base modified to be able to represent such data structure and constraints.

Generally the graph generation approach follows a top-down approach, where a graph is being built from the first step of an intrusion towards action on objectives. Other graph generation algorithms (especially those trying to model critical path for defenses) utilize a bottom-up approach. Since adversary emulation follows sequentially through different phases on an attack, it is suitable to maintain a chronological order also for graph generation. In theory, a bottom-up approach is also possible and allows to start with a certain goal that an attacker wants to achieve and backtracks from there. Backtracking has the disadvantage that Techniques without connection to the final goal might never be instantiated, even if they could be part of the adversary profile. Another disadvantage of a backtracking method is a potentially very broad "top-level" (earlier tactics), when all dependencies have to be modeled. This in turn results in a bias for higher amounts of early techniques (e.g. initial access).

The graph generation function generally iterates over each tactic (as a phase of an attack) and instantiates a number of attack techniques for that tactic. An instantiation of a Technique contains a selection of a Technique itself, association with a target and association of an IOC that should be used. Possible targets and applicable IOC are extracted from the knowledge base and selected randomly for each instantiated Technique. This ensures that a Technique may occur more than once (with different targets) and that sufficiently random attributes occur. Each node is saved after all relevant attributes have been chosen (see also Section 4.2 for the database and knowledge base design).

The number of instantiated nodes per tactic can be controlled by a user. By limiting the number of nodes per tactic, a graph can remain manageable and, just like a regular adversary would not perform all possible attacks, not each available attack technique needs to be emulated or simulated. This restriction is realized by requiring and processing such a parameter during the generation phase.

Available Techniques per Tactic also include Meta-Techniques (see Section 4.1.4) associated at that Tactic level. When instantiating a Meta-Technique that are generally fixed regarding their edges and nodes, missing predecessors and successors are automatically instantiated as well. This can result in more nodes per Tactic than specified in constraints by a user. If such a behavior is not desired, a user can either exclude Meta-Techniques or manually modify a graph after generation.

In theory, graphs generated by this algorithms have to be syntactically correct and fulfil semantic correctness as defined in the Design chapter 4. Constraint data to achieve semantic correctness can be derived from the knowledge base. Interaction with a graph, e.g. rendering or exporting, is performed via the NetworkX library that also validates graphs when they are built and processed. In case a non-valid graph that violates graph constraints is encountered, e.g. because instantiation failed, NetworkX will raise an exception.

Despite a possibility to edit graphs via the administrative interface, it is not possible to add constraints to graphs afterwards. Removing constraints and thus adding more potential combinations after generating a graph is not practical and cannot be supported. The same applies to adding additional constraints. Though adding new constraints and effectively reducing the number of combinations is technically possible after graph generation, a user has to generate a completely new graph with new constraints or manually manipulate a graph accordingly. In practice, this limitation should not be relevant for identified use cases from Section 3.1

5.1.2 Graph Rendering

Graphs are not only a means to structure information and relationships. They are also capable of visualizing those structures. The graph rendering function transforms individual entries for nodes and corresponding edges from the database into a graphic representation. This allows users to quickly identify different building blocks of an attack graph as well as transitions and sequence. Since graphs will be used during planning and documentation of adversary behavior execution, an easily comprehensible image supports those processes, especially if results are rendered dynamically into existing graphs.

As depicted in Section 5.1 (Graph Generation), information about graphs is stored as a double-linked list. That list needs to be transformed into a representation that can be used for drawing an image. To allow rendering operations on attack graphs, all nodes and edges of a graph are extracted from the database and added to a NetworkX graph (see Section 4.3). This graph uses adjacency lists and highly optimized dictionary data structures when handling graphs. That data structure can subsequently used for exporting and rendering an image. To build a graph, it needs to be explored and each element added to the visual representation. Once a NetworkX graph has been fully built, data can be passed on to the drawing engine.

Figure 5.2 shows a high-level overview of the rendering function used in the prototype. A graph is either rendered, when it is displayed as a result of the generation function (see Section 5.1) or when properties of nodes are changed. This happens, when results are entered or status information is modified by a user or program. Graphs are not stored persistently within the application, but always generated on-the-fly and served as part of the web application. For persistent storage by a user, each graph has a download option.

As stated before, for normal web application operations no image is written to disk, but all graphs are rendered in-memory and directly provided as part of an http response. Using that mechanism, no cleanup of residual files is required. Since the format of edges and nodes changes during an adversary emulation process, static images would need to be updated regularly. This unnecessary disk access can be avoided by serving directly from memory. A disadvantage of this approach, if no caching is performed, consists in additional rendering load, when a graph is requested.

Attack steps are sequential and therefore attack graphs also need to follow that sequential layout. In this thesis, a top-down layout, with start of an intrusion on the top of graphs, has been chosen.

As a first step, all nodes associated with a graph are extracted from the database. Those nodes contain all required references in the database to render a graph. Those nodes are added to a NetworkX directed graph and linked with each other according to entries regarding predecessor and successor of each node. That graph is stored in memory and each node subsequently added to the data structure. Once all nodes have been added, meta-information about the nodes can be added. This includes visualization of status, results and tactic color.

Once all formatting has been performed, a graph with the NetworkX data structure is converted into an image. This is done with a third-party program, Graphviz, as stated in Section 4.3. The core of graph rendering uses the Graphviz program with the PyGraphviz library for drawing graphs and layouting nodes and edges. Graphviz has the advantage over matplotlib (another standard plotting library) that it allows finer control of layout and formatting of nodes. Other solutions require file-conversions for web output (e.g. TikZ) or do not have a NetworkX programming interface, thus making interaction difficult. Graphviz and its dot-Language are capable to render attack graphs with the necessary information an operator of an adversary emulation or a simulation would require.

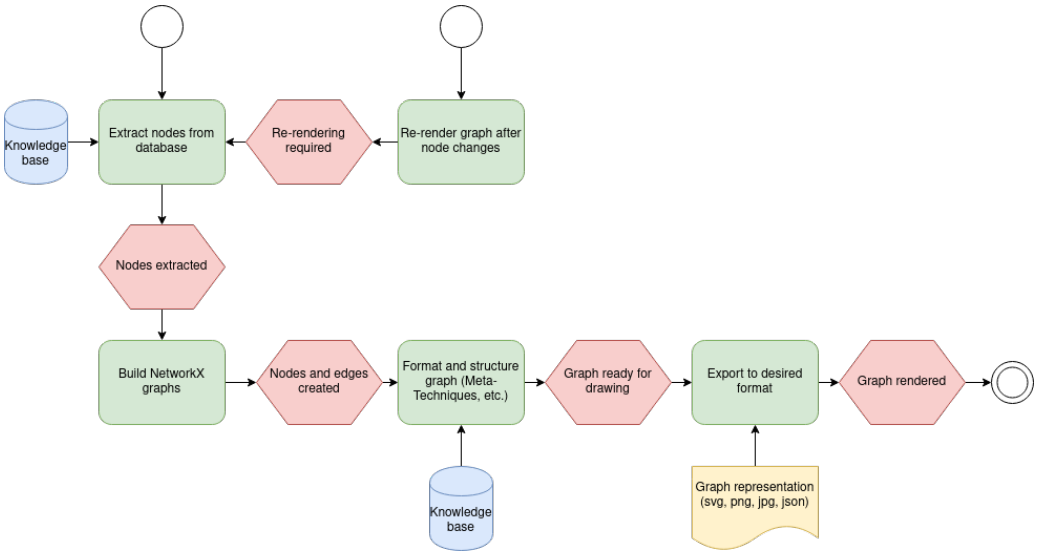


Figure 5.2: Event process chain for graph rendering workflow

To allow a visual differentiation of different status a Technique execution can have, nodes have either a solid border, a dashed border or no border. Techniques with no borders have not been emulated or simulated yet, whereas a dashed line indicates a work-in-progress state. Solid border receive Techniques that have been completed, regardless of result. The Node format is depicted in Figure 5.3.

As stated in Section 4.1, edges represent the result of a Technique's execution. Every edge that has been created during graph generation is displayed, but with a different format, even unsuccessful ones. Details on the format are available in Figure 5.4. Attack graphs represent TTP of adversaries and as such, edges are not dynamic. An adversary profile does not change, even if some Techniques fail in a target environment. Every generated graph starts out with light-grey, directed edges indicating the result of the Technique an edge goes out from. As a result, an attack step may be successful, partially successful (e.g. only in certain situations) or unsuccessful. An unsuccessful Technique does not indicate failure of an adversary emulation or simulation, since the goal is an improvement of security posture. Unsuccessful attacks show already good controls or good processes to handle an intrusion. During debriefing it is important to also show, where defenders are already countering attackers sufficiently.

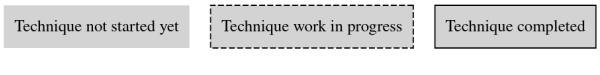


Figure 5.3: Node format

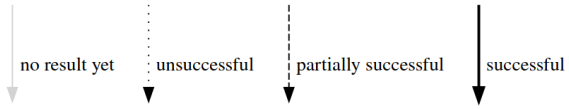


Figure 5.4: Edge format

Additional to a status indication with different borders, each Tactic has a unique color indication. Different "cluster" of Techniques can be easily identified based on this property. The color coding is from green (Initial Access) to red (Impact). ATT&CK for ICS has 11 different Tactics, thus the gradient also has 11 individual colors. These colors corresponds with an increasing "impact" of a certain Technique with regards to the assets: compromising an outer perimeter is less critical (green) than being able to manipulate control equipment (red). Colors may be customized in the prototype's backend individually for each Tactic.

Meta-Techniques are not represented as regular nodes in the graph, but as zero-width, hidden elements. In Figure 5.6 such an element is at the junction of the edges. This allows two edges to

Figure 5.5: Color gradient for Tactics

join or split for conjunction/disjunction cases. Graphviz does not provide a default solution for such a requirement, so this implementation was chosen. The result of a Meta-Technique with conjunction (join) is calculated by the worst inbound result. In the provided example, even though T0 is *successful*, T1 is only *partially successful*. This results in the hidden T2 Technique (junction point) having only an outbound edge indicating *partially successful*. These types of dependencies are evaluated during graph generation

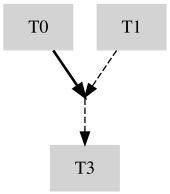


Figure 5.6: Exemplary edges with Meta-Techniques (T0 successful, T1 partially successful, thus T3 partially successful)

The graph rendering supports multiple output formats; for the web application Scalable Vector Graphics (SVG) are preferred, since they are very small and scale well, for exporting graphs to be used in reports or similar PNG files are created, as they provide a high compatibility and good size/quality tradeoff. Details to different export formats (including machine-readable text-based formats) can be found in Section 4.3.3. The prototype has functionality for JPG, PNG, SVG (default) and JSON format (where JSON is not an image format, but a machine readable graph representation in this case). Graph rendering is independent from graph generation itself. This modular approach allows changes to the output without changing the database or other core functionalities.

In certain edge cases, it is possible for a Technique to neither have an inbound or outbound edge. To have an option to prevent these single Techniques to be rendered allows the operator to maintain easily comprehensible graphs. Single nodes still exist in the NetworkX representation and in the database, they are just not rendered for visualization. This truncation is optional, but may be used to limit display only of nodes containing a minimum number of edges.

As an additional rendering option, Tactics as descriptive nodes may be added as well. This helps an unfamiliar operator to easier associate different Techniques to the right phases of an intrusion. This legend loses its advantage, if many out-of-order nodes are part of a graph. It is best used, when a sequential graph is in place and every node is correctly placed on the same level as their corresponding Tactic.

This section detailed two main functions of the prototype: generation and rendering of attack graphs. As described in the Design in Chapter 4, primary input to this application is derived from a knowledge base. Details on the database implementation are covered in the next section.

5.2 Knowledge Base and Database

This section covers implementation aspects and details regarding individual tables of the knowledge base and database in general. An overview on the design is provided in Section 4.2.

Choosing an SQL-compatible database, this prototype uses SQLite, allows to manipulate data easily also with other means than the prototype application. For an initial data set, MITRE ATT&CK for ICS data was extracted from the official repository and imported into the database. Where necessary modifications to data was performed, before it was ingested into the knowledge base.

Extending the diagram from the database design in Section 4.2, each table is implemented with a number of attributes. An entity-relationship model in full-size is available in Figure 5.7. Different relations (e.g. foreign keys, many-to-many relations, etc.) are indicated by different arrow-types linking fields.

As stated in Section 4.2 (Database Design), names of tables generally align with definitions used in MITRE ATT&CK frameworks (see Section 2.1.5). This mapping allows for easy transfer and common language in the information security community. The same applies for attributes stored with each object in ATT&CK.

Figure 5.7 shows the detailed database implementation for the graph generator in ER notation. The tables AttackGraph and AttackNode allow storing and retrieving attack graphs and instantiated Techniques themselves. The other tables represent the expert system's knowledge base as attributes, constraints and dependencies for graph generation. Each logical unit in the diagram is colored according to their purpose. AttackGraph and AttackNode are one logical unit and every other table has to be seen individually. Tables required for intermediary and administrative purposes, as well as those modeling many-to-many relationships, have not been modeled/drawn explicitly, but are either system generated and automatically populated with data. Adding them to the diagram would only introduce unnecessary complexity.

