TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technology

Department of Software Science

Centre for Digital Forensics and Cyber Security

ITC70LT

Terézia Mézešová 144708IVCM

ATTACK PATH DIFFICULTY – AN ATTACK GRAPH-BASED SECURITY METRIC

Master thesis

Supervisor

Dr. Hayretdin Bahsi

Tallinn 2017

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Terézia Mézešová

02.01.2017

Abstract

Information security decision-makers use attack graphs to model exploitable hosts and attack paths in their organizations' networks and systems. Attributes of the attack graphs, such as number and length of the paths, likelihood of exploitation and others, are measured by security metrics. Such security metrics evaluate the security level of the network and can be used one of the sources for risk assessment, threat analysis, or comparison of network hardening measures.

We present a security metric based on difficulty of an attack path. Difficulty is assessed from underlying vulnerabilities in the attack paths and their CVSS scores. We filter out the Impact metrics group and only consider CVSS metrics, which point towards the difficulty of exploitation.

We conduct a survey among cyber security experts to categorize attackers with regard to their technical skills and map the attacker categories to individual values for of considered CVSS metrics.

This thesis is written in English and is 48 pages long, including 7 chapters, 3 figures and 11 tables.

Annotatsioon

Ründevektori keerukus — ründeskeemil põhinevad turvalisuse näitajad

Infoturbe otsusetegijad kasutavad ründeskeeme, et luua oma organisatsioonide võrkudes ja süsteemides ärakasutatavaid hoste ning ründevektoreid. Infoturve hindab ründeskeemi atribuute, näiteks ründevektorite hulka ja pikkust, ärakasutamise tõenäosust jne. Selline infoturve hindab võrgu turvalisuse taset ja seda saab kasutada riskihindamise, ohu analüüsi või võrgu turvalisuse parandamise meetmete võrdluse allikana.

Meie infoturve põhineb ründevektori keerukusel. Keerukust hinnatakse ründevektorite nõrkade kohtade ja nende CVSS hinnangute põhjal. Me filtreerime välja mõjunäitajad ja võtame arvesse ainult neid CVSS näitajaid, mis viitavad ärakasutamise keerukusele.

Me teostame uuringut küberturvalisuse ekspertide seas, et kategoriseerida ründajaid nende tehniliste oskuste järgi ja kaardistada ründajate kategooriaid kaalutletud CVSS näitajate individuaalsete väärtuste jaoks.

Lõputöö on kirjutatud inglise keeles. See on 48 lehekülje pikkune, sisaldab 7 peatükki, 3 joonist ja 11 tabelit.

Table of abbreviations and terms

NVD	National Vulnerability Database
CSIRT	Computer Security Incident Response/Readiness Team
CERT	Computer Emergency Response Team
CVSS	Common Vulnerability Scoring System
AV	Attack Vector
AC	Attack Complexity / Access Complexity
PR	Privilege Required
UI	User Interaction
E	Exploit Code Maturity
RC	Report Confidence
Au	Authentication

Table of contents

List of figures

List of tables

1. Introduction

Cyber security has become a highly publicized topic in the last few years due to a number of unprecedented cyber attacks such as hacks on the SWIFT financial transactions software, disclosure of two-year-old breach of 500 million user accounts and passwords at Yahoo, accusations that the US presidential elections have been hacked, and many more [1]. Governments and organizations have started to take a closer look at their cyber security practices, preparedness of their employees to handle an incident, or the state of their networks. Cyber exercises serve this purpose and are developed to enhance relevant actors' decision making capabilities, or technical personnel's knowledge of attack and defense techniques.

Designing technical cyber exercises has a unique set of challenges such as creating a scenario and the underlying network infrastructure to support the goals of the exercise and provide tools for the participants. Organizers lack systematic tools to design diverse set of challenges with different difficulty levels for both offending and defending sides. We propose a security metric to rate paths in an attack graph according to technical difficulty required from an attacker. Our security metric improves measuring side of cyber exercises. We provide designers with a tool to evaluate their game network from the systematic data before the exercise and during the participants can be awarded points according to how difficult were the paths they have successfully exploited.

Our difficulty metric expresses factors such as position of an attacker with regard to the target network, possibility to use automated tools, or need to create own exploit code. These relate to technical or programming skills an attacker has to have to successfully exploit a path.

Cyber security professionals use attack graphs to model and analyze security properties of computer networks in organizations. They show possible paths of exploitation an attacker might take to gain access to an asset. Attack graph-based security metrics were developed for analyzing network conditions and vulnerabilities occurring

in a system to help decision makers evaluate and prioritize risk management, patch managements, network hardening, or asset protection measures.

Our security metric quantifies difficulty of the steps attacker makes to get to an asset and is determined from CVSS score about exploitable vulnerabilities in the network. CVSS is a standardized scoring system for vulnerabilities and contains information on conditions needed for successful exploitation and the weight of impact on the asset.

First, categories of attackers are created by using expert knowledge in a survey. For that, exploitability variables are filtered and their values assigned to categories of attackers. Then, CVSS metrics values are assigned to these categories, which further align the attacker categorization. Difficulty rating can be expressed according to the attacker category or numerically. CVSS score calculation is modified to reflect only relevant variables to get numerical difficulty rating value. These CVSS derivations are used for the attack path difficulty security metric.

Related works on attack graphs and CVSS based security metrics are discussed in section 2. Section 3 describes the methodology of conducted survey and results are presented in the attacker categorization model in section 4. Difficulty rating for attack paths and an example are presented in sections 5 and 6.

1.1 Problem statement

Participants in a technical cyber exercises expect to tackle various types of challenges. The designers try to include challenges of similar difficulty level across all possible intrusion scenarios or sets of similarly difficult scenarios. This depends on the profile of the participants and if the exercise is targeted towards the defending or offending team. Designers try to apply difficulty levels by putting flags, but those are not systematic.

Scope of the thesis is to find most appropriate formal model to represent the intrusion scenarios in a cyber exercise and create a systematic comparison of intrusion paths by their difficulty. This includes determination of measurable properties in the chosen model that will yield the difficulty level and how to express the difficulty itself.

Risk management defines threat agent as someone with motivation, resources and skills [2]. Cyber exercises usually do not span more than a couple of days, including time the teams get to familirize themselved with the scenario, network infrastructure, and exercise rules. The attacking Red Team members are constrained on time and money they can invest into the attacks. This should be taken into consideration by the exercise designers. Purpose of the exercise is to challenge the skills of the participants, therefore it is safe to assume they are all equally motivated and that difficulty of a scenario can be assessed based on how challenging it is for the participants' skills by determing lower boundary on required skills for the exploitation, e. g. if they only need to search for an already existing exploit code, if they need to modify it, or create their own from scratch.

Possible intrustion scenarios must be known to the designers. Attack graphs are a formal abstract model for representation of all possible paths an attacker may use for an intrusion. They model network connectivity between hosts in a network, running services and vulnerabilities present in the hosts and provide a base for evaluating various properties of the network.

Intrusion scenarios are reprented as attack paths, which together constitute an attack graph. Difficulty of individual paths can differ. Attack path contains information about which hosts need to exploited, in which order and what vulnerabilities are in them. Path difficulty is a function of corresponding difficulties of vulnerabilities occuring in the path.

When a vulnerability is publicly disclosed, vendors describe the conditions under which it is exploitable and how severe it is in terms of impact on the confidentiality, integrity, and availability of data in an organization's network. Common Vulnerability Scoring System (CVSS) standardizes this into groups of metrics and over 79 000 vulnerabilities have their CVSS score calculated [3]. Individual CVSS metrics aim to describe the conditions under which a vulnerability is exploitable and the impact of exploitation on an asset. The exploitability metrics can be used to measure difficulty of an individual vulnerability, because they can be correlated to technical skills.

The different levels of difficulty yield a classification of attackers into appropriate levels as well. There are several existing cyber threat models and categorization systems. Cyber security professionals were approached and expert knowledge was collected on which categories of attackers make sense in practice. There are two approaches on expressing the difficulty: ordinal scale (by a category of an attacker), and by a numerical value (determined from CVSS). These can be combined when attacker categories correspond to CVSS metric values intervals.

1.2 Application areas

We evaluate difficulty of an attack path by using currently best systematic data set in cyber security (vulnerability and exploit databases, and common vulnerability scoring system). Our metric can quantify the difficulty of each step in the path.

Calculating attackers' capabilities contributes to risk assessments in assessing the threat agents. Organizations can adjust their threat portfolios according to difficulty analysis, analyze changes in their threat profiles when introducing new assets or exchanging underlying technical solutions, design custom technical cyber exercises for continuous training of their employees or evaluating their technical skills.

Cyber exercises

Main application area is technical cyber exercises. Organizers of technical cyber exercises have to provide attack paths in a fictional organization's systems that are both realistic, and equally challenging for participants. For exercises where Red Teams are scored, it is of interest for the game organizers to be aware of how difficult their paths are to fairly award points for reached targets. The difficulty-based attack path rating is able to rate and rank attack paths based on CVSS scores of vulnerabilities occurring in the system.

The exercise designers create an attack graph of the intended computer network. The attack graph contains information about what kind of programs and services are running on the hosts in the network and the vulnerabilities in those hosts. Attack graph consists of a number of paths, each representing a different intrusion scenario. They may have a common start and varying goals, or two or more paths may lead to the same goal. Many attack graph representations suffer from a state explosion problem, meaning there is exponentially a lot more nodes in the graph than hosts

in the represented network. Our path difficulty metric determines the difficulty from information about vulnerabilities, and therefore the attack paths can be downscaled to only those nodes representing a presence of vulnerability.

Upon calculating the difficulty of each path, the designers are able to assess if the result is satisfactory or if any paths need to be adjusted to be more suitable for the exercise. The numerical value of the path difficulty rating can constitute awarded points when Red Team reaches the path goal.

Cyber threat intelligence

Cyber threat intelligence aims to provide organizations with knowledge to better protect their infrastructures against cyber attacks. Samtani et al. review cyber threat intelligence portals in [4]. Portals providing threat intelligence collect attack data from honeynets, intrusion detection/prevention systems and malware. They identified a main system gap in using only data sources from already captured attack data. Security decision makers in organizations can use attack graphs and the difficulty metric as additional source for their threat intelligence portals. They can be presented with possible attack paths within their system, the difficulty of those paths and design and evaluate their countermeasures accordingly.

2. Background information and related work

This section summarizes the key definitions of an attack graph and its types, and provides a literature review of attacker categorizations.

2.1 Attack graphs

An **attack graph** is defined in [5] as representation of „system states using a collection of security-related conditions, such as existence of vulnerability on a particular host or a connectivity between different hosts." It is an abstract representation of paths an attacker may use to compromise the security of a system [6]. Exploitable vulnerabilities and their causal relations in the network are modeled.

Yi et al. compared and analyzed attack graph generation technology in [7]. They compared the tools from their scalability and complexity aspects. Analyzed tools were open source such as MulVAL, TVA, Attack Graph Toolkit, and commercial tools such as Cauldron, FireMon, Skybox View. Based on their analysis this thesis uses **MulVAL** to generate attack graphs because it is an open source and still accessible tool for Linux systems. It has an easy to use command line interface, although proper installation and initial setup require significant effort.

MulVAL [8] is capable of working with Nessus or OVAL scanning reports, however the NVD feeds [9] component is outdated as of time of this thesis' publication. It is still possible to generate attack graphs by manually writing the network configuration and vulnerability information into the input file. MulVAL uses a reasoning engine to produce a logical attack graph, described below.

Alhomidi and Reed provided an extensive overview of attack graph representations in [10]. They identified over 10 types of attack graphs and for each type describe their nodes and edges representation and its construction or scalability.

This thesis uses a **logical attack graph**, which is based on the causality relations between system configurations and the attacker's potential privileges. It is a goal-

oriented dependency attack graph. Logical attack graph was chosen because it can be easily expressed in a text form and is supported by the attack graph generation tool MulVAL.
All attacks are expressed in a propositional formula in terms of network configuration parameters [10]. It is possible to obtain all attack scenarios using depth-first search. Two logical relations are represented – an OR condition enabled by any of its incoming neighbors; and an AND condition requiring all its incoming neighbors to be true.

An excerpt of example attack graph of a small network with a webserver running on two ports, is given in Figure 1. It is a MulVAL-generated and visualized attack graph. AND-conditioned nodes are in elliptic shape, OR-conditioned in diamond shape and leaf nodes are in rectangle shape.

Determining unique paths in the graph is possible via depth-first search from a chosen goal. An AND-conditioned node includes all its incoming neighbors in a current path, while an OR-conditioned node stems as many new paths, as the number of its incoming neighbors.

Note how in Figure 1, there are at least two possible paths to reach the goal denoted by node 6: via node 4 indicated in blue or via node 5 indicated in red. Blue path is a sequence of nodes $P_b = (1 - 2,4,6)$. Nodes 1 and 2 are conditions for AND-node 4, which is why they are written via dash in the path, to indicate they are on the same level and both need to be true to pass through node 4. In red path, nodes 1 and 3 must be true to pass through node 5. Red path is the sequence $P_r = (1 - 3, 5, 6)$.

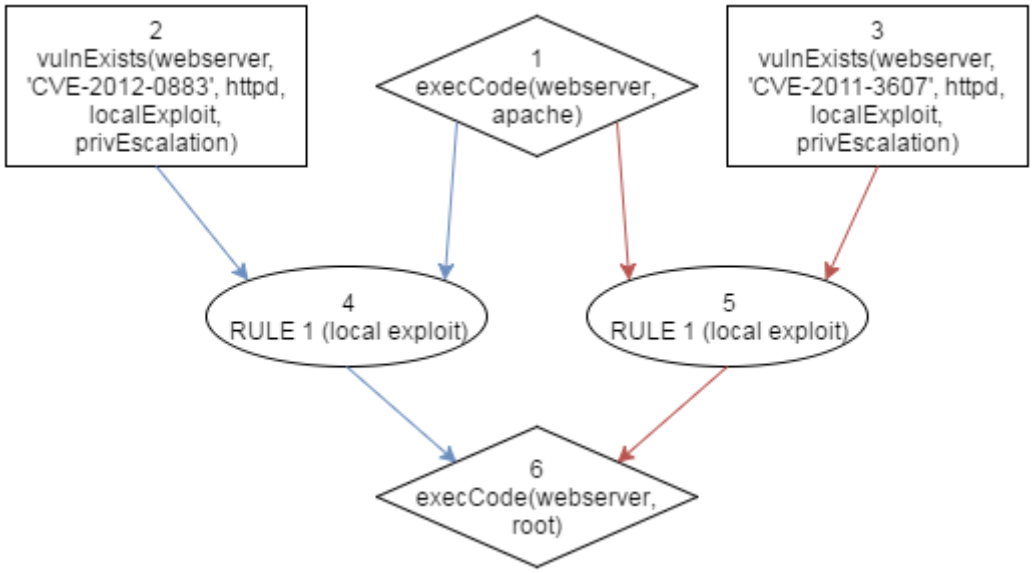

Figure 1. Excerpt example attack graph

Security-relevant properties about a network can be revealed by analyzing an attack graph. Idika in their dissertation [11] provide an approach for aggregating the capabilities of existing attack graph-based security metrics. **Attack graph based security metric** is a value produced from measuring internal attributes of a network [11]. Such attributes might for example be average length of a path in the attack graph, number of paths, probability of exploitation per path, etc. [11]. The value is derived by applying an analysis over an attack graph. Using these values separated or aggregated provides security analysts with a comprehensive description of generated attack graphs to help risk or patch management.