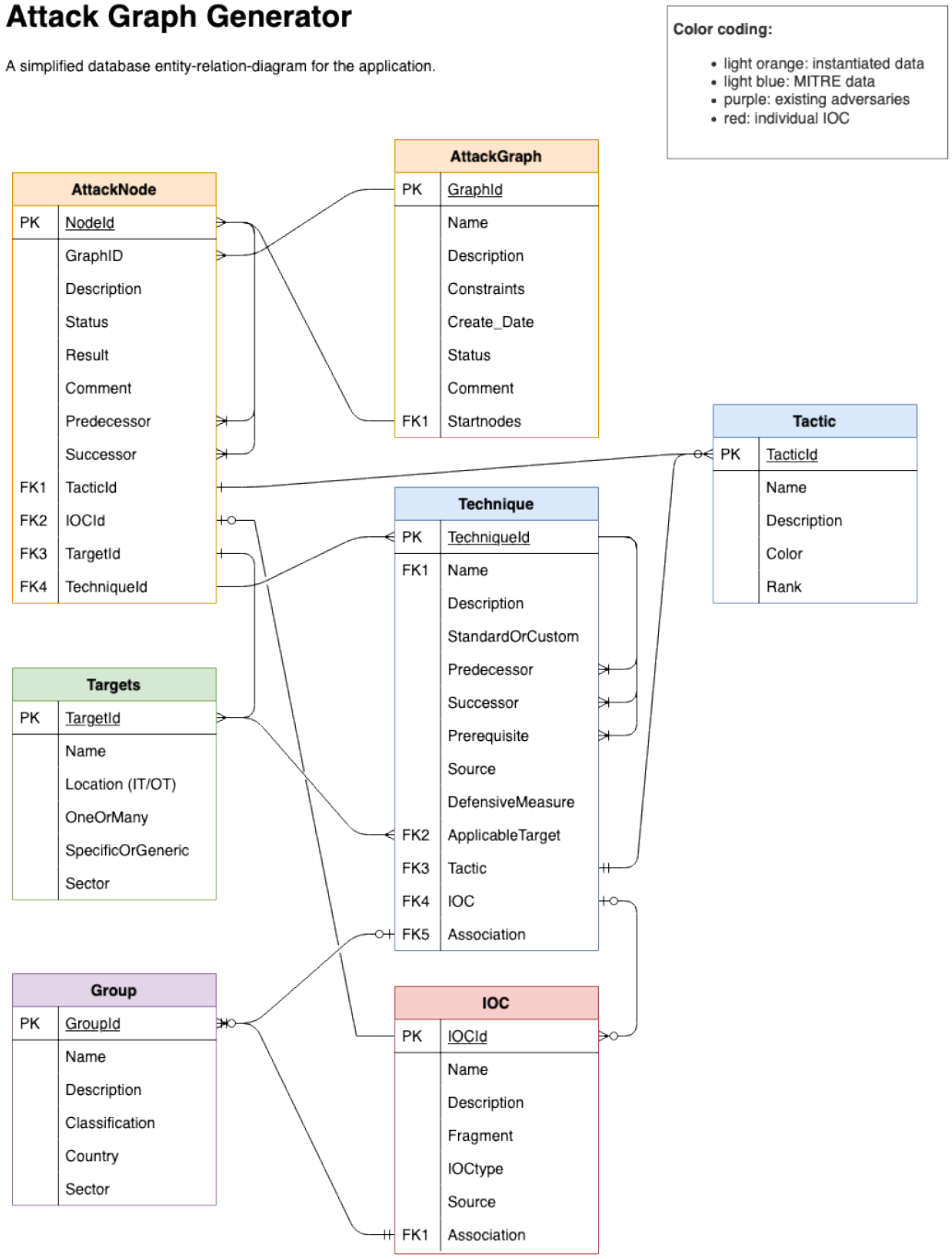


Figure 5.7: Entity-Relationship diagram

5.2.1 AttackGraph

Generated attack graphs are the central result of the application and basis for Purple Teaming or adversary behavior execution in general. Graphs themselves are a set of nodes and edges in a double linked list (see Section 4.1.2). Nodes are instantiated from the knowledge base and edges are automatically assigned based on links (predecessor/successor) between each node, even though not all possible links may be established during generation. Section 4.1 contains further information about the graph's design itself.

Fields of the AttackGraph table are mostly meta information about a graph itself. Each graph has a name, a description and a date. This meta data allows a user to filter different graphs and add information about graphs that can be used during adversary emulation. By default names are randomly generated imitating names of APT groups. This assists the idea of "building your own APT" for the purpose of adversary behavior execution. As part of the prototype, a name generator is available to randomly create adversary names that resemble real world adversaries.

Constraints are also stored for documentation purposes. These constraints are affecting graph creation as properties. If no constraints are given, the attack graph is constructed randomly respecting the dependencies and logics from the knowledge base (e.g. sequences of techniques, restrictions to certain assets, etc.). When using constraints the knowledge base may be restricted beyond the existing logics. This can happen e.g. when user might want to skip a certain phase, exclude certain assets or limit the amount of techniques per phase.

Attack graphs usually go through a lifecycle starting with an automatically generated graph. A graph might need some manual amendments or corrections by an expert before it can be used for emulation or simulation. After a finalizing the graphs they are considered "in execution" until they are finished. The status of an attack graph is represented by a corresponding field in the table. If used for automated simulation scenarios, the status can be neglected. In those cases, graphs are likely to be generated and immediately put to use.

5.2.2 AttackNode

"AttackNodes" represent instantiated steps of an attack and thus nodes of an attack graph. Each entry is an action an adversary traversing an attack graph takes. Each node has to be unique and associated to a single attack graph. A Technique can be instantiated in a graph multiple times, but only with different attributes (target, indicators and successors/predecessors). An instantiation of a Technique with the same attributes as an already existing one, could not be distinguished and would in practice not exist.

The table definition includes details on an instantiated technique with a description for each node, by default data of the corresponding Technique is used.

Each node is linked to a single Technique from the knowledge base. Since one technique may be used in different phases (Tactics) of the MITRE ATT&CK model, each node has be be mapped to one phase. This is done with a reference to the corresponding table. Every node has a certain rank in the graph indicating where it should be placed. The phases of an attack can be considered mostly sequential with exceptions of backwards steps for certain actions not modeled in this thesis. This sequence is represented by the rank each node has as an attribute.

Each node contains information about predecessors and successors as recursive links to other nodes of the same attack graph. This construct allows traversing and constructing the graph and subgraphs without building a complete graph first. Since recursive queries on databases can be resource extensive, queries have to be chosen carefully. Before a reference to node can

be made, it must have been instantiated and saved already.

Since nodes are instantiated attack techniques, they must associated to a certain target. This target can be any component in the domain designated as a target objective for the corresponding Technique. These targets are called assets in the ATT&CK universe.

Similar to the target attribute each node has a corresponding indicator of compromise. This is a referenced to an identical table and represents indicators an attacker has to create during emulation. Based on those indicators defenders can verify their detection capabilities.

For tracking purposes during adversary emulation and Purple Teaming, individual attack nodes need to be tracked regarding their execution state. Especially if an emulation is performed in a team with different operators and over a longer period of time, a status may move from "open" over "work-in-progress" to "executed". Finished nodes can be tagged with a result and a corresponding comment. With this information a debriefing with a defender team can be facilitated more easily. An important part of Purple Teaming and adversary emulation is management and documentation and thus corresponding fields shall be used during attack graph execution. When used for simulation or machine learning, those fields may be ignored.

5.2.3 Technique

Techniques represents different types of attacks an adversary might chose to perform to achieve a goal. Each chain of attacks in the real world is unique in its details, but can be divided into smaller, more generic building blocks that can be observed across different attack chains. Those building blocks have been analyzed by MITRE in ATT&CK frameworks and are called Techniques. They allow an adversary to reach goals at a certain point within an attack chain. Since a course of events of an attack follows a pattern (see Section 2.1.4), it is possible to create a graph with sequences and dependencies of different Techniques.

The Technique table is the core of the knowledge base, since it contains rules for chaining different nodes, valid targets and dependencies. It represents a hacker's arsenal and every node of an attack graph is an instantiation of a Technique with specific targets and attack fragments.

Each Technique has a name and a description. For MITRE-based data names and descriptions can be copied from MITRE repositories. For custom Techniques a user has to enter a new name and corresponding description. A name could e.g. be "Credential dumping" and a description may contain technical background on how to bypass protection of the Windows LSASS process to retrieve credentials from memory of a target system. Descriptions should be in such level of detail that it allows for proper emulation by a white-hat hacker.

The field Custom indicates, whether a technique is part of a well-known repository, e.g. MITRE ATT&CK, or custom made by a user of the attack graph generator. This allows for exclusion of non-standard techniques from graphs, if a user choses to do so.

Techniques have a predecessor and a successor realized as a recursive many-to-many reference to the Technique table. Similar to an instantiated version of a Technique (a Node), Techniques may have multiple potential predecessors and successors. This means multiple Techniques might lead to the Technique in question and one Technique might lead to multiple new techniques. This concept realizes the double-linked list from Graph Generation (see Section 5.1) and can be easily traversed also during Graph Rendering (see Section 5.1.2).

To prevent arbitrary and unrealistic or even contradicting sequences of techniques, each Technique can be constrained using prerequisites. A prerequisite is similar to a predecessor and successor, since it is realized as a many-to-one reference to the Technique table. Certain techniques can only be executed if other techniques have been performed before. In a future work,

prerequisites might be extended to combinations of targets and techniques, but this requires extensive knowledge and modeling of a target's infrastructure. If certain attack chains generated by the application are not feasible (as per expert judgement), manual corrections can be applied to a resulting graph.

Each Technique should be mapped to source. The source may be a web-resource, a scientific paper or an incident description. For MITRE Techniques the source should point to the online resources of ATT&CK. For custom entries an author should create an appropriate reference. A missing reference does not prevent the graph generation.

The Target field references those assets, on which a Technique can be applied to. An exploitation of a web service might not be applicable to a field device without any web functionality. When a Technique is instantiated as an attack node, one of the possible targets is selected.

Each Technique is associated with one or many Tactics. As described in 2.1.5 ATT&CK Techniques are the "how" and the Tactics can be considered the "what" with regard to an attack. Tactics can be seen as mostly sequential and allow for creating a chain of events. This is done with a reference to the corresponding Tactic entry.

Association of a Technique to one or many adversary groups allows to include or exclude certain Techniques based on certain known adversary groups. Groups might be extracted from MITRE, with the benefit of a prepared mapping to techniques, Fraunhofer Malpedia or other listings tracking APT groups. If a Technique does not have a known adversary group, the field may be left blank.

As optional descriptive fields, Techniques receive a mitigation and detection entry. This allows to provide the defenders with specific recommendation, when execution of a Technique shows deficiencies. If attack graphs are used in exercises or Purple Teaming, a direct mapping of defensive actions to attack techniques is possible with this data. Due to potentially unstructured information, mitigation and detection is a free text field.

By adding timestamps for updates and creation times, an operator can identify updates to a Technique or filter for new or recently changed Techniques. Any comments and information not part of the description can be added via a free text comment field in each Technique.

5.2.4 Tactic

Tactics represent different high-level steps an attacker usually goes through to achieve a final goal. Details on MITRE concepts and how Tactics are part of an attack chain can be found in Section 2.1.5. Tactics can be considered different phases or meta-level building blocks of a cyberattack.

Each phase has a name that identifies it. Initial data is taken from MITRE frameworks (see Section 2.1.5) and additional tactics from the ICS framework and marked as such. A user of the attack graph generator may create additional tactics. If new tactics are added, they should also be mapped to corresponding techniques. The description field helps a user to understand purpose and goal of a certain Tactic.

Each Tactic and thus also each Technique may have an individual color associated to it. Colors may be used to distinguish different phases and to visually cluster similar Tactics. As depicted in Figure 5.5, Tactics close to each other have similar colors. The color of nodes is irrelevant for machine processing of the graph data and can be disregarded then.

To allow for an easier handling of attack graph rendering each Tactic has an associated rank. This indicates how close a Tactic is to the actual goal of a hacker. This assumes that a hacker

always wants to move as close to the process as possible. The higher the distance between ranks for two Tactics, the less likely are they going to be encountered at the same time from the same adversary. As an example: when an adversary can already interfere with a process, he is unlikely to use spear phishing to gain initial access.

5.2.5 IOC - Indicators of Compromise

An IOC represents one or more technical fragments that needs to be created by an attacker in an exercise and detected by defenders accordingly. In a non-exercise context, IOC are used to search for malicious activity or to attribute adversary behavior (see Section 2.1.2).

For traceability reasons and to distinguish similar IOC, a unique name should be given. This could for example be "malicious domain www.example.com" or ""eviltext"-string in binary". Even though duplicate names are possible, they should be avoided to limit negative effects or confusion during usage of the prototype. If an IOC does not have detailed information associated with it, it is up an operator how to include that data in their workflow.

For a detailed description a second field is available. This allows for detailed instructions on how to create that particular IOC and potentially further background information on the fragment itself.

The fragment field may contain raw IOC data. This field can range from simple texts (e.g. domains) to hex-values (e.g. shell code) up to complex Binary Large Object (BLOB) (e.g. binary programs). A person emulating an adversary should be capable to incorporate those fragments into their attack steps. Aside from using already known fragments for attacks it is also possible to develop new indicators that might not be known to defenders. Using abridged and unknown indicators give a good indication how mature a defense is. The more generic and more diverse indicators are, the more difficult is a proper detection.

For interoperability with other tools and existing data, a standard format for IOC is highly recommended, but specification is beyond the scope of this work. Established standards are e.g. bro/zeek, YARA and SIGMA rules, OpenIOC or MISP events. Depending on the use case for generated attack graphs, machine readable IOC are required. A human adversary might be able to transform free text or unstructured data into IOC of an emulated attack, whereas a machine might struggle to do so in simulation.

The potential association of an IOC to an existing adversary group can be represented by creating a reference to the Group table. If detection coverage for a certain known adversary and its IOC should be evaluated, this reference can be used for filtering only for associated IOC.

The field IOCtype acts as a classifier for a fragment to be able to select only a certain type of IOC. If defenders can only detect domains, but have no host based visibility, it might be feasible to use only network-based indicators. The types allow for structuring of the IOC database that extends the knowledge base of the attack graph generator. David Bianco's Pyramid of Pain in Figure 2.2 shows different indicators and Steffens provides in [98] details how indicators may be composed. Each IOC may also contain information about its source. This field is primarily for reference and documentation purposes. An author of an IOC might have used a certain source for creating specific data or a link to download fragments. Another potential use for this field consists in linking to third party databases, e.g. Security Information and Event Management (SIEM) or IDS systems. IOC have to be seen as part of threat intelligence information and thus should be properly documented regarding their source. This allows to establish an intelligence lifecycle as described in Section 2.1.

5.2.6 Target

A target is the object that represents real world systems, networks or devices. Sometimes they are also called assets, but in industrial environments assets are usually physical components that are part of a process. Those targets are the elements that make up the environment in which an adversary is active and where activities of an attack graph execution take place. As described in Section 2.4.1, ICS environments are usually structured similar and contain certain generic asset types. These assets types are also referenced in MITRE ATT&CK for ICS framework. Assets referenced in the ICS framework are generic and may need to be specified, when attack graphs are put into action. An option to overcome this specification post-graph-creation is to model the environment to targets. Modeling of an infrastructure into a formal machine readable representation is beyond the scope of this thesis. When attacks should be simulated though, an initial mapping of real world components to the targets has to be performed as a first step.

The table Target contains relevant information about objects Techniques can be applied to. Every Target has a name that should identify it clearly. This name could be a type of component (e.g. historian) or a specific device (e.g. Siemens S7 315). Target naming should align with a standard model architecture, like depicted in Section 2.4.1. Though those architectures are comparably individual for each organization, common concepts can established.

To distinguish between IT and OT domain where necessary, a field is available. Assets that are available in both domains may be tagged as such. This distinction may be used to tailor an attack graph to one of the domains. The sector in which assets are used may also help in creating more appropriate attack groups. This sector and the domain is an attribute for every target in the knowledge base.

For actual emulation it might be relevant to a white-hat hacker to know, whether an asset mapped to a Technique is a single component or a group of components. Same applies for the distinction of generic targets, e.g. asset groups (any PLC) and specific target (e.g. one certain controller type).

5.2.7 Group

A group represents an existing, real adversary with specific behavior and attributed previous intrusions. The adversary behaviour is called TTP (see Section 2.1.3). Other commonly used synonyms for these groups are intrusion sets or campaigns. The naming of APT groups is usually artificial and is a product of security researchers discovering or analyzing threat actors. Some companies might give adversaries numbers, like APT28, other researchers use names of animals representing countries, like Fancy Bear, and even others might use names of metals from the periodic table, like Xenotime. This inconsistency makes working with adversary groups based on names difficult. MITRE uses the most common names in their repositories and associated groups in case a threat actor has been given multiple names.

To resolve issues with unclear naming, a description field is available. This field allows to detail modus operandi, TTP and further background information. Tracking and evolution of adversary groups is a topic that exceeds the scope of this thesis though.

Clear associations of threat groups to countries, sectors and to differentiate intention of groups is prone to errors. Groups overlap concerning their TTP and may even use other group's indicators and fragments as false flags. These attributes should be carefully used and may provide guidance instead of hard facts. The attribution problem is also covered in Section 2.1.

A classification into cybercrime or nation-state actor is not possible with clear distinction.

Some state actors might also act like a cybercriminal or a cybercrime actor might be backed or tolerated by nation states. Even with uncertainties related to attribution, a good mapping of groups to techniques and to IOC may be used to create more relevant attack graphs when used correctly as constraints (for details on attribution and its flaws refer to Section 2.1).

5.3 Limitations

This section focuses on limitations of the prototype and its approach. Limitations are present due to design, implementation or prototyping decisions (see Chapter 4 and 5). Following aspects have to be considered when utilizing the graph generator for a certain use case and when evaluating generated graphs themselves. Given use cases and requirements from Chapter 3, the following aspects are limitations currently present in the implemented prototype.

Models of infrastructures generalize and structure existing environments. They make use of common archictectural and design patterns, but will not match the real world in all its details. A generalization always imposes a limitation on operations, in this case attack modeling, performed on a model. Generated graphs are only as dynamic and specific as the underlying expert system data. An only generically modeled target environment will only result in generic targets being mapped to Techniques in instantiated attack nodes. Depending on the intended use of attack graphs, a graph with generic targets might not be feasible. During adversary emulation or simulation, targets of each Technique might need to be translated to an exercise or training environment. A human operator is more flexible and can also use more generic graphs. Results from the attack graph generator are a tradeoff between specific and realistic attacks from real life and generic models that allow for more possible graphs. Generic graphs may include combinations that are less likely to happen or may even be impossible to realize.

Reducing ATT&CK and its Tactics to a linear, sequential attack graph oversimplifies generation and execution of attack graphs. Tactics are generally sequential, but not as strict as the Cyber Kill Chain would be. In reality cyberattacks are complex and recursive with many dependencies. Adversarial groups in ATT&CK are mapped to Techniques, but without any specific order (see the ATT&CK Navigator for this purpose). Such properties can hardly be modeled with an approach like it is depicted in this thesis. Since goal of this thesis was not to create automatic attack tools, but to support adversary behavior execution with attack graphs, this limitation can be accepted. To create attack graphs resembling real attack paths, forensic investigations and timelining is required. This is beyond the scope of this thesis.

ICS environments are usually complex and often deviate from standard architectures like the Purdue model or the automation pyramid (see section 2.4.1). This complexity needs to be made manageable and necessary domain knowledge needs to be transferred into the knowledge base of the application for relevant attack graphs to be generated. The initial data set provided by MITRE can be considered a starting point for further refinement. Especially with the diversity encountered in different sectors using industrial control systems (see Section 2.4.1) a customization of techniques, targets and IOC is required. If graphs are e.g. used in a petrochemical facility, unmodified data sets might result in a targeted substation compromise that is only relevant in electric domains.

Another limitation stems from artificial reduction of an attack to a simple graph. The number of steps in an attack, the number of techniques and a strict sequential process reduces and simplifies complexity of intrusions, especially those for industrial control systems. Attacks described in Chapter 2.5 each have involved previously unknown vectors and exploits, called zero days. It is impossible to account for those vectors in a knowledge base without making assumptions that are beyond the scope of this thesis. A possibility to mitigate this limitation is to continuously add new attack vectors into the knowledge base. This can even be done if they are experimental or not yet found in-the-wild.

Considering primarily the ATT&CK framework for ICS also creates limitations. It is generally possible though for an operator to add further techniques and procedures to the prototype. Direct access to industrial control environments as an external adversary is very unlikely and breaches, like ones that resulted in power outages in Ukraine, involved an initial access via a office networks of targeted companies (see also Section 2.4). Office network level access has not been in scope of this thesis though. This means that only a part of a full attack chain has been covered. The aforementioned limitations are intentional though to maintain manageability.

As stated before, the application and knowledge base omit the difficulty involved in breaching a perimeter before gaining access to control environments. At the same time it also does not take into account an information an adversary might gain, while breaching an office perimeter (e.g. credentials, knowledge about an environment and information in the context of social engineering). Maintaining such a state[1] during graph generation or execution has been excluded from considerations. It is up to a user to take those parameters into account during debriefing or evaluation of an adversary behavior execution activity.

Changing constraints and parameters to graph generation after a successful generation iteration is not supported. If changes are required, a completely new graph with new parameters or a modified copy of an existing graph has to be made. This limitation should not affect users that want to use graphs for adversary behavior execution. If an existing graph needs to be edited, a user has the possibility to use an administrative interface to directly manipulate nodes in the database. This functionality can also be used when executing simulations or machine learning activities.

Even though the prototype has aforementioned limitations, key functionalities and concepts are available to answer the research questions from Section 1.1. Many identified limitations are picked up again in the Conclusion of this thesis (see Chapter 8) and their solution is subject to future research in this domain.

[1] A state contains, but is not limited to, adversary knowledge, detection capabilities, failures of Techniques and potentially many more attributes.

Chapter 6

Prototype Interfaces and Example Graphs

This section showcases some of the implementation and graph generation results. An interactive prototype made an evaluation of the Graph Design (Section 4.1) and Implementation (Section 5) easier and has shown a capability to derive attack graphs from MITRE ATT&CK for ICS.

6.1 User Interface

The following screenshots depict some of the main websites a user interacts with. These interfaces support requirements from Section 3.2 and show how different aspects of identified use cases have been implemented in a prototype. Not every feature has been realized fully or to a production level, but generated graphs and prototype functionality is sufficient to answer the research question from Section 1.1. To provide a reader with an impression of the application and its interfaces selected screenshots are described in the course of this section. All screenshots are available in the addendum of this thesis.

Figures 6.1 and 6.2 show the regular user interfaces. Exemplary screenshots of the administrative interfaces are placed in the addendum. Additional screenshots from the prototype are available in the addendum to this thesis as well. A fully interactive version is provided with the prototype itself.

The User Interface (UI) does not only display data, but guides a user through the process of constraint selection and graph interaction. To explain concepts and architecture as well as the database design, a user can also access system architecture and concept depicted Figures 4.7 and Figure 5.7 directly in the prototype. During expert interviews, this helped to discuss architectural questions.

The administrative UI is derived from standard Django administration components and not customized visually. Functionally, additional filters and selectors are implemented to allow for quicker manipulation of existing data and entry of new data. Even though usability of this interface can be improved, it already shows how data manipulation can occur without directly interfacing with the database.

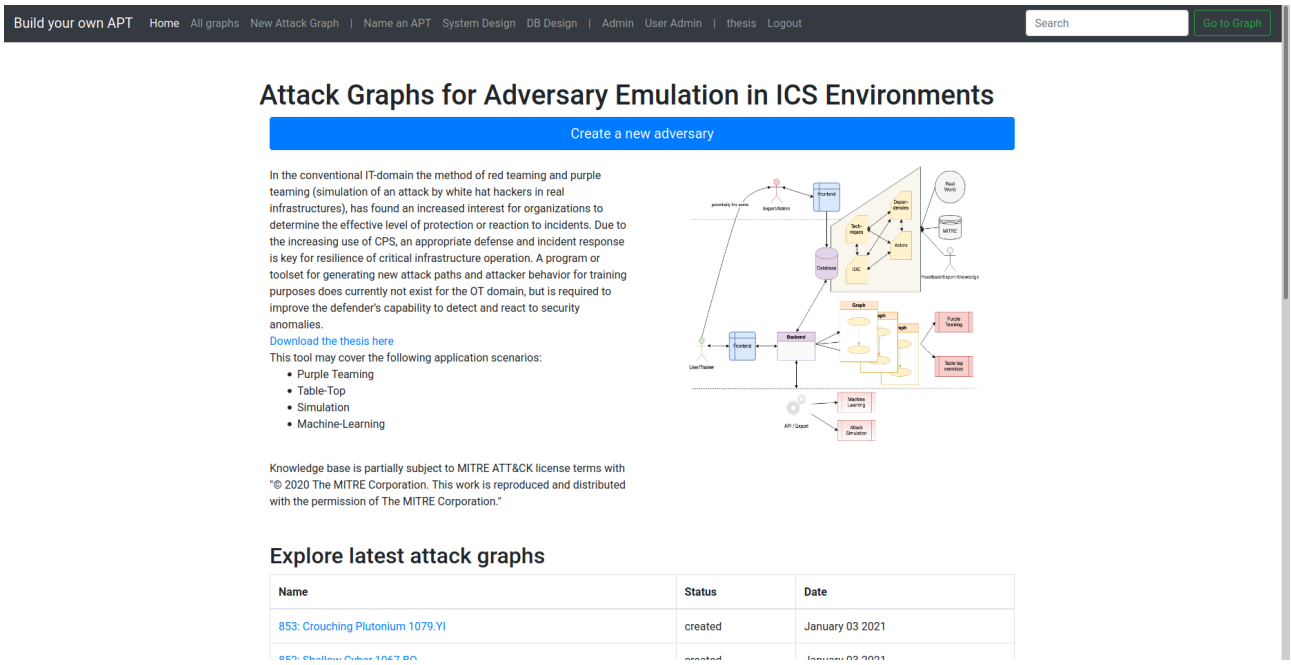


Figure 6.1: User Interface

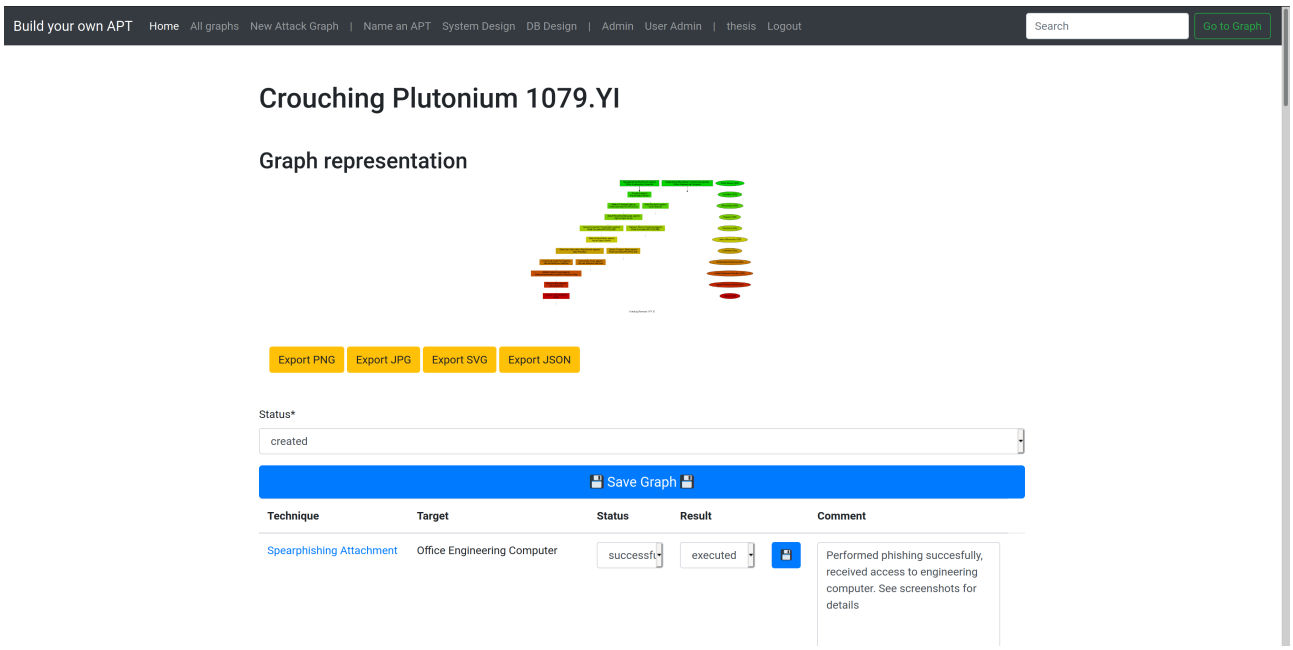


Figure 6.2: User Interface

6.2 Example Graphs

To verify in how far generated graphs can support adversary emulation and simulation multiple iterations of graph generation with different parameters have been performed. Results are graphs of different sizes and different constraints. To demonstrate capabilities of the graph generator a set of graphs has been chosen and are discussed in the following sections of this thesis. All graphs have been generated with a standard data set of MITRE ATT&CK for ICS framework data and have not been modified afterwards with an exception to constraints regarding targets and metadata.