Attack graphs provide security analysts with information about vulnerabilities on network hosts. Differences between the vulnerabilities are an important factor while assessing the risk, thus Zhang et al. in [12] propose a metric where "vulnerabilities belonging to the same application are grouped into one node representing a successful exploitation of any of them." They argue that a host with ten vulnerabilities in one application has a lower risk than a host with one vulnerability in ten applications. Their model computes a numeric value representing the cumulative likelihood for an attacker to succeed for individual

machines [12]. They compute likelihood of exploitation and do not correlate it with how difficult it is to exploit a path with regard to the technical skills of an attacker.

Abraham and Nair present in [6] a stochastic security metric that takes into account vulnerability age. The model understands the relationship between vulnerability and its lifecycle events (discovery, exploit code availability, official patch availability) and is applied to identify vulnerability trends and anticipate security gaps.

2.2 CVSS

Common Vulnerability Scoring System (CVSS) [13] is a standard for assigning a numerical score to vulnerabilities based on their relative severity. It is represented by a vector string of key-value pairs and it is calculated to a value between 0 and 10, where the higher the value, the more critical the vulnerability is, due to ease of exploitation or criticality of the impact.

The key-value pairs in vector string are represented by acronyms for particular metric names and values, for example, vector string "AV: N/AC: L" means the vulnerability has score Attack Vector : Network and Access Complexity: Low.

CVSS provides standardized vulnerability scores, open framework and enables prioritized risk.

It comprises of three metric groups:

1. Base
2. Temporal
3. Environmental

Metrics in the Base group are mandatory and yield a base score. Temporal and Environmental scores further increase or decrease it. Analysts or vendors upon publication of vulnerability calculate Base and Temporal scores.

Base group metrics characterize exploitability and impact on an asset. Exploitability subgroup reflects the ease and technical means of the exploit – they are the characteristics of the thing that is vulnerable. Impact subgroup reflects the direct consequences on the asset that suffers.

Temporal group reflects characteristics that may change over time, but not across environments. It is possible to adjust it for better reflection of organization's environment.
Environmental group provides metrics to allow end-user organizations recalculate the vulnerability score with regard to their own computing environment.

Cheng et al. [14] interpret individual CVSS scores to preserve the semantics and dependency relationships between the vulnerabilities. They introduce effective base score, which takes an adjusted value defined over any ancestor of an exploit. They compare their approach over attack graphs and Bayesian network models. One of discussed aspects is that difference between scores may imply a minimum skill of an attacker to exploit vulnerability.

Keramati et al. [15] propose a CVSS-based metric to assess the impact of attack occurring on the network. It also considers the relationships among the vulnerabilities in the attack graph. The aim is to measure the amount of gained improvement of hardening measures. Their assumption is that there is only one attacker in the network. This thesis builds on the remark of [15] that Temporal score is an indicator of how difficult it is to successfully exploit vulnerability and that attack paths with longer lengths require more effort from an attacker.

Apart from CVSS, NCSC-NL CERT team developed a TARANIS risk assessment tool for creating useful alerts to their constituents and other CERTs [16]. Their objection towards CVSS is that is obscures the impact and probability scores. They separate the CVSS Base and Temporal metrics, which refer to impact and probability and score them in 2 matrices by means of answering questions. This gives the reader an immediate idea of both, although it is possible to do so with CVSS scores as well. Apart from CVSS, their chosen questions reflect approaches of US-CERT, SANS Internet Storm Center and Microsoft.

2.3 Attacker categorization

Raymond [17] presented traffic analysis attacks over common network protocols. He considered attributes for the attackers: *internal* or *external*; *passive* or *active*; *static* or *adaptive*. An attacker can have any combination of these. This classification is sufficient for analyzing security properties of network communication protocols and a solid base for determining severity of threat.

Fonash [18] identified capabilities within cyber security ecosystem and categorized cyber attacks methodologies and the capabilities needed to both perform the attack and to strengthen the resilience to them. Technically speaking, the relevant methodologies are *attrition attacks, malware, hacking, social engineering attacks*, and *multiple component attacks*.

Meyers et al. [19] proposed taxonomy of cyber adversaries based on the method of engagement and classified the attacks. It reflects well the popular terms used in cyber security news sources such as *script kiddies, hacktivists, cyber punks, black hat* and *white hat hackers.* The provided skills, maliciousness and motivation for these categories allow an attacker to be categorized into more than just one. Meyers et al. do not correlate the given method of engagement for the adversary class with the attacks classification.

Salles-Loustau et al. [20] took different approach to measuring attackers' skills and analyzed their results on attack sessions recorded on a honeynet. They assess the skills with a list of criteria – actions describing an activity during an attack and the degree of competency an attacker showcases. Downside to this approach is that a manual assessment by digital forensic analysis is needed to get the degree of competency. Each criterion is scored with a value between 0 and 1 and overall skill is sum of those over observation period. The observed actions include abilities to *hide traces*, *check system state*, *edit configuration files*, and *assess adequacy of used software tools*.

The reviewed literature presents attacker categories determined from captured and analyzed attack data. This gives a systematic overview on different types of threat and initial attack vectors that is suitable for use in risk assessment. An attacker classification is missing, which focuses on the attackers' skills that could be correlated with available vulnerability information. During design phase of a cyber exercise, the designers only have system data available, as attack data are produced during the exercise.

3. Survey of experts

Attacker categories from the literature review differentiates among types of attackers based on their motivations and sponsorships. Types of attackers from these models are not mutually exclusive, e. g. *cyber punks* and *script kiddies* in [19] are presented as 2 distinctive attacker types, yet from the technical skill perspective, *cyber punks* can cover all types of attackers. Researched models are not granular enough to appropriately cover differences between technical skills.

We conduct a survey among cyber security experts to establish an appropriate attacker categorization based on the technical skills of the attackers. The survey serves as a base to create an attacker categorization model focused solely on technical skills of attackers. Utilizing collective knowledge of experts to reach a conclusion on a specification is a common approach. Many working group with focus on technology specifications use similar approach of reviewing drafts semi-publicly within the expert network until a conclusion is reached and final version is published.

Survey lays ground to answer the research question of how to express the difficulty. It provides an attacker categorization model, usable in practice, and the difficulty of a vulnerability can be expressed with the appropriate category from the model.

We use modification of Delphi method survey in this thesis. It is described in [21] and [22] as an iterative process to collect judgments of experts using series of questionnaires focused on problems or solutions. Each questionnaire is developed based on the results of the previous one. The feedback is iterative and controlled, the iterations allow a conclusion to be reached.

The survey was conducted in two rounds. Different set of questions are used in the two rounds. The first round comprised of broadly set questions on identified issues among attacker capabilities from the related literature. The collected feedback was used to enhance specifically set questionnaire in the second round. Conclusion of the second round serves as the base of the attacker capabilities model presented in this thesis.

Participants' profile

Out of 8 experts that participated in both rounds of the survey, 2 have experience in CSIRT and 6 with penetration testing experience of more than 5 years. Penetration testers were chosen for their ability to assess ease of attack methods and overview of what can be achieved with available tools and where the attacker needs technical knowledge. CSIRTs collect information about security incidents and can provide a defender point of view on the skills in the attacker categorization.

3.1 First round

Initially, we proposed four categories of attackers:

- *automated tools*, as found online with no interaction from human element besides downloading and running
- *script kiddies*, as attackers who use only publicly available attacking tools and have minimal IT knowledge to configure them
- *moderately skilled attackers*
- *highly skilled attackers*

Determining the capabilities from attack graph is done from technical information about vulnerabilities. It provides a way to assess technical skills people have. The scope for the difficulty metric is related to skills only and not motivation and resources, therefore some types of attackers identified in literature review, such as white hat or black hat hackers are irrelevant as categories in our model. Script kiddy is a widely known term with no conclusive definition of their technical skills. This survey attempts to reach such definition.