Three different graphs and corresponding playbooks have been generated. They have been added to the addendum of this thesis. Aforementioned generated graphs show a versatility in graph generation, capabilities of the default knowledge base and a possibility to assist adversary emulation and simulation with previously unknown threat actor behavior. Each generated graph type acts as an input for evaluation (see Chapter 7) and interviews in Section 7.3. By using different graphs that maintain similar parameters for each evaluation, different interview partners give comparable results. In contrast, when just using random graphs evaluation, results are hardly comparable.

The first graph (depicted below in Figure 6.3) represents a completely random graph without any constraints by a user. Limitations of certain techniques imposed by the knowledge base are still in place, otherwise attack paths might not be consistent regarding combination of targets and techniques or sequence of events.

The second graph (added to the addendum in Figure 8.10) is a constrained graph, where a user has provided a set of techniques and targets that must not be used. All TTP that are related to discovery or lateral movement have been excluded. This could be relevant for a user with a target environment, where lateral movement and discovery techniques might result in impairments to the infrastructure. Additionally, the maximum width of the graph has been increased to three to have a more complex adversarial profile.

The third graph (added to the addendum in Figure 8.11 is also constrained. In this case, the graph generation function only had access to TTP known to be used by Sandworm (Ukraine incident) and XENOTIME (Trisis incident) (for details see Section 2.4.2). This graph demonstrates the possibility to create a fusion of two known threat actors and thus creating a “new” adversary that has not been encountered yet and might challenge a defender team.

Figure 6.3: Examplary generated pseudo-random attack graph

Chapter 7

Evaluation

After designing an approach and implementing a solution to answer the research question from Section 1.1. This approach as well as the solution has to be critically reflected and validated against experts in threat intelligence, adversary behavior execution and threat modeling in ICS infrastructures. The following sections describe the method of evaluation, an author's review as well as summarized results from expert interviews.

The interviews provided valuable insights to validate generated attack graphs and the prototype itself. Where possible, comments and identified bugs were corrected or implemented. Details on results and conclusions from the interviews are available in Section 7.3.

7.1 Method of Evaluation

This section describes the evaluation approach of this thesis' results. An iterative evaluation process was not considered due to constraints from a methodology (see Section 1.2) and time perspective. Interviews have been performed, when a prototype was available. This ex-post evaluation provides not only feedback on the current level of prototype implementation, but also gathers input for future research and future work. The evaluation is two-fold: the author's review and a set of expert interviews with a walkthrough of application and graphs each.

Goal of this evaluation is to identify, whether the implemented prototype and corresponding graphs are suitable for adversary behavior execution in ICS. Additionally, requirements should be validated and potential future research areas identified.

Before conducting expert interviews a questionnaire was developed that covers questions on an experts background, relevance and goals of adversary emulations and feedback on prototype and graphs. In a second step a number of experts in domains like threat intelligence, adversary behavior execution, OT and threat modeling have been selected. To also assess requirements and opinions of non-corporate experts, the expert group also contains researchers and professors. A consideration of government or other experts (e.g. military) failed, since none were able to participate in an interview within the given time frame. The experts that were consulted as part of the following interviews are not representative. Each interview was scheduled to take one hour and was performed remote via videoconferencing. The prototype and graphs were shown via screen sharing of the live application. A representative and all-encompassing survey was beyond the scope of this thesis and would not directly benefit in answering the research question.

The questionnaire is designed to assess an expert's knowledge and involvement in different domains relevant to this thesis, provide context to adversary emulation and requirements on

attack graphs as well as provide feedback on the prototype and generated graphs. The questions act as a validation instance for approaches and concepts employed in developing the prototype. Additionally, they allow to deduct in how far the prototype may be used in an adversary behavior execution context within ICS environments. Experts were not provided questions or access to the prototype before the individual interviews. This prevents bias during the actual conversation. Following is a summary of interview results as well as a reflection this thesis' results by the author.

7.2 Personal Review

Before providing summary interview results, this section covers a personal review of results by this thesis' author. The review takes the same considerations into account as the questionnaire for the interviews does. Section 5.3 already provides an overview on limitations that are due to design or implementation decisions. Every model has its limitations and so do attack graphs generated by the prototype of this thesis. The approach to design attack graphs and a corresponding prototype to answer the research question has proven viable and playbooks for Purple Teaming can be automatically generated based on a user's constraints.

Generated graphs generally fulfil requirements and correctness stated in Chapter 3. Due to the design of graph generation, all graphs have to be syntactically correct and can only contain valid chains of attacks (with regards to the knowledge base). Quality and relevance of graphs are dependent on a skilled user and a comprehensive dataset in the knowledge base. Requirements regarding functional and non-functional aspects for an attack graph generator have been fulfilled with design decisions and concepts in Chapter 4. Even though only a prototype has been developed and only a comparably small, but de-facto standard, knowledge base was available, generated graphs contain enough information for a human operator to use and document adversary behavior execution on ICS environments.

Though a simulation was not within scope of this thesis, it remains an open question in how far graphs with a detail level of this thesis are suitable for simulative approaches. Other attack graph concepts have shown: once an appropriate model of a target environment is available, simulation can be performed based on attack graphs. Modelling complex infrastructures is beyond the scope of this thesis though. The implemented prototype generally provides already machine readable representations, but a full machine-learning use case still has to be realized in future work.

Datasets providing information on ICS attacks and their specifics are scarce and have to be put into context. Documented incidents show that ICS intrusions have similarities to conventional IT intrusion, but at the same time have key differences. This makes a transfer of datasets from one domain to another difficult. Given the comparably generic nature of the ATT&CK for ICS framework, the knowledge base either has to be modified for each target organization or a human operator has to be able to translate generic targets and hacking steps to actual activities.

During testing of the prototype, not all graphs have resulted in end-to-end attack chains, but some iterations produced fragmented individual graphs. Though this behavior is explainable with the knowledge base and the graph generation algorithm, such fragmented graphs are not desirable from an exercise perspective. It leaves room for interpretation how an adversary performed a transition from one graph to another and complicates debriefing activities. Additionally, while also dependent on algorithm and knowledge base constraints, a lateral movement within individual Tactics has not been realized. Adversary behavior execution does not nec-

essarily need chronological execution for Purple Teaming exercises, but when also taking time and previous activities into account, an extension of the graph design is necessary.

Attack graphs in general have seen a high coverage in academic research and multiple approaches and use cases. With some exceptions, only few organization are utilizing those academic graphs in practice. This was considered during graph design, to allow a practitioner to make use of graphs without raising complexity to an unusable level.

Adversary behavior execution in general has potential to improve defenders and to effectively assess a security level. As already stated in Section 2.2 a certain maturity level needs to be available before activities results in any benefits. Using attack graphs to structure those activities is an effective means to support the whole exercise process. Use cases identified in Section 3.1 can benefit from results of this thesis.

Limitations depicted in Section 5.3 already show areas for improvement and further development of a prototype. Generally, adding additional features or more complexity has to be weighed against ease-of-use. To independently evaluate the results of this thesis a set of expert interviews were performed. Those results and findings are covered in the next Section 7.3.

7.3 Expert Interview Results

Expert interviews performed as part of this thesis occurred in November and December 2020. Anonymized summaries of those interviews are included in the addendum to this thesis. The questionnaire used to lead through the interviews has been added to the addendum as well. In addition to interviews with experts that work for companies with ICS connections, results on this thesis have also been discussed with experts from academia. Overall, thirteen experts agreed to participate in the interviews and provided valuable feedback. Details regarding goals and method of those interviews can be found in Section 7.1. Each interview was scheduled for one hour and where necessary extended to allow room for discussion.

7.3.1 Expert's Background

Most of the experts had some relationship to ICS environments, but answers also showed that there is no "one-size-fits-all"-definition for this domain and that know-how ranged from threat modeling to penetration testing. Only a few participants originated from an engineering domain and moved into cybersecurity. This shows that even though ICS is an engineering field, security is still often looked at from an IT perspective. Projects like this thesis may be able to change that.

Most experts already are doing some type of adversary behavior execution, but only two stated that they are already doing full Purple/Red Teaming exercises. The concept of differentiating emulation and simulation required explanation during interviews and combining both aspect into "adversary behavior execution" helped to build on similar terms. A clear differentiation between emulation and simulation was also not possible for many experts, since in reality most activities are hybrid.

The ATT&CK framework was at least cursory known by all participants. The level of use and proficiency varied. One expert is author of multiple ATT&CK Techniques and can be considered most involved into the project. Especially German speaking experts and those not working in consultancies do not utilize MITRE frameworks (yet). Other methodologies or less technical approaches have been used in those cases. ATT&CK was mostly regarded as a

reference framework to validate and implement defensive measures, but some already use it to plan for emulation and simulation as it is intended in this thesis.

7.3.2 Purple Teaming Exercises and Attack Graphs

Adversary behavior execution or Purple Teaming has not yet been realized by most of the experts. Single aspects, like table-top exercises or penetration testing, are common, but a full-scale reproduction of adversarial behavior has only been performed by one expert. That expert works in a training center specializing on Purple Teaming exercises. Generally, the concept of Purple Teaming gains attention as per the experts' opinions. Organizations often still lack a maturity of security organization and technology to gain value from these types of exercises. As goal for such activities were named among others: training the defender team with realistic scenarios, validating security controls, improving detection and response processes and many more. Realistic scenarios were named as one of the key success criteria.

Except for one expert performing Red Teaming in a dedicated training environment for ICS, none have done active, potentially disruptive, exercises in ICS environments. All participants stated that industrial environments have special requirements and intricacies. Operators are often reluctant to take unnecessary risks. Where in IT environments "only" IT systems are affected, are physical processes involved with OT environments and those may also impact safety. Cybersecurity in OT is also still a comparably new field, in which resources according to experts are scarce. Existing resources are more effectively used in directly improving security than by performing exercises.

Attack graphs as a general concept are well known to participants of the interviews. One expert modeled the Ukraine 2015 incident as an attack graph (see [95]), but did not directly use any of the existing approaches covered in Section 2 of this thesis' foundations. Many interviewees considered existing approaches to be too complex for real-world use. The idea to structure and describe approaches as graphs received good feedback though.

7.3.3 Protype and Graph Walkthrough

Interviewed experts all have different backgrounds and thus naturally had different requirements for attack graphs and a tool that supports scenario generation and execution of those scenarios. This shows that there is not one solution to support all activities in this domain, but a general concept needs to be adapted to needs and use cases of operators. A common requirement was using real world data as input, ideally peer reviewed and community approved. Other experts wanted to have an integration with existing infrastructure description to model attack graphs against. More consultant- and management-centric experts required visualization of graphs and ease-of-use and ease-of-understanding for their daily job. Many requirements identified in Chapter 3 have also been confirmed by the experts. This makes them viable as part of this thesis.

As part of the interviews, all experts were guided through the prototype and generated graphs. All experts confirmed that generated graphs were generally valid attack chains. Many stated that this is since they were derived from a public dataset like MITRE ATT&CK. Not all graphs were flawless and experts required some amendments to e.g. targets or Techniques, if those graphs should be put to practical use. Since graphs were not pre-created, but created during walkthroughs, some flaws in the knowledge base's constraints resulted in discussions. Discussion of generated graphs showed that they are not self-explanatory. Most experts considered shown example graphs as a single successful attack and not always as one complex adversary profile.

But even with flawed graphs, the experts confirmed that this thesis' attack graphs can be used in an adversary behavior execution or in Purple Teaming scenarios.

In general, interviewed experts expressed a high interest in the idea to use attack graphs for attack modeling and also using it for adversary emulation. Even though adversary behavior execution has not yet found a regular use in ICS environments, attack graphs may also be used as part of risk assessments and threat modeling. The workflow and application itself was deemed appropriate and fit-for-purpose. The depicted lifecycle from planning, to execution and finally to documentation has been regarded as suitable for use cases like Purple Teaming exercises. Since shown application was a prototype, there has also been expert feedback on room for improvement. This also included ideas regarding the application workflow itself.

Even though the prototype already implemented a workflow to generate graphs and it was possible to generate graphs, experts provided recommendations or even identified flaws. One commonly expressed improvement aspect lies in the constraint selection. Many considered an exclusion-only model as counterintuitive. Some experts also desired more control about graph generation parameters and transparency on the process. Others expressed a need to have a legend next to each graph to explain each phase. Some bugs that were identified during the demonstration have already been mitigated shortly after the interview. That included missing length checks on fields and other minor flaws. Overall, quality and functionality of the prototype was deemed appropriate and fit-for-purpose by the experts.

7.3.4 Use Cases

During the expert interviews additional ideas for future work and research were identified: either from the experts own field of expertise or as part the discussion on generated attack graphs and application walkthrough. Discussions during the interviews have been very valuable and show that adversary behavior execution has potential. The potential for future work can generally be divided into three categories: defense modeling, guided graph generation and risk/threat modeling. Each category will be covered in the following paragraphs.

In the context of defense modeling experts mentioned potential to use attack graphs and a corresponding application to properly plan security controls. Results from graph executions can be used to select appropriate measures to counter threats. With a simple walkthrough of an attack graph, weak spots in detection and prevention capabilities might be identified and, if necessary, countered. One expert recommended an evolutionary approach, expanding attack graphs and subsequently implementing more and more counter measures as appropriate.

A second often suggested use case and extension of the prototype is in a guided graph generation approach. Instead of giving away already built graphs, an operator could be guided to create graphs compliant with the knowledge base and at the same time customize it to a target environment. Another expert recommend to use graph generation for post-incident documentation and intrusion analysis to identify potential access vectors.

As a third and most mentioned category interview participants named a risk analysis and threat modeling use case. Attack graphs can help understand and visualize risks, if they are complemented with additional information about probabilities and impact on operations. As part of table-top exercises threats can be evaluated and appropriate risk management decided upon. Attack graphs may be used as an additional tool to validate assumptions.

Summarizing, the expert interviews confirmed requirements and provided an independent review of generated attack graphs for feasibility. These results as well as the author's personal review can be used in the following Conclusion, Chapter 8, to answer the research question.

Chapter 8

Conclusion

As already outlined in the introduction and motivation to this thesis, cyberattacks are becoming more sophisticated and previously air-gapped and even components that are not computerized become a liability for many organizations. Digitalization is a chance for higher productivity, but also results in security and safety risk, when systems are abused or fail. This is especially the case for industrial control systems, why research concentrated on this domain.

Defenders need to be empowered to detect and react to incidents in their ICS environments. Training scenarios for the OT domain are scarce and often simply reproduce incidents that were encountered in the past. This thesis solves the issue with static scenarios by providing pseudo-randomized attack graphs that represent new, previously unknown scenarios. Limited capabilities of ICS operators for trainings can be efficiently used in Purple Teaming exercises that identify deficiencies while improving defender's know-how and processes.

The goal of this thesis was to provide an answer to the research question in Section 1.1 whether it is possible to support adversary behavior emulation and simulation in ICS with attack graphs and a corresponding tool. Summarizing, attack graphs outlined in this thesis are suitable to be used in aforementioned use cases and that they may be generated and executed with a tool. They provide previously unknown, but realistic, scenarios from a community-accepted data repository.

Given foundations on cyberattacks and ontologies, adversary behavior execution, attack graphs and industrial control systems, requirements were derived for a solution to the research question. Those foundations have been considered during identification of necessary use cases for attack graphs and graph design as well as the prototype design of an expert system.

Previous chapters and the implemented prototype show that it is possible to generate suitable attack graphs from public data repositories like MITRE ATT&CK. Existing attack graph approaches are not suitable to act as a basis for adversary behavior execution, why in this thesis a simple and versatile graph design has been developed. The prototype's graph have shown syntactic validity, but need to interpreted by an expert to be used in real-life scenarios.

Expert interviews have shown that scope of applicability of attack graphs and use cases can be extended to threat modeling, defense modeling and practical identification of weak spots in organizations. Even though maturity in ICS environments has not yet reached a level where adversary behavior execution is fully feasible, individual aspects of emulation and simulation have already been introduced into the industrial IT domain. The prototype of this thesis can be used according to the experts in many aspects of the information security context in ICS environments.

Given aforementioned feedback during expert interviews (Section7.3), critical personal evaluation (Section 7.2 and identified limitations of the current implementation (Section 5.3), multiple

fields for potential future work have been identified.

Attack graphs, cyberattacks and modeling of offensive and defensive activities are extensive domains. The prototype may be extended in future work, concepts and designs refined and additional use cases implemented. As previously described, not all features of attack graphs and use cases have been covered by the prototype.

One major contributing factor for potential research brings the convergence of IT and OT. Systems are becoming more and more similar and even though their protection goals may differ, IT threats will grow into the OT domain. As technology converges, threats converge as well. A distinction might not even be feasible in the future any more. Adding new Techniques and also IT attacks to the knowledge base allows to prepare for this convergence with adversary behavior execution.

The prototype already has basic project management capabilities for adversary behavior execution and exercises. This project management can be extended to an end-to-end solution for exercise management.

Another promising research field lies in an integration of formal attack description languages (like MAL) and infrastructure modeling notation. The potential when environments can be properly used as inputs for attack graphs and defense modeling is immense. This may be achieved by an automatic transformation of infrastructure and function diagrams to machine readable formats and subsequently into tools like the attack graph generator. Existing infrastructure documentation may also be directly transformed into a machine-readable format and ingested into the attack graph generator.

In future work, the tool set and the attack graphs may also be augmented by lower-level techniques and even malware samples and adversary strings that allow for a hands-on implementation of Red and Purple Teaming. Adding lower level data sets also increase capabilities for simulation and machine learning.

A machine learning use case has been prepared with the current prototype, but not yet realized. Future research may include aspects like adversarial resilience learning to improve detection and response for defenders and optimization of attack steps for Red Teams. Todays intrusion detection systems are more and more moving to heuristic approaches that may also be augmented by attack graphs and a corresponding tool.

The prototype primarily uses publicly available data derived from MITRE ATT&CK. With data sets that contain likelihood, difficulty and impact for each attack step, it is also possible to create Bayesian networks. Using those countermeasures or attacks may be planned with maximum return on invest. Attack graphs could also aid in decision making processes for architectures and security assessments, when they are translated into risk.

Another extension of attack graphs shown in this thesis could be establishing defender profiles (mapping Blue Team capabilities) to show, whether a certain chain of attacks should be detected and to validate protective efforts to be effective. Threat and defense modeling is one logical extension of attack graphs.

This thesis succeeded in positively answering the research question and validating its results against experts. During the research, many new fields for future work have been identified that will augment the concepts of attack graphs for adversary behavior execution provided in this work.

References

[1] Magnus Åkerman. *Implementing Shop Floor IT for Industry 4.0*. PhD thesis, Chalmers Tekniska Hogskola (Sweden), 2018. https://www.researchgate.net/publication/326224890_Implementing_Shop_Floor_IT_for_Industry_40.

[2] Otis Alexander, Misha Belisle, and Jacob Steele. MITRE ATT&CK for Industrial Control Systems: Design and Philosophy. *MITRE Product*, 2020.

[3] Mohammed Alhomidi and Martin Reed. Attack graph-based risk assessment and optimisation approach. *International Journal of Network Security & Its Applications*, 6(3):31, 2014.

[4] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224, 2002.

[5] Andy Applebaum, Doug Miller, Blake Strom, Henry Foster, and Cody Thomas. Analysis of automated adversary emulation techniques. In *Proceedings of the Summer Simulation Multi-Conference*, pages 1–12, 2017.

[6] Andy Applebaum, Doug Miller, Blake Strom, Chris Korban, and Ross Wolf. Intelligent, automated red team emulation. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 363–373, 2016.

[7] Michael J Assante and Robert M Lee. The Industrial Control System Cyber Kill Chain. *SANS Institute InfoSec Reading Room*, 1, 2015. https://www.sans.org/reading-room/whitepapers/ICS/industrial-control-system-cyber-kill-chain-36297 (accessed September 21, 2020).

[8] European Central Bank. TIBER-EU FRAMEWORK - How to implement the European framework for Threat Intelligence-based Ethical Red Teaming. Europa.eu, 2018. https://www.ecb.europa.eu/pub/pdf/other/ecb.tiber_eu_framework.en.pdf (accessed October 28, 2020).

[9] Bundesverband der Energie-und Wasserwirtschaft e.V. BDEW. Whitepaper requirements for secure control and telecommunication systems, May 2018. https://www.bdew.de/media/documents/Awh_20180507_OE-BDEW-Whitepaper-Secure-Systems-engl.pdf (accessed October 12, 2020).

[10] David Bianco. The Pyramid of Pain, 2014. http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html (accessed November 6, 2020).

[11] Bernhards Blumbergs. *Specialized Cyber Red Team Responsive Computer Network Operations*. PhD thesis, Tallinna Tehnikaülikool, 2019. https://digi.lib.ttu.ee/i/?12015.

[12] Sunders Bruskin, Polina Zilberman, Rami Puzis, and Shay Shwarz. Sok: A survey of open source threat emulators. *arXiv preprint arXiv:2003.01518*, 2020.

[13] Jen Burns and Anthony Masi. ATT&CK content available in STIX. MITRE Cybersecurity Blog, may 2018. https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/attck%E2%84%A2-content-available-in-stix%E2%84%A2-20-via (accessed October 23, 2020).

[14] Christina Büsing. *Graphen-und Netzwerkoptimierung*. Springer-Verlag, 2010.

[15] Eric Byres. The air gap: SCADA's enduring security myth. *Communications of the ACM*, 56(8):29–31, 2013. https://cacm.acm.org/magazines/2013/8/166309-the-air-gap.

[16] Sergio Caltagirone, Andrew Pendergast, and Christopher Betz. The Diamond Model of Intrusion Analysis. Technical report, Center For Cyber Intelligence Analysis and Threat Research Hanover Md, 2013. https://apps.dtic.mil/sti/pdfs/ADA586960.pdf.

[17] Timothy Casey. Threat agent library helps identify information security risks. *Intel White Paper*, 2, 2007. http://making-security-measurable.1364806.n2.nabble.com/attachment/7587486/0/Intel%20Corp_Threat%20Agent%20Library_07-2202w.pdf (accessed November 3, 2020).

[18] Center for Internet Security, CIS. Tabletop Exercises - Six Scenarios to Help Prepare Your Cybersecurity Team, October 2018. https://www.cisecurity.org/wp-content/uploads/2018/10/Six-tabletop-exercises-FINAL.pdf (accessed October 9, 2020).

[19] Davide Cerotti, Daniele Codetta Raiteri, Giovanna Dondossola, Lavinia Egidi, Giuliana Franceschinis, Luigi Portinale, and Roberta Terruggia. A Bayesian Network Approach for the Interpretation of Cyber Attacks to Power Systems. In *ITASEC*, 2019.

[20] Cybersecurity and Infrastructure Security Agency (CISA). Alert (TA17-293A) Advanced Persistent Threat Activity Targeting Energy and Other Critical Infrastructure Sectors. CISA Alerts, March 2018. https://us-cert.cisa.gov/ncas/alerts/TA17-293A (accessed November 3, 2020).

[21] Chris Dale. Red, Blue and Purple Teams: Combining Your Security Capabilities for the Best Outcome. SANS Institute, October 2019. http://www3.sans.org/media/analyst-program/red-blue-purple-teams-combining-security-capabilities-outcome-39190.pdf (accessed October 23, 2020).

[22] A. de Ruijter and F. Guldenmund. The bowtie method: A review. *Safety Science*, 88:211 – 218, 2016.

[23] Alessandro Di Pinto, Younes Dragoni, and Andrea Carcano. TRITON: The First ICS Cyber Attack on Safety Instrument Systems. Proc. Black Hat USA, 2018. https://i.blackhat.com/us-18/Wed-August-8/us-18-Carcano-TRITON-How-It-Disrupted-Safety-Systems-And-Changed-The-Threat-Landscape-Of-Industrial-Control-Systems-Forever-wp.pdf (accessed November 3, 2020).

[24] Wilhem Dolle and Jan Hoff. BSI KRITIS Sektorstudie Energie, 2015. https://www.kritis.bund.de/SharedDocs/Downloads/Kritis/DE/Sektorstudie_Energie.pdf (accessed November 3, 2020).

[25] Mathias Ekstedt, Pontus Johnson, Robert Lagerström, Dan Gorton, Joakim Nydrén, and Khurram Shahzad. Securi CAD by Foreseeti: A CAD Tool for Enterprise Cyber Security Management. In *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, pages 152–155. IEEE, 2015.

[26] Lars Fischer, Jan-Menno Memmen, Eric M. S. P. Veith, and Martin Tröschel. Adversarial Resilience Learning — Towards Systematic Vulnerability Analysis for Large and Complex Systems. In *ENERGY 2019, The Ninth International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, page 24 to 32, 2019.

[27] Lars Fischer and Michel Messerschmidt. Ohne Security keine Safety in Kritischen Infrastrukturen — Begriffliche Trennung und Zusammenführung, 2020. https://ag.kritis.info/wp-content/uploads/2020/05/AG_KRITIS-Ohne_Safety_keine_Security_in_KRITIS.pdf (accessed August 3, 2020).

[28] Nigel George. Django's Structure – A Heretic's Eye View, 2020. https://djangobook.com/mdj2-django-structure/ (accessed June 18, 2020).

[29] Martin Glinz. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 21–26. IEEE, 2007.

[30] Benjamin Green, Sylvain Andre Francis Frey, Awais Rashid, and David Hutchison. Testbed diversity as a fundamental principle for effective ICS security research. *Serecin*, 2016.