Script kiddies and automated tools are separated in the first proposal, so that the lowest skill level category only represents attack tools functional without human interaction and that the next category are people, who use these tools, are able to configure them, but do not have technical background or understanding regarding their functionality and the used attack methods.

First round of survey focused on converging on the number of categories and their determining skills. Determining the skills based on the vulnerability information as described in section 5 brings to the surface the question of how to deal

with differences among the two used CVSS versions. Significant difference between the two versions is change from number of authentication gates as a metric to the type of privilege required for exploitation. We ask if number of authentication gates impacts the skill to be able to assess how vulnerabilities that only have CVSS version 2 score will impact the difficulty-based metric.

Attack paths may contain several vulnerabilities that must be exploited in the given sequence. If all these vulnerabilities are similarly easy to exploit, does the length of the sequence impact how easy it is to exploit the full path?

Following are the questions asked in the first round and some highlighted feedback from the experts:

1. How many categories of attackers would you consider? What would be the identifying skill in each category?
    a. 3 categories – (1) script kiddies with minimal IT knowledge using only automated tools and able to change only some parameters of the tools; (2) moderately/average skilled hackers with general deep understanding of IT and IT security with some academic knowledge, can exploit known bugs and know general attack techniques; (3) highly skilled hackers who can discover and leverage zero days and have long hands-on experience and know sophisticated attack techniques
    b. (1) fully automated attacks using botnets; (2) script kiddies of type "download & run"; (3) sophisticated attackers able to change parameters of an exploit, use Metasploit; (4) hackers with deep technical knowledge capable of writing exploits themselves
    c. 3 categories seem sufficient enough as even a script kiddy who downloads a very powerful automated tool can get quite far and look as a skilled attacker
    d. 3 categories – (1) script kiddies with minimal technical knowledge and ability to configure tools for the target, then (2) moderately skilled attackers who have some basic technical knowledge and (3) professional attackers such as penetration testers

2. Does the number of authentication gates impact the attacker's skill?
    a. to pass through multiple authentication gates it has to be at least moderately skilled attacker; similarly, getting access as a privileged user requires more capability
    b. passing number of authentication gates does not have an impact as it can very well happen that the credentials of the same user will work; obtaining credentials can be more of a goal and as a skill it could be assigned to all categories
    c. the number of authentication gates does not matter as many times there are single-sign-on functionalities and one pair of credentials can get an attacker quite far; similarly, with the type of the credentials, an attacker is either able to obtain them or not
    d. both matter and impact the skill - the authentication gates can be on different systems and therefore different technical knowledge to bypass them might be needed or completely different tools to get access as privileged user
3. Does number of vulnerabilities needed to be exploited to reach the attack goal impact the attacker's skill?
    a. number of vulnerabilities has an impact and is a key skill – to successfully combine them requires more skilled attacker
    b. script kiddies can have a higher chance of successfully getting to the attack goal if more vulnerabilities are present in the system, however more skilled attacker can have a chance of successful attack even if there are less vulnerabilities present
    c. the skill is impacted by the variety of areas those vulnerabilities are in – the more diverse the vulnerabilities are, the more skills the attacker needs
    d. number of steps could be an indicator as they may involve skills from different areas and persistence to reach the goal

Based on the answers, the **conclusion of first round** of the survey is following:

1. a model with 3 categories of attackers is sufficient enough; differentiating between automated tools and script kiddies as two low skilled categories does not provide substantial advantage; in the second round this model will be assumed:

a. script kiddies – have minimal technical knowledge and only use publicly available tools, automated or manual
b. moderately skilled attackers – have technical knowledge/education and know general known attack techniques
c. highly skilled attackers – have deep technical knowledge and experience and can discover new vulnerabilities and write functional exploit codes

2. the number of authentication gates slightly impacts the attacker's skill, as well as the type of credentials (whether standard/local user account has to be obtained or a privileged one)
3. the number of vulnerabilities impacts the attacker's skills because they might require different exploitation techniques

3.2 Second round

In the second round the experts were presented with two-part questionnaire. First part consisted of identified skills and a participant marked which attacker category shall be able to do it. For cross-referencing, a second part consisted of full sentence statements regarding an attacker category and a skill. These full sentence statements reflected the same skills and the category was our preliminary suggestion. They were to be marked as True or False, depending on whether a participant agreed or disagreed with the statement. This additional check that participant is actively thinking about their response is often used in psychological or social studies surveys.

The skills presented in both parts are the same and come from the descriptions of researched attacker categories and capabilities and CVSS metrics. In this round of the survey descriptions of filtered CVSS metric values were used, as well as additional capabilities identified through literature review. The participants were not told which those were.

Conclusion of the feedback from second round is presented in Table 1 and Table 2 below. In Table 1 the numbers represent how many participants marked the category for a particular skill. The numbers in Table 2 represent how many participants marked the statements as True or False.

Table 1. Results of part 1 of second round

	Skill **Can ___ ?**	**script kiddies**	**moderately skilled attackers**	**highly skilled attackers**
1	be distinguished as participant on the network	5	4	2
2	arbitrarily modify messages or status of the network	1	6	6
3	change resources they control once attack is in progress	1	6	6
4	use brute force methods to compromise a system	6	6	5
5	utilize malware to compromise a system	6	6	6
6	perform unauthorized intentional harm to a system	6	6	5
7	use deception or manipulation to obtain data or access to system	2	5	6
8	perform a single attack using multiple techniques	2	5	6
9	hide traces after intrusion	0	5	6
10	after attack restore files they deleted	0	2	5
11	check for presence of other users in the system	1	5	6
12	after attack delete files they downloaded	2	4	6
13	check system state before attack	1	4	5
14	modify attacking software to target specifics	2	6	6
15	modify target system before attack so attack can be performed	2	3	6
16	change password of compromised user	6	4	4
17	create dedicated user to keep access into a system	4	6	5
18	has knowledge of strengths/limitations of the attack software of choice	1	6	6
19	download and run already written scripts	6	6	5
20	engage in denial of service attacks	6	6	4
21	write own exploit code	0	1	6
22	perform attacks to brag about them	6	4	2
23	perform attacks for profit	2	4	5

24	exploit vulnerabilities which require physical access	1	3	6
25	exploit remotely exploitable vulnerabilities	6	6	6
26	invest some time and effort to execute an exploit	3	6	6
27	exploit vulnerabilities which require login to computer	3	6	6
28	obtain leaked credentials	5	6	6
29	obtain local user credentials	5	6	6
30	obtain privileged user credentials	2	6	6
31	exploit a vulnerability which requires an action by a different user	2	5	6
32	slightly modify code of functional exploit for target environment	0	6	6
33	configure a functional exploit for target environment	3	6	6
34	create own functional exploits	0	1	6
35	demonstrate proof-of-concept attacks	0	4	6
36	create exploit code based on proof of concept demonstration of attack	0	3	6
37	exploit confirmed vulnerabilities	4	6	6
38	exploit vulnerabilities which presence is indicated but not confirmed	0	2	6
39	reproduce proof of concept demonstration of attack on a different target	0	6	6
40	exploit multiple vulnerabilities in one attack	2	6	6
41	successfully pass multiple authentications of different types	1	4	6