[31] Andy Greenberg. The Untold Story of NotPetya, the Most Devastating Cyberattack in History, 2006. https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/ (accessed November 3, 2020).

[32] Andy Greenberg. *Sandworm: A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers*. Doubleday, 2019.

[33] Simon Hacks, Alexander Hacks, Sotirios Katsikeas, Benedikt Klaer, and Robert Lagerström. Creating Meta Attack Language Instances using ArchiMate: Applied to Electric Power and Energy System Cases. In *2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)*, pages 88–97. IEEE, 2019.

[34] Booz Allen Hamilton. When the lights went out: Ukraine cybersecurity threat briefing, 2016. https://www.boozallen.com/content/dam/boozallen/documents/2016/09/ukraine-report-when-the-lights-went-out.pdf (accessed November 3, 2020).

[35] Jörg Hoffmann. Simulated penetration testing: from" Dijkstra" to" Turing Test++". In *Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015.

[36] Eric M Hutchins, Michael J Cloppert, Rohan M Amin, et al. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011. https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf (accessed October 11, 2020).

[37] Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 121–130. IEEE, 2006.

[38] ISO Central Secretary. Industrial Communication Networks — Network and system security-Part 2-1: Establishing an industrial automation and control system security program. Standard, International Organization for Standardization, Geneva, CH, 2010.

[39] ISO Central Secretary. Information technology — Security techniques — Information security management systems — Overview and vocabulary. Standard ISO/IEC 27000:2018(E), International Organization for Standardization, Geneva, CH, 2018.

[40] Somesh Jha, Oleg Sheyner, and Jeannette Wing. Two formal analyses of attack graphs. In *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, pages 49–63. IEEE, 2002.

[41] Chris Johnson. Securing the Participation of Safety-Critical SCADA Systems in the Industrial Internet of Things, 2016.

[42] Christopher Johnson, Mark Badger, David Waltermire, Julie Snyder, and Clem Skorupka. Guide to cyber threat information sharing. Technical report, National Institute of Standards and Technology, 2016.

[43] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. A Meta Language for Threat Modeling and Attack Simulations. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–8, 2018.

[44] Kerem Kaynar. A taxonomy for attack graph generation and usage in network security. *Journal of Information Security and Applications*, 29:27–56, 2016.

[45] Jason Kick. Cyber exercise playbook. Technical report, The MITRE Corporation, 2014. https://apps.dtic.mil/sti/pdfs/ADA624910.pdf (accessed November 3, 2020).

[46] Eric D Knapp and Joel Thomas Langill. *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Syngress, 2014.

[47] Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Patrick Schweitzer. Foundations of attack–defense trees. In *International Workshop on Formal Aspects in Security and Trust*, pages 80–95. Springer, 2010.

[48] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer science review*, 13:1–38, 2014.

[49] Sara Kraemer, Pascale Carayon, and Ruth Duggan. Red team performance for improved computer security. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48, pages 1605–1609. Sage Publications Sage CA: Los Angeles, CA, 2004.

[50] Glenn E Krasner, Stephen T Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.

[51] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review*, 35:100219, 2020.

[52] Ralph Langner. *Robust control system networks: How to achieve reliable control after Stuxnet*. Momentum Press, 2011.

[53] Matthew Lauder. Red dawn: The emergence of a red teaming capability in the Canadian forces. *Canadian Army Journal*, 12(2):25–36, 2009.

[54] Robert Lee, Joe Slowik, Ben Miller, Anton Cherepanov, and Robert Lipovsky. Industroyer/Crashoverride: Zero things cool about a threat group targeting the power grid, 2017. https://www.blackhat.com/docs/us-17/wednesday/us-17-Lee-Industroyer-Crashoverride-Zero-Things-Cool-About-A-Threat-Group-Targeting-The-Power-Grid.pdf (accessed November 6,2020).

[55] Robert M Lee, Michael J Assante, and Tim Conway. German Steel Mill Cyber Attack. *Industrial Control Systems*, 2014. https://ics.sans.org/media/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf (accessed September 20, 2020).

[56] Robert M. Lee, Michael J. Assante, and Tim Conway. Analysis of the cyber attack on the Ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 388, 2016.

[57] Antoine Lemay, Joan Calvet, François Menet, and José M Fernandez. Survey of publicly available reports on advanced persistent threat actors. *Computers & Security*, 72:26–59, 2018.

[58] Elizabeth LeMay, Michael D Ford, Ken Keefe, William H Sanders, and Carol Muehrcke. Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE). In *2011 Eighth International Conference on Quantitative Evaluation of SysTems*, pages 191–200. IEEE, September 2011.

[59] Olof Leps. Modellierung und Implementierung hybrider Testumgebungen für cyber-physische Sicherheitsanalysen. In *Hybride Testumgebungen für Kritische Infrastrukturen*, pages 69–119. Springer, 2018.

[60] Richard Lippmann, Kyle Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Mike Artz, and Robert Cunningham. Validating and restoring defense in depth using attack graphs. In *MILCOM 2006-2006 IEEE Military Communications Conference*, pages 1–10. IEEE, 2006.

[61] Viliam Lisỳ and Radek Píbil. Computing optimal attack strategies using unconstrained influence diagrams. In *Pacific-Asia Workshop on Intelligence and Security Informatics*, pages 38–46. Springer, 2013.

[62] Tim Malcomvetter. Emulation, Simulation, & False Flags. Medium, February 2020. https://medium.com/@malcomvetter/emulation-simulation-false-flags-b8f660734482 (accessed October 6, 2020).

[63] Sean Malone. Using an expanded cyber kill chain model to increase attack resiliency. *Black Hat US*, 2016.

[64] Vasileios Mavroeidis and Siri Bromander. 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)*, pages 91–98. IEEE, 2017.

[65] Rob McMillan. Definition: Threat Intelligence. *Gartner.com*, 2013. https://www.gartner.com/en/documents/2487216 (accessed October 11, 2020).

[66] Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing. Ranking attack graphs. In *International Workshop on Recent Advances in Intrusion Detection*, pages 127–144. Springer, 2006.

[67] Tobias Meudt, Malte Pohl, and Joachim Metternich. Die Automatisierungspyramide - Ein Literaturüberblick. TUprints, June 2017.

[68] Doug Miller, Ron Alford, Andy Applebaum, Henry Foster, Caleb Little, and Blake Strom. Automated adversary emulation: A case for planning and acting with unknowns. *MITRE Technical Papers*, 2018.

[69] MITRE. ATT&CK for Industrial Control Systems, 2020. https://collaborate.mitre.org/attackics/index.php/Main_Page (accessed September 27, 2020).

[70] Glenn Murray, Michael N Johnstone, and Craig Valli. The convergence of IT and OT in critical infrastructure. *Proceedings of 15th Australian Information Security Management Conference*, pages 149–155, December 2017.

[71] Thanh Thi Nguyen and Vijay Janapa Reddi. Deep reinforcement learning for cyber security, 2020.

[72] Andrew Nicholson, Stuart Webber, Shaun Dyer, Tanuja Patel, and Helge Janicke. SCADA security in the light of Cyber-Warfare. *Computers & Security*, 31(4):418–436, 2012.

[73] SP NIST. 800-53, revision 4. *Security and Privacy Controls for Federal Information Systems and Organizations*, 2013.

[74] Steven Noel, Sushil Jajodia, Lingyu Wang, and Anoop Singhal. Measuring security risk of networks using attack graphs. *International Journal of Next-Generation Computing*, 1(1):135–147, 2010.

[75] Jacob G Oakley. *Professional Red Teaming: Conducting Successful Cybersecurity Engagements*. Apress, 2019.

[76] Xinming Ou, Wayne F Boyer, and Miles A McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 336–345, 2006.

[77] E Ouzounis, P Trimintzios, and P Saragiotis. Good practice guide on national exercises, 2009. https://www.enisa.europa.eu/publications/national-exercise-good-practice-guide (accessed November 3, 2020).

[78] Adrian Pauna, Konstantinos Moulinos, Matina Lakka, J May, and T Tryfonas. Can we learn from scada security incidents. *White Paper, European Union Agency for Network and Information Security, Heraklion, Crete, Greece*, 2013.

[79] P Pawlinski, P Jaroszewski, J Urbanowicz, P Jacewicz, P Zielony, P Kijewski, and K Gorzelak. Standards and tools for exchange and processing of actionable information. *European Union Agency for Network and Information Security, Heraklion, Greece*, 2014.

[80] C. Phillips and L. Swiler. A graph-based system for network-vulnerability analysis. In *NSPW '98*, 1998.

[81] Paul Pols and Jan van den Berg. The Unified Kill Chain. *CSA Thesis, Hague*, pages 1–104, 2017.

[82] Bob Radvanovsky. Debate over IT, OT and Control Systems, Nov 2019. https://web.archive.org/web/20200224195733/http://icsmodel.infracritical.com/ (accessed November 3, 2020).

[83] Thomas Rid and Ben Buchanan. Attributing cyber attacks. *Journal of Strategic Studies*, 38(1-2):4–37, 2015.

[84] SJ Rockefeller. A kill chain analysis of the 2013 target data breach. *tech. rep., Committee on Commerce, Science and Transportation, Tech. Rep.*, 2014. https://web.archive.org/web/20161006082550/http://www.public.navy.mil/spawar/Press/Documents/Publications/03.26.15_USSenate.pdf.

[85] Martin Rosso. An Attack Simulation Methodology for Empirical SOC Performance Evaluation. Master's thesis, Radboud University Nijmegen, 08 2019. https://www.ru.nl/publish/pages/769526/z02_rosso_thesis.pdf.

[86] Marsha D Rowell. Cyber indicators of compromise: a domain ontology for security information and event management. Technical report, Naval Postgraduate School Monterey United States, 2017.

[87] Sudip Saha, Anil Kumar S Vullikanti, Mahantesh Halappanavar, and Samrat Chatterjee. Identifying vulnerabilities and hardening attack graphs for networked systems. In *2016 IEEE Symposium on Technologies for Homeland Security (HST)*, pages 1–6. IEEE, 2016.

[88] Chris Salter, O Sami Saydjari, Bruce Schneier, and Jim Wallner. Toward a secure system engineering methodolgy. In *Proceedings of the 1998 workshop on New security paradigms*, pages 2–10, 1998.

[89] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[90] Austin Scott. Purple Teaming ICS Networks, 2019. https://dragos.com/blog/industry-news/purple-teaming-ics-networks-part-1-of-3/ (accessed November 3, 2020).

[91] Ensar Seker and Hasan Huseyin Ozbenli. The Concept of Cyber Defence Exercises (CDX): Planning, Execution, Evaluation. In *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–9. IEEE, 2018.

[92] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 273–284. IEEE, 2002.

[93] Oleg Sheyner and Jeannette Wing. Tools for generating and analyzing attack graphs. In *International symposium on formal methods for components and objects*, pages 344–371. Springer, 2003.

[94] Manuel Silva. Logical controllers. *IFAC Proceedings Volumes*, 22(18):249–259, 1989.

[95] Joe Slowik. Anatomy of an attack: Detecting and defeating crashoverride. *VB2018, October*, 2018.

[96] Joseph Slowik. Evolution of ics attacks and the prospects for future disruptive events, 2019. https://www.dragos.com/wp-content/uploads/Evolution-of-ICS-Attacks-and-the-Prospects-for-Future-Disruptive/-Events-Joseph-Slowik-1.pdf (accessed November 3, 2020).

[97] Teodor Sommestad, Mathias Ekstedt, and Hannes Holm. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *IEEE Systems Journal*, 7(3):363–373, 2012.

[98] Timo Steffens. *Attribution of Advanced Persistent Threats*. Springer, 2020.

[99] STIX Project. TTP vs Indicator: A simple usage overview, 2018. https://stixproject.github.io/documentation/concepts/ttp-vs-indicator/ (accessed October 6, 2020).

[100] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to Industrial Control Systems (ICS) Security. *NIST special publication*, 800(82):16–16, 2011. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf (accessed September 22, 2020).

[101] Blake E Strom, Andy Applebaum, Douglas P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. MITRE ATT&CK: Design and Philosophy. *MITRE Product*, 2018.

[102] Chih-Che Sun, Adam Hahn, and Chen-Ching Liu. Cyber security of a power grid: State-of-the-art. *International Journal of Electrical Power & Energy Systems*, 99:45–56, 2018.

[103] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, volume 2, pages 307–321 vol.2, 2001.

[104] Wiem Tounsi and Helmi Rais. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & security*, 72:212–233, September 2018. https://www.researchgate.net/publication/320027747_A_survey_on_technical_threat_intelligence_in_the_age_of_sophisticated_cyber_attacks.

[105] Robert J Turk. Cyber incidents involving control systems. Technical report, Idaho National Laboratory (INL), 2005.

[106] Brian Van Leeuwen, Vincent Urias, John Eldridge, Charles Villamarin, and Ron Olsberg. Cyber security analysis testbed: Combining real, emulation, and simulation. In *44th Annual 2010 IEEE International Carnahan Conference on Security Technology*, pages 121–126. IEEE, 2010.

[107] Eric M. S.P̃. Veith, Lars Fischer, Martin Tröschel, and Astrid Nieße. Analyzing cyber-physical systems from the perspective of artificial intelligence. In *International Conference on Artificial Intelligence, Robotics and Control*, International Conference Proceedings by ACM, pages 1–9. ACM, 2019.

[108] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An attack graph-based probabilistic security metric. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 283–296. Springer, 2008.

[109] Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring the Overall Security of Network Configurations Using Attack Graphs. In *Proceedings of the 2007 ACM workshop on Quality of protection*, pages 49–54, 2007.

[110] Joe Weiss. Aurora Generator Test. *Handbook of SCADA/Control Systems Security*, pages 107–114, 2016.