Table 2. Results of part 2 of second round

	Statement	**True**	**False**
1	Script kiddies can be distinguished as participant on the network	6	2
2	Moderately skilled attackers can arbitrarily modify messages or status of the network	7	1
3	Moderately skilled attackers can change resources they control once attack is in progress	8	0
4	Script kiddies can use brute force methods to compromise a system	8	0
5	Script kiddies can utilize malware to compromise a system	7	1
6	Script kiddies can perform unauthorized intentional harm to a system	8	0
7	Script kiddies can use deception or manipulation to obtain data or access to system	4	4
8	Moderately skilled attackers can perform a single attack using multiple techniques	7	1
9	Script kiddies can hide traces after intrusion	0	7
10	Script kiddies can after attack restore files they deleted	0	4
11	Script kiddies can check for presence of other users in the system	4	4
12	Script kiddies can after attack delete files they downloaded	5	3
13	Script kiddies can check system state before attack	3	4
14	Script kiddies can modify attacking software configuration to target specifics	5	3
15	Script kiddies can modify attacking software source code to target specifics	0	8
16	Script kiddies can modify target system before attack so attack can be performed	2	6
17	Script kiddies can change password of compromised user	7	1
18	Script kiddies can create dedicated user to keep access into a system	6	2
19	Script kiddies have knowledge of strengths/limitations of the attack software of choice	1	7
20	Script kiddies can download and run already written scripts	8	0
21	Script kiddies can engage in denial of service attacks	8	0
22	Highly skilled attackers can write own exploit code	8	0
23	Script kiddies perform attacks to brag about them	8	0

24	Moderately skilled attackers perform attacks for profit	8	0
25	Highly skilled attackers can exploit vulnerabilities which require physical access	8	0
26	Script kiddies can exploit remotely exploitable vulnerabilities	8	0
27	Script kiddies are able to invest some (any kind is considered as some) time and effort to execute an exploit	7	2
28	Script kiddies can exploit vulnerabilities which require login to computer	6	2
29	Script kiddies can obtain local user credentials	6	2
30	Script kiddies can obtain leaked user credentials	7	1
31	Script kiddies can exploit a vulnerability which requires an action by a different user	4	4
32	Moderately skilled attackers can bypass multiple different types of authentication schemes in one attack	8	0
33	Script kiddies can modify functionality of exploit code	0	8
34	Script kiddies can configure a functional exploit for target environment	5	3
35	Highly skilled attackers can create own functional exploits	8	0
36	Highly skilled attackers can demonstrate proof-of-concept attacks	8	0
37	Moderately skilled attackers can create exploit code based on proof of concept demonstration of attack	4	4
38	Script kiddies can exploit only confirmed vulnerabilities	6	2
39	Moderately skilled attackers can exploit vulnerabilities which presence is indicated but not confirmed	3	4
40	Moderately skilled attackers can reproduce proof of concept demonstration of attack on a different target	7	1
41	Moderately skilled attackers can exploit multiple vulnerabilities in one attack	8	0
42	Moderately skilled attackers can successfully pass multiple authentication gates	8	0

3.3 Survey conclusion

Lack of publicly available datasets make it difficult to assess different categories of attackers and what are their similarities and differences. Expert survey provided a secondary research track to assess the correctness of initial analysis. The attacker categorization and capabilities model that came out of the survey is presented in the next section in detail.

The skills for which 5 or more people agreed on a particular skill category were accepted. After correlating the two parts of the survey the following are the most inconclusive questions:

- to what degree can script kiddies configure found functional exploit codes for the target environment
- whether moderately skilled means they can create their own exploit code based on proof of concept demonstrations or for vulnerabilities which have not been confirmed yet

For first – configuration of functional exploit codes by script kiddies – 5 people marked the statement True despite only 3 people assigning the skill to script kiddies. Therefore, the statement is considered true.

For second – moderately skilled attackers creating their own exploits based on proof of concepts – 4 people marked it as true and 4 as false and only 3 people assigned it for moderately skilled attackers, therefore this skill is **not** assigned to them.

4. Attacker categorization and capabilities model

Following three categories are settled as a result of the expert survey. Models that categorize people's skills have low, intermediate, and advanced levels. Our attacker categorization takes the same approach.

The values CVSS provides to determine severity of a vulnerability do not have a wide range and are not too granular themselves. Model with more categories would be too granular to map into the CVSS values.

Using the exploitability metrics from CVSS, the survey mapped the values as attacker skills and grouped them into the categories described in this section. Difficulty of a vulnerability will be expressed with an appropriate category.

4.1 Script kiddies

Expert survey results showed that having 2 low level of attacker categories is redundant, because the capabilities of the automated tools by themselves are limited. First category are **script kiddies**, representing low skilled attackers with basic IT knowledge and minimal technical or programming skills.

Their limited capabilities generally mean they only run the attack software they found online without any modifications. Survey participants agree that script kiddies are capable of being distinguished as participant on the network, using brute force methods and found malware samples or generators to compromise a system, engage in denial of service attacks. Their actions are intentional and will most likely brag about their attacks. They invest some time and effort to execute an exploit, which consists of downloading and running already written scripts (however, not by them). Therefore, they exploit only confirmed and remote vulnerabilities. Their basic technical knowledge makes them capable of configuring functional exploit for the target environment in terms of configuring the IP address or network range, ports or hostnames. Script kiddies can obtain credentials from various leaked databases and in many cases this provides access on a local user level.

4.2 Moderately skilled attackers

Moderately skilled attackers differ from script kiddies in not necessarily bragging about their exploits anymore, but are more likely to perform them for profit (monetary, educational or other). Apart from this, they have all other capabilities of script kiddies. They might have technical or engineering education and experience with general attack patterns and techniques. Capabilities that set them apart from script kiddies is, that unlike them, moderately skilled attackers can reproduce proof of concept attack demonstrations on different targets and modify code of a functional exploit for target environment. They can arbitrarily modify messages on or status of the network, change the resources they control during execution of an attack, even combine multiple components to perform an attack. Such components might be deception and social engineering tactics to obtain data or access to a system, or engage action of another user or even exploit more vulnerabilities in one attack.

Their skills include following from criteria defined in [20]: hide traces after intrusion, check presence of other users, delete files they downloaded after attack goal is reached, adequately choose attacking software/technique and create a user to keep access to the system. They are also able to change password of compromised user, however, this leads to discovery of the attack, which is undesirable.

They can exploit vulnerabilities which are not only remotely exploitable, but also those that require login to the system or computer and can obtain both local and privileged user account credentials and bypass multiple authentication gates to do so.

It was argued that due to wider usage of single sign-on features in many organizations, script kiddies shall have capability of bypassing multiple authentication gates. When single sign-on is enabled, one only has to pass through one authentication gate, so a script kiddy would only need one pair of credentials. If this authentication is two factor and contains a one-time password or code that is delivered to the user via another channel, this is a second authentication gate an attacker has to bypass. Therefore, multiple authentication gates are defined as subsequent authentications via different methods or on different platforms. Moderately skilled attackers are able to exploit vulnerabilities behind them.

4.3 Highly skilled attackers

Highly skilled attackers constitute the most skilled group of attackers in the presented model. Their most characteristic trait is deep technical knowledge as a whole and professional experience in particular domain areas. Professional penetration testers are within this category.

They have the skill set of moderately skilled attackers. One distinction is that they have technical knowledge to know how to hide and not be distinguished as a participant on the network. Therefore, from criteria in [20] they are capable of restoring files they deleted during an attack to hide the traces of intrusion.

Highly skilled attackers are characterized as those who can exploit vulnerabilities, which have not yet been publicly disclosed, or their presence is just indicated. They demonstrate proof of concept attacks and create functional exploit codes based on such proof of concept demonstrations, whether of others or their own. They can write a functional exploit code tailored for the target environment.

Physical access to a vulnerable component is not inherently a technical skill. Bypassing physical measures requires a completely different skill set which is out of scope for this thesis. For this reason, it was removed as a skill upon further discussion, despite initially having been included in the survey and marked for the highly skilled attackers' category.

Table 3 shows an overview of the skills assigned to the categories from the survey results.

Table 3. Attacker categorization and capabilities

	Skill **Can ___ ?**	**script kiddies**	**moderately skilled attackers**	**highly skilled attackers**
1	be distinguished as participant on the network	X	X	
2	arbitrarily modify messages or status of the network		X	X
3	change resources they control once attack is in progress		X	X
4	use brute force methods to compromise a system	X	X	X
5	utilize malware to compromise a system	X	X	X
6	perform unauthorized intentional harm to a system	X	X	X
7	use deception or manipulation to obtain data or access to system		X	X
8	perform a single attack using multiple techniques		X	X
9	hide traces after intrusion		X	X
10	after attack restore files they deleted			X
11	check for presence of other users in the system		X	X
12	after attack delete files they downloaded		X	X
13	check system state before attack		X	X
14	modify attacking software to target specifics		X	X
15	modify target system before attack so attack can be performed		X	X
16	change password of compromised user	X	X	X
17	create dedicated user to keep access into a system		X	X
18	has knowledge of strengths/limitations of the attack software of choice		X	X
19	download and run already written scripts	X	X	X
20	engage in denial of service attacks	X	X	X
21	write own exploit code			X
22	perform attacks to brag about them	X		

23	perform attacks for profit		X	X
24	exploit vulnerabilities which require physical access			X
25	exploit remotely exploitable vulnerabilities	X	X	X
26	invest some time and effort to execute an exploit	X	X	X
27	exploit vulnerabilities which require login to computer		X	X
28	obtain leaked credentials	X	X	X
29	obtain local user credentials	X	X	X
30	obtain privileged user credentials		X	X
31	exploit a vulnerability which requires an action by a different user		X	X
32	slightly modify code of functional exploit for target environment		X	X
33	configure a functional exploit for target environment	X	X	X
34	create own functional exploits			X
35	demonstrate proof-of-concept attacks			X
36	create exploit code based on proof of concept demonstration of attack			X
37	exploit confirmed vulnerabilities	X	X	X
38	exploit vulnerabilities which presence is indicated but not confirmed			X
39	reproduce proof of concept demonstration of attack on a different target		X	X
40	exploit multiple vulnerabilities in one attack		X	X
41	successfully pass multiple authentications of different types		X	X

5. Difficulty-based attack path rating

Having defined the categories of attackers, this section describes the relations between them and metrics from Common Vulnerability Scoring System.

The objective is to determine the minimal category level of an attacker based on the path information from the attack graph. Assume a situation that two paths of different difficulties exist between start point and goal nodes, then from design point of view, we can assume a skillful attacker can understand all the paths requiring difficulty level equal and lower to their level and follow the easiest path. Designer of cyber exercise should remove the paths between a particular start and goal nodes, which have lower levels of difficulty than wanted for the exercise.

5.1 Mapping attacker categories to CVSS metrics

Following section describes the relationship between particular CVSS metrics and the capabilities categories. The CVSS metrics, which do not have an impact on capability categories, are excluded as they were also excluded from the survey. A mapping table is then presented for each metric in the group. In the table, the lowest possible capability category is given, as the categories are subsets of each other.
Skills from round two of the survey are referred to by their order number as presented in Table 3.

5.1.1 Base score metrics

Attack Vector (AV) reflects context in which the vulnerability is exploitable. Script kiddies can exploit remotely exploitable vulnerabilities – with value Network. Adjacent vulnerabilities require access to local area network and Local are exploitable after user has logged in to a system. These are assigned to moderately skilled category as survey results suggest. As previously stated, getting physical access to site is not considered a technical skill, thus no category is mapped to vulnerabilities with value Physical. When Attack Vector metric is omitted in the calculation

of the difficulty score in this case, difficulty score can assess how hard it is to exploit the vulnerability from technical point of view as if the attacker had access to it. Then, decision makers will not be encouraged to assume that an occurring vulnerability requiring physical access does not have to be remediated, because it would be impossible to get to the vulnerable component.

Table 4. Attack vector metric

AV	**Description**	**Skill (as in Table 3)**	**Mapped category**
Network (N)	remotely exploitable via network	25	script kiddies
Adjacent (A)	exploitable from the same LAN	-	moderately skilled
Local (L)	not bound to the network stack; requires login to computer	27	
Physical (P)	requires physical access to the hardware	24	-

Attack Complexity (AC) describes conditions that are out of attacker's control that must be fulfilled for the vulnerability to be exploited. As per specification, no effort in preparation or execution from the attacker is needed if the value is Low. Any, even small, amount of effort will be scored as High. Script kiddies have to look up attack tools and their functionality online, which is considered an effort; therefore, they are assigned to High. This result shows that for all values, the mapped category is script kiddies. For this reason, this metric can be omitted from the calculation.

Table 5. Attack complexity metric

AC	**Description**	**Skill**	**Mapped category**
Low (L)	no special circumstances needed to exploit	-	script kiddies
High (H)	successful attack cannot be accomplished at will, but requires the attacker to invest in some amount of effort in preparation or execution	26	

Privileges Required (PR) describes the level of privileges the attacker must possess before successfully exploiting the vulnerability. While script kiddies can obtain any kind of leaked credentials, without such available dataset they can get access

as a local user – vulnerabilities which are scored Low. Subsequent escalations to get access as privileged user are mapped to moderately skilled attackers in the survey – metric value is High. It can be argued that script kiddies can get privileged access straightaway from the obtained leaked credentials, however in many organizations, privileged access is done via strong authentication methods, either via stronger password entropy or two factor authentication with one time passwords. This surpasses the script kiddy's definition of attacker who "downloads and runs" attack tools found online.

Table 6. Privileges required metric

PR	**Description**	**Skill**	**Mapped category**
None (N)	attacker is unauthorized and does not require any access to files to carry out an attack	-	script kiddies
Low (L)	attacker is authorized as a local user	29	
High (H)	attacker is authorized with privileges providing control of component-wide settings and files	30	moderately skilled

User Interaction (UI) reflects the requirement for interaction of a user other than the attacker for a successful exploit. CVSS specification does not provide details on the types of the other user's interactions for consideration. Survey participants were not conclusive for this particular skill. In the cross-referencing True/False statements half agreed that script kiddies can exploit such vulnerability with Required user interaction. In assigning part, however, only 2 marked the skill for script kiddies and 5 for moderately skilled attackers. Generally, an attacker needs to use deception to get a user to interact with the vulnerable component at appropriate time, and deception tactics are assigned to moderately skilled attackers, we assign this skill to them as well.

Table 7. User interaction metric

UI	**Description**	**Skill**	**Mapped category**
None (N)	vulnerability can be exploited without interaction from another user	-	script kiddies
Required (R)	some action taken by another user is required	31	moderately skilled

5.1.2 Temporal score metrics

Exploit Code Maturity (E) measures the likelihood of vulnerability being exploited. It is based on current state of attacking methods, available exploits and public availability of easy to use tools. Fully functional exploit code exists for vulnerabilities scored as High or Functional, may it be automated or manual attack tool. Using such exploit codes is the core definition of script kiddies as results of our survey show. When threat intelligence does not provide any available functional exploits, only Proof of concept demonstrations, attackers must write the code themselves. For this, they need deeper programming and platform specific knowledge, which is how moderately skilled attackers are defined. Survey results support this. Highly skilled attackers are able to demonstrate proof of concept attacks, implying they are able to create their own exploit code from scratch for vulnerabilities scored as Unproven.

Table 8. Exploit code maturity metric

E	**Description**	**Skill**	**Mapped category**
Not defined (X)	-	-	-
High (H)	functional autonomous code exists or details about the exploit are widely available and it works in every situation	33	script kiddies
Functional (F)	functional exploit code exists	32	
Proof of concept (P)	attack demonstration exists but is not practical for most systems; or the exploit code requires modification	32 36 39	moderately skilled
Unproven (U)	no exploit code is available or it is theoretical	34	highly skilled

Report Confidence (RC) measures the degree of confidence in the existence of the vulnerability and the credibility of the known technical details. For vulnerabilities scored as Confirmed, a functional reproduction is possible and available, which script kiddies can find and use. Reasonable report confidence means that explanation of an exploit is available, for example as a proof of concept demonstration. Moderately skilled attackers were assigned this capability in the survey. Unknown vulnerabilities

have not yet been publicly disclosed, but might be indicated by threat intelligence reports. Survey participants have assigned this to the highly skilled attackers.