[111] Nina Wilhelmson and Thomas Svensson. *Handbook for planning, running and evaluating information technology and cyber security exercises*. Försvarshögskolan (FHS), 2011.

[112] Theodore J Williams. The Purdue enterprise reference architecture. *Computers in industry*, 24(2-3):141–158, 1994.

[113] Jackson Wynn, Joseph Whitmore, William Coconato, and Samuel McCracken. Critical Infrastructure Cyberspace Analysis Tool (CICAT). *MITRE Product*, 2020. https://www.mitre.org/sites/default/files/publications/pr-20-0021-critical-infrastructure-cyberspace%20-analysis%20-tool-%28cicat%29-capability-description_0.pdf.

[114] Muhammad Mudassar Yamin, Basel Katt, and Vasileios Gkioulos. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. *Computers & Security*, page 101636, 2019.

[115] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. A taxonomy of cyber attacks on SCADA systems. In *2011 International conference on internet of things and 4th international conference on cyber, physical and social computing*, pages 380–388. IEEE, 2011.

Acronyms

AEG Attack Execution Graph
API Application Programming Interface
ANN Artificial Neural Networks
APT Advanced Persistent Threat
ATT&CK Adversarial Tactics, Techniques and Common Knowledge
BLOB Binary Large Object
CAPEC Common Attack Pattern Enumeration and Classification
CDX Cyber Defense Exercise
CERT Computer Emergency Readiness Team
CISA Cybersecurity & Infrastructure Security Agency
CPS Cyber Physical Systems
CVE Common Vulnerabilities and Exposures
CWE Common Weakness Enumeration
DAG Directed Acyclic Graph
ER Entity-Relationship
GML Graph Modelling Language
GNN Graph Neural Network
GUI Graphical User Interface
ICS Industrial Control System
IDS Intrusion Detection System
IOC Indicators of Compromise
JSON Javascript Object Notation
MDP Markov Decision Process
MAL Meta Attack Language
OT Operational Technology
PLC Programmable Logic Controller
SCADA Supervisory Control and Data Acquisition
SIEM Security Information and Event Management
SOC Security Operations Center
STIX Structured Threat Information Expression
SVG Scalable Vector Graphics
TTP Tactics, Techniques and Procedures
TTX Table-top Exercise
UI User Interface

Addendum

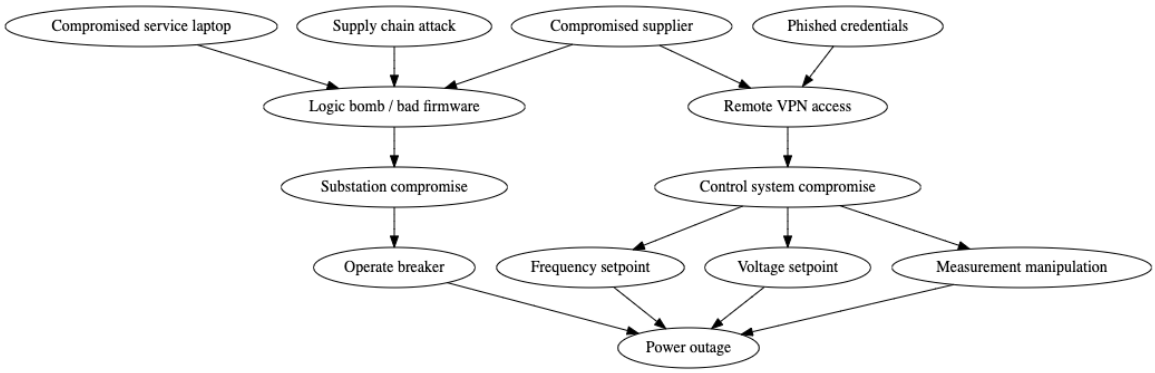


Figure 8.1: Exemplary simplified attack graph

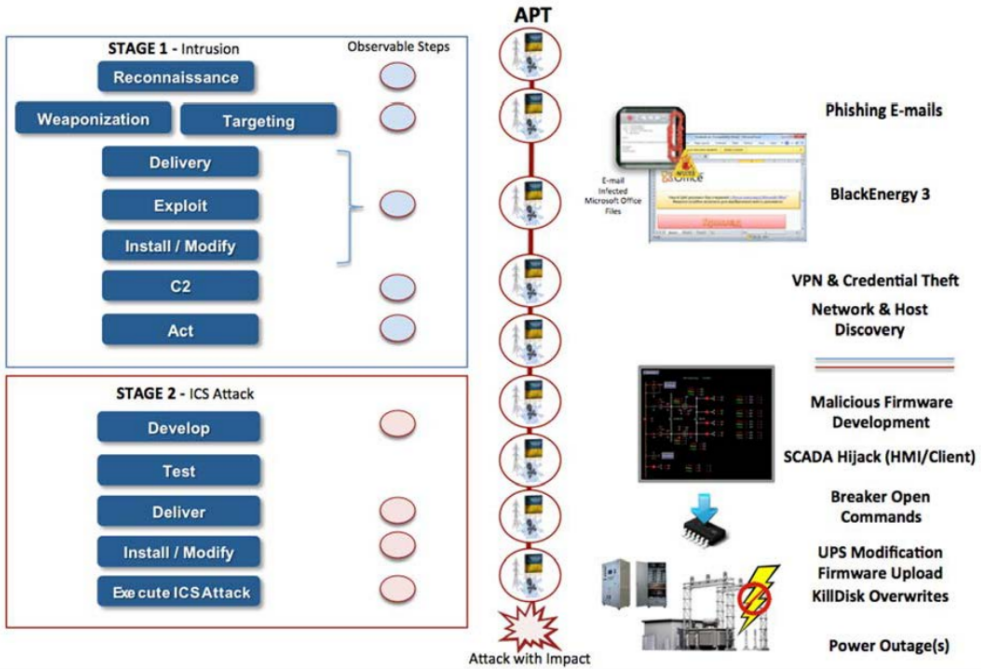


Figure 8.2: Reconstructed events leading to the power outage in the Ukraine in 2015 [56]