Table 9. Report confidence metric

RC	**Description**	**Skill**	**Mapped category**
Not defined (X)	-	-	-
Confirmed (C)	source code of the vulnerable component is available to independently verify or vendor has confirmed the presence of the vulnerability and functional reproduction is possible	37	script kiddies
Reasonable (R)	there is reasonable confidence that reproduction of the vulnerability and explanation on how to do it is available	39	moderately skilled
Unknown (U)	there are reports of impact that indicate the presence, but they differ or nature of vulnerability is uncertain; it is not certain if a Base score can be applied	38	highly skilled

Other vulnerability attributes

Affected application or service and /or operating system are taken from the available description of the vulnerability and can be presented to the analyst as additional contextual information on the attacker capabilities in a given domain. An attacker might fall into different categories in different domains, analysis is not possible purely from an attack graph as forensic data of detected attacks is needed for such correlation.

Currently, the OS or application information is text only. In future work it is possible to incorporate such information into the path difficulty metric.

5.2 Difficulty of vulnerability

Difficulty of a vulnerability v reflects the minimal skills an attacker shall have in order to successfully exploit it. It is expressed as the minimal attacker category from the presented model. It is calculated like the CVSS score

$$Difficulty(v) = min\{exploitability(v), temporary(v)\} \quad (1)$$

where

$$exploitability(v) = 8.22 \times AttackVector_v \times AttackComplexity_v \times PrivilegeRequired_v \times UserInteraction_v \quad (2)$$

$$temporary(v) = exploitability(v) \times ExploitCodeMaturity_v \times ReportConfidence_v \quad (3)$$

Exploitability and *Temporary* are rounded up to two decimals. The constant *8.22* is taken from the CVSS specification [13] so the difficulty score reflects the CVSS score. Table 10 shows numerical values used for calculation.

Table 10. Metrics numerical values [13]

Metric name	**Ordinal CVSS metric value**	**Numerical metric value**
AV	Network (N)	0.85
	Adjacent (A)	0.62
	Local (L)	0.55
	Physical (P)	0.2
AC	Low (L)	0.77
	High (H)	0.44
PR	None (N)	0.85
	Low (L)	0.62
	High (H)	0.27
UI	None (N)	0.85
	Required (R)	0.62
E	Not defined (X)	1
	High (H)	1
	Functional (F)	0.97
	Proof of concept (P)	0.94
	Unproven (U)	0.91

RC	Not defined (X)	1
	Confirmed (C)	1
	Reasonable (R)	0.96
	Unknown (U)	0.92

5.2.1 Qualitative rating scale

CVSS assigns the highest score to the vulnerabilities, which are most likely to be exploited with automated tools or those which do not need any particular effort from the attacker before the exploit and have a big impact on the organization if exploited. The impact subgroup metrics were omitted from difficulty metric as they do not provide substantial information for scope of our analysis. Critical vulnerabilities are widely and easily exploitable, which for our difficulty rating yields the lowest attacker category level.

The rating scale is defined in Table 11. The scale for the categories is similar to the scale CVSS uses for assessing criticality of the vulnerability. In risk assessment the criticality is determined based on the likelihood and ease of exploitation. Assigning the critical and high risked vulnerabilities to script kiddies does not violate the established consistencies security analysts expect from security metrics. The border values for the qualitative scale were calculated from formulas (1) and (4) by substituting the values for each metric. The lower boundaries were expanded to cover the version 2 downgrade impact and because not all individual metric values have to be of the same category as described in section 5.2.3.

Expressing the difficulty value numerically means, that the lower the numerical value, the more difficult a path is to exploit.

Table 11. Qualitative rating scale

Difficulty score	**Attacker category**
2.51 – 3.88	script kiddies
1.01 – 2.50	moderately skilled
0 – 1.00	highly skilled

5.2.2 Impact of CVSS version downgrade

CVSS version 3 is used to calculate the difficulty. It was released in 2015 and introduced new metrics for more accurate characterization of vulnerabilities. Most of currently known vulnerabilities are scored according to version 2, although for newly disclosed vulnerabilities, version 3 score is provided, too and the versions are used in parallel. This section highlights the differences between the versions and how the calculation changes if only version 2 score is available for the vulnerability.

Attack Vector in version 3 reflects the Access Vector metric in version 2. The value *Access Vector: Local* was further divided into values Attack Vector: Local and Physical. It should be possible to determine from the vulnerability's description if physical access is required. In such case, we assign the values as if version 3 was being used. Otherwise it can be assumed physical access is not imperative and we and use value of Attack Vector: Local for the calculation.

Attack Complexity reflects the Access Complexity metric. The values *Access Complexity: Medium* and *High* were grouped into values Attack Complexity: High. We group these in the same manner and use for both values *Medium* and *High* use value High for calculation.

Privilege Required metric is a newly introduced metric. Related *Authentication* metric from version 2 was omitted from version 3. While the *Authentication* metric quantified the characteristic based on how many authentication gates the attacker had to pass for a successful exploit, the Privilege Required qualitatively describes the level of privilege in the OS the attacker must have prior to exploit. If it is possible to determine the type of needed privilege from vulnerability description, then we proceed as if version 3 was used otherwise it can be assumed that None is required.

Values *Authentication(Au): None(N)* and *Single(S)* and skills 28 and 29 from Table 3 can be correlated and those were assigned to script kiddies in the calculation. Numerical values are 0.7 and 0.56 respectively [23]. Survey results were conclusive

that moderately skilled attackers can bypass multiple authentication gates, so this category is mapped to value *Multiple(M)*, numerical value from [23] is 0.45.

User Interaction is a newly introduced metric and does not stem from any version 2 metrics. If it is possible to determine from vulnerability's description if user interaction is required, then we proceed as if version 3 is used. If it is not possible, it is assumed user interaction is not required and we continue with value None to the calculation.

Exploit Code Maturity in version 3 is a rename of the Exploitability metric in version 2. Their values are the same.

Report Confidence is the same in both versions. The values *Uncorroborated* and *Unconfirmed* were renamed to Reasonable and Unknown. The descriptions were rephrased for better understanding, but their scope remains the same and therefore calculation is done with version 3 values.

When vulnerability is only scored with CVSS version 2, difficulty formula (1) remains the same and formula (2) is modified to formula (4).

$$exploitability(v) = 8.22 \times AttackVector_v \times AttackComplexity_v \times PrivilegeRequired_v \times Authentication_v \times UserInteraction_v \quad (4)$$

5.2.3 Alternative difficulty calculation

If just expressing the attacker category for vulnerability is sufficient and numerical score not needed the determination can be as follows:

- if Temporary metrics are not available, take the most skilled category out of: Attack Vector, Attack Complexity (optional), Privilege Required, User Interaction, and Authentication (if version 2 is considered)
- if Temporary metric values are available for vulnerability, take the most skilled category out of: Attack Vector, Attack Complexity (optional), Privilege Required, User Interaction, Authentication (if version 2 is considered), Exploit Code Maturity, and Report Confidence

Overall difficulty of vulnerablity is then determined:

- vulnerability is assessed with script kiddie capability if all individual metric values are script kiddie
- vulnerability is assessed with moderately skilled if one or more metric values are mapped to moderately skilled attackers
- vulnerablity is assessed with high skilled if at least one metric value is mapped to highly skilled attackers

5.3 Difficulty rating of attack path

An attack path is a sequence of steps an attacker needs to take to get from their starting position, often from the outside of the organization's network, to the attack goal, a particular action on a particular asset.

It is assumed that the attacker is able to exploit all vulnerabilities and their preconditions along the attack path up to and including the attack goal. This thesis uses MulVAL [8] generated logical attack graphs. During traversal of a logical graph, an OR node distinguishes among the same amount of paths as it has outcoming edges. An AND node includes all its outcoming edges (each for the nodes part of the AND condition) into a path currently being traversed. We define a *supernode,* as an aggregation of all the AND-conditioned nodes into one.

A downscaled attack path is considered the nodes which do not represent vulnerabilities have been removed. This can be done because other nodes represent conditions

in the network, which must be explicitly stated for the attack graph engine to be able to generate an attack graph. Ideally, these are reflected in the vulnerability's CVSS score.

An OR node functions as path divider, therefore each OR-conditioned node is on a different path. All AND-conditioned nodes are a one step in the path. For the attacker to successfully proceed through the AND node, all its conditioned vulnerabilities $a_1, \dots, a_k$ must be exploited. Therefore, the difficulty score of the aggregated *supernode A* is the minimum of all difficulty scores:

$$Difficulty\{A\} = \min_{a \in A}\{Difficulty(a)\} \qquad (5)$$

After downscaling an attack path and aggregating the AND nodes, an attack path is left as a sequence of vulnerabilities, denoted as $P = (v_1, \dots, v_n)$.

Difficulty rating of path is minimum of all difficulty scores in the path.

It can happen that all vulnerabilities are scored with the same difficulty. Exploiting subsequent vulnerabilities after the first one, however, requires more skills from the attackers, than if they exploited those vulnerabilities individually. When we are presented with an attack path, we see the needed conditions that are preconditions for the next vulnerability in the path, but they do not necessarily have to be post conditions of the just exploited one. Attackers must assess these conditions themselves and if necessary, bring the system to needed state.

Variable value x is introduced to simulate additional effort of the attacker, when exploiting multiple vulnerabilities in path. Value assigned to x is arbitrary. However, it is recommended to keep it small, e.g. 0.05.

$$DifficultyRating\{P\} = \min_{v \in P}\begin{Bmatrix} Difficulty(v_1), \dots, \\ Difficulty(v_i) - (i-1)x, \dots, \\ Difficulty(v_n) - (n-1)x \end{Bmatrix} \qquad (6)$$

6. Attack graph analysis

We presented a model for defining difficulty of attack paths from metrics used in the CVSS. An example attack graph is presented in this section and the difficulty score of the paths is assessed.

An example attack graph provided by [24] is shown on Figure 2 and Figure 3. There is a web, mail, and citrix server, and workstations, divided into two subnets. There are six unique paths in the attack graph (downscaled paths only with vulnerabilities):

A. node 10 (CVE-2010-0490)
B. node 19 (CVE-2010-0483) → node 27 (CVE-2010-0494)
C. node 41 (CVE-2002-0392)
D. node 50 (CVE-2010-0483)
E. node 59 (CVE-2010-0490)
F. node 74 (CVE-2010-0812)

CVSS scores and information for the temporal scores are taken from National Vulnerability Database [9] and CVE Details [3], public vulnerability data sets which correlate CVE, OVAL and CWE definitions, reference affected versions and references for the vulnerabilities. Exploit-DB [25] is checked to assess what exploits are available for these vulnerabilities. Exploit Code Maturity (E) metric is either assessed as Proof of Concept (P) if no exploit code or Metasploit [26] modules are found. Values Functional or High if either of those exist. All these vulnerabilities have been confirmed by the vendors, so their Report Confidence (RC) score is Confirmed (C).

All vulnerabilities have only CVSS version 2 scores defined. The calculation is done with Formula (4). User Interaction (UI) is assessed from the description of the vulnerability. Presence of "user assisted" indicates user interaction is Required (R). Privilege Required (PR) is None (N) for all in this example because none require

authentication at all – Authentication (Au) is of value *None* (N). Attack Complexity is included in the calculation.

- *CVE-2010-0490*: AV: N, AC: M, Au: N, E: P, RC: C, PR: N, UI: N

$Difficulty_{CVE-2010-0490} = 1.46$ (moderately skilled attackers)

Alternative method of determination the difficulty yields:
Following categories are assigned to individual metrics (in order as the vector string above): AV: N – script kiddies, AC: M – script kiddies, Au: N – script kiddies, E: P – moderately skilled, RC: C – script kiddies, PR: N – script kiddies, UI: N – script kiddies → *moderately skilled attackers* shall be the overall category

- *CVE-2010-0483*: AV: N, AC: H, Au: N, E: F, RC: C, PR: N, UI: R
  One Metasploit module is found.

$Difficulty_{CVE-2010-0483} = 1.10$ (moderately skilled attackers)

Alternatively: AV: N – script kiddies, AC: H – script kiddies, Au: N – script kiddies, E: F – script kiddies, RC: C – script kiddies, PR: N – script kiddies, UI: R – moderately skilled → *moderately skilled attackers* shall be the overall category

- *CVE-2010-0494*: AV: N, AC: M, Au: N, E: P, RC: C, PR: N, UI: R

$Difficulty_{CVE-2010-0494} = 1.06$ (moderately skilled attackers)

Alternatively: AV: N – script kiddies, AC: M – script kiddies, Au: N – script kiddies, E: P – moderately skilled, RC: C – script kiddies, PR: N – script kiddies, UI: R – moderately skilled → *moderately skilled attackers* shall be the overall category

We can see that the numerical score for CVE-2010-0494 is lower compared to difficulty score for CVE-2010-0483, because there are two metrics which values are mapped to moderately skilled attackers – Exploit Code Maturity and User Interaction.

- *CVE-2002-0392*: AV: N, AC: M, Au: N, E: H, RC: C, PR: N, UI: R
  Three exploit codes are available.

$Difficulty_{CVE-2002-0392} = 1.13$ (moderately skilled attackers)

Alternatively: AV: N – script kiddies, AC: M – script kiddies, Au: N – script kiddies, E: H – script kiddies, RC: C – script kiddies, PR: N – script kiddies, UI: R – moderately skilled → *moderately skilled attackers* shall be the overall category

- *CVE-2010-0812*: AV: N, AC: L, Au: N, E: P, RC: C, PR: N, UI: N

$Difficulty_{CVE-2010-0812} = 2.55$ (script kiddies)

Alternatively: AV: N – script kiddies, AC: L – script kiddies, Au: N – script kiddies, E: P – moderately skilled, RC: C – script kiddies, PR: N – script kiddies, UI: N – script kiddies → *moderately skilled attackers* shall be the overall category

For the CVE-2010-0812, the disparity is visible even with the numerical score, which is on the lower boundary of script kiddies. This might be caused by imprecise determination of value for Exploit Code Maturity. There is a collision in the different vulnerability naming systems, so it is possible an exploit is mistakenly labeled. We will consider the **script kiddies for the path F** difficulty score.

Paths A, C, D, E, and F only consist of the singular vulnerability; therefore, the difficulty rating of the path is the same as the difficulty of the vulnerability. Path B consists of two vulnerabilities and its difficulty is calculated according to Formula (6).

A. $DifficultyRating(A) = 1.46$ (moderately skilled attackers)

B. $DifficultyRating(B) =$
   $min\ \{Difficulty_{CVE-2010-0483}, Difficulty_{CVE-2010-0494} - x\} =$
   $min\ \{1.10, 1.06 - 0.05\} = min\ \{1.10, 1.01\} = 1.01$
   (moderately skilled attackers)

C. $DifficultyRating(C) = 1.13$ (moderately skilled attackers)

D. $DifficultyRating(D) = 1.10$ (moderately skilled attackers)

E. $DifficultyRating(E) = 1.46$ (moderately skilled attackers)

F. $DifficultyRating(F) = 2.55$ (script kiddies)

We can see that path F is the easiest with score 2.55, exploitable by script kiddies