User Interface Screenshots

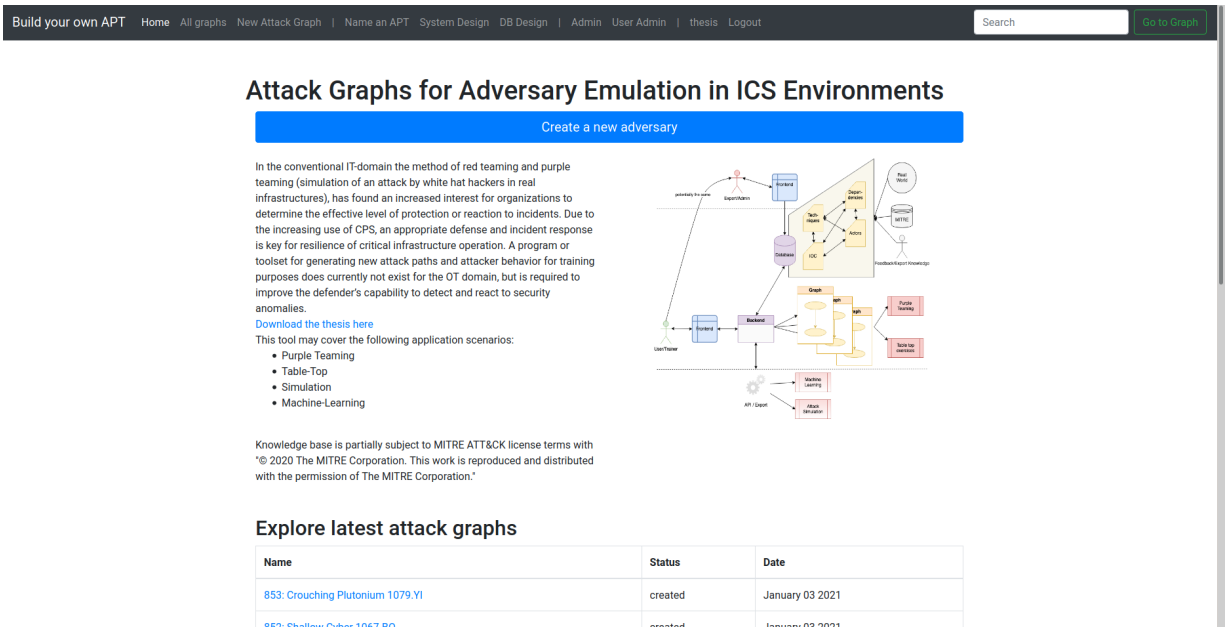


Figure 8.3: User Interface

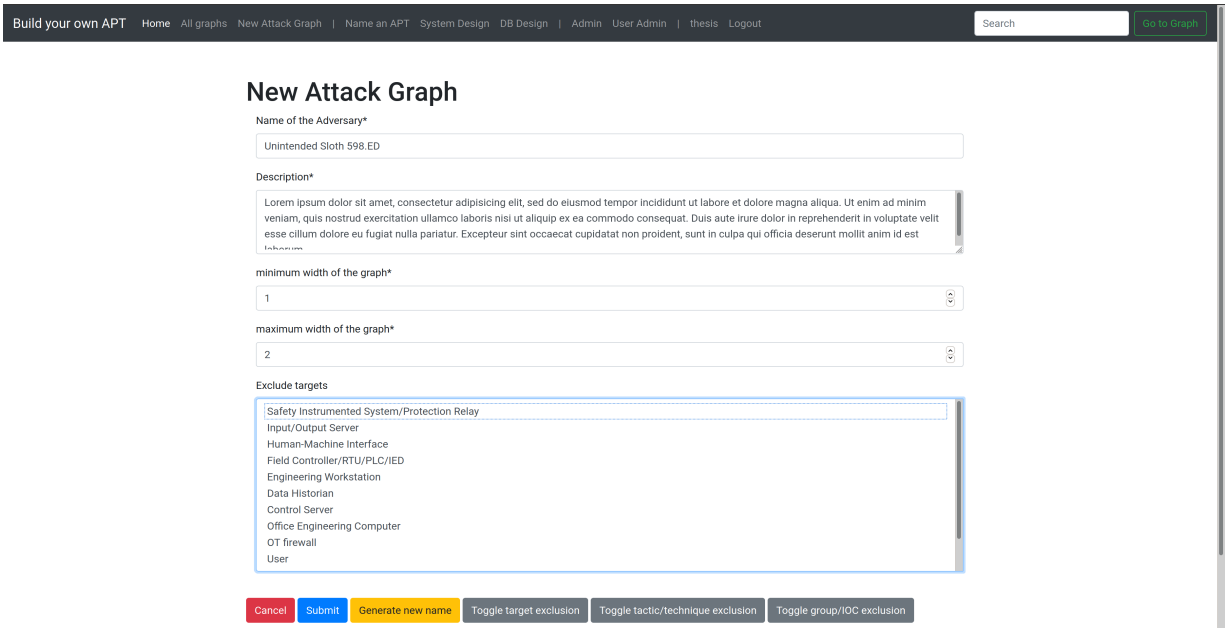


Figure 8.4: User Interface

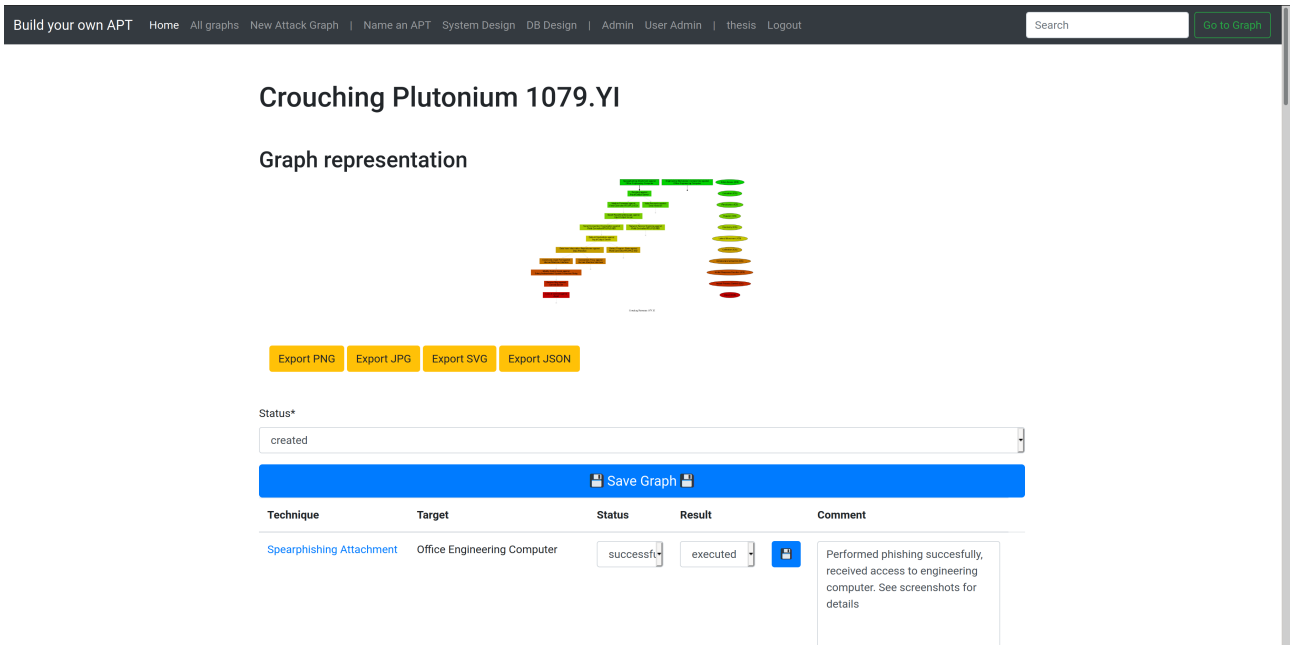


Figure 8.5: User Interface

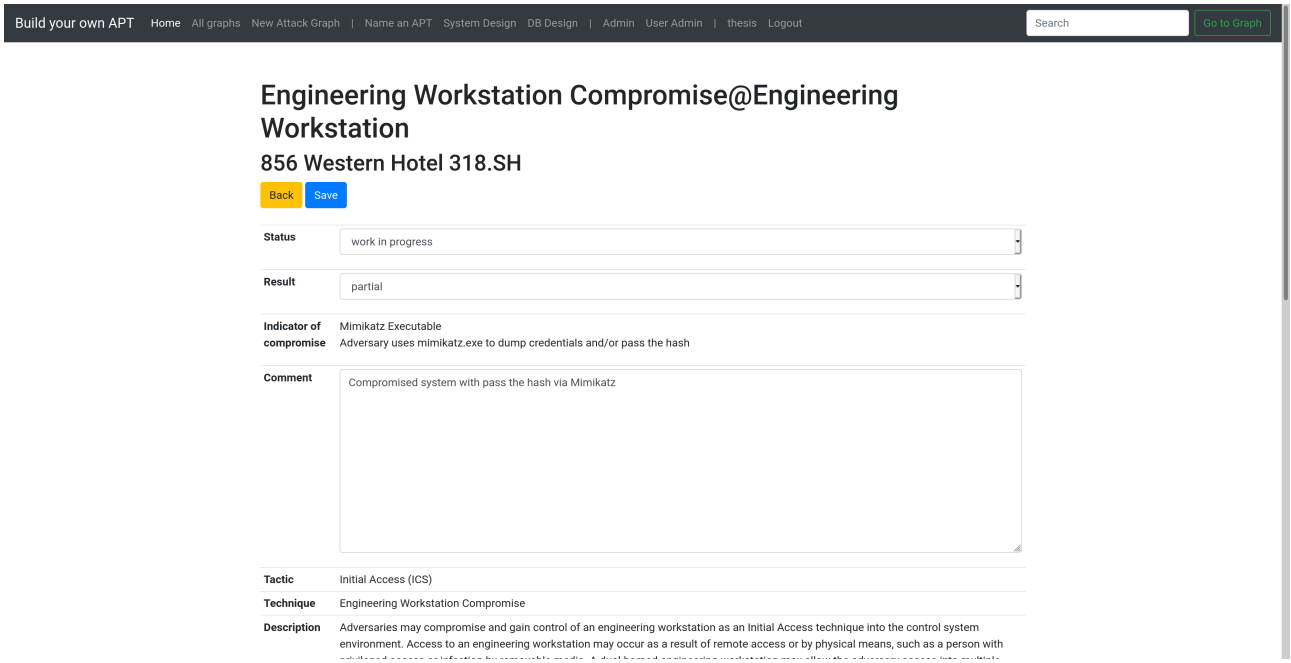


Figure 8.6: User Interface

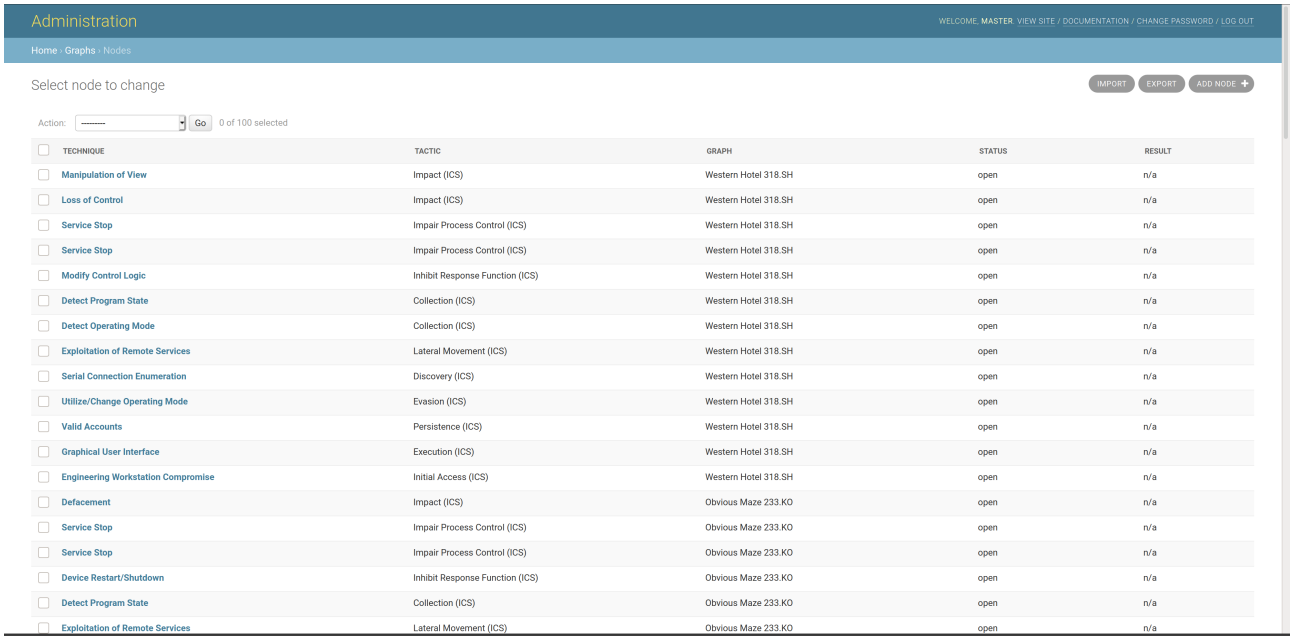


Figure 8.7: Administrative User Interface (Technique Overview)

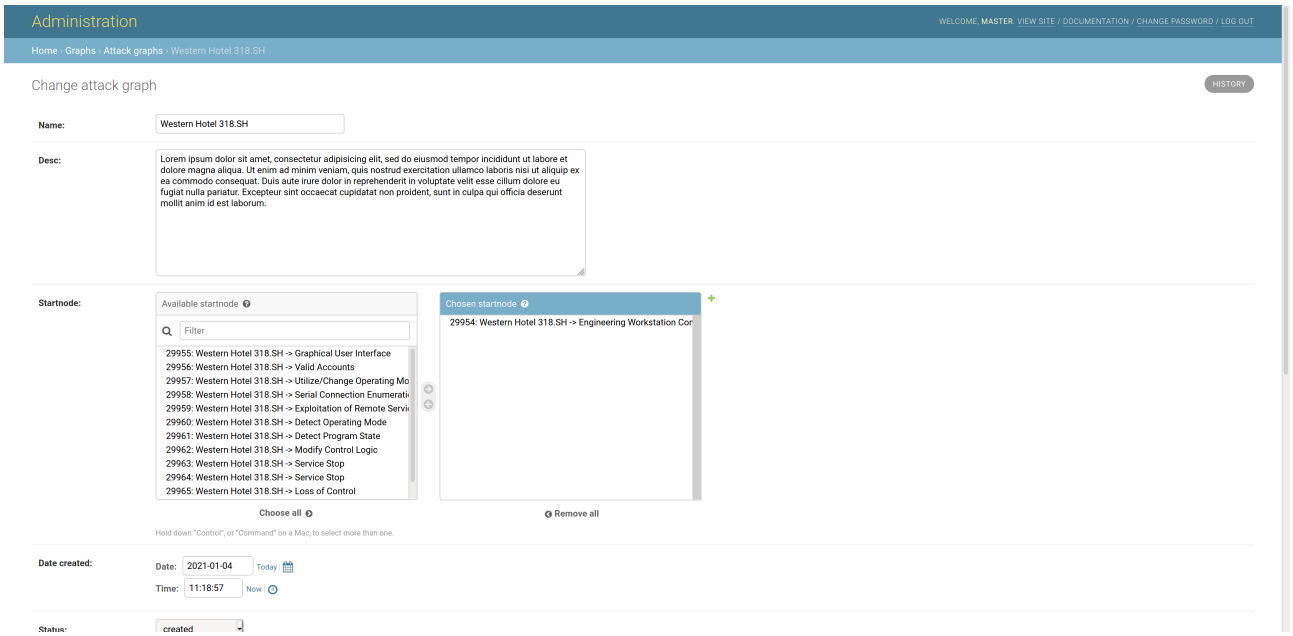


Figure 8.8: Administrative User Interface (Graph Editing)

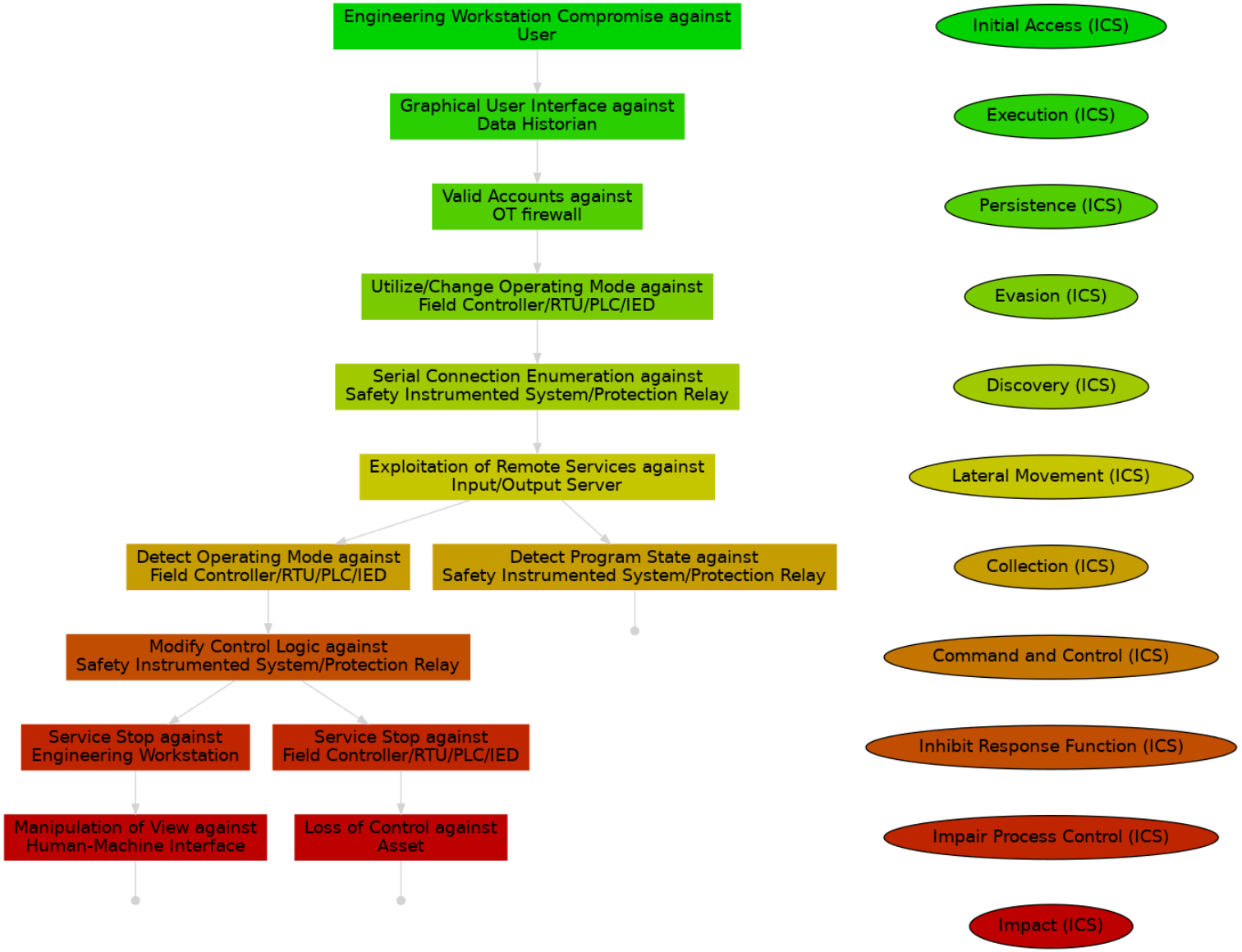
Example Graphs

Shallow Cyber 1067.BO

Figure 8.9: Example graph (randomized)

Obvious Sloth 75.ZU

Figure 8.10: Example graph (constrained)



Western Hotel 318.SH

Figure 8.11: Example graph (Sandworm-XENOTIME constraints)

Questionnaire

Research question:
Is it possible – and if yes to what extent – to algorithmically generate attack graphs that can be used for practical adversary behavior execution in ICS environments and can the process be supported by a corresponding application?

General context questions

1. Which organization are you related to (corporate/academia/government/other)?
2. In how far are you in contact with ICS systems?
3. How far did you have contact with adversary **emulation** or adversary **simulation**?
4. How have you used the ATT&CK framework previously?

Adversary behavior execution (ICS specific, else general enterprise context)

5. Are you or your organization already doing Purple Teaming or adversary emulation? If not are, you considering? What would be goals of adversary emulation in your opinion or experience?
6. If you have contact with ICS environment, have you considered active exercises like Purple Teaming in your ICS environments? If not, why?
7. How have you used attack graphs in the past? If so, did you model adversary behavior (TTP) in that context?
8. What requirements (functional and non-functional) would you see for a tool regarding graph generation for adversary behaviour modeling with attack graphs?

Graph walk-through (Interviewee is guided through the application)

9. Given the following graphs, would you consider those a valid attack chains in ICS (considering a generic ICS landscape)?

Application walk-through (Interviewee is guided through the application)

10. How far is the protype use case, application flow and the context clear?
11. Is there any room for improvement, when it comes the application and workflow?

Application in the context

12. Where else do you see potential use cases for such a tool in your area of work?

Summary of interviews

redacted for published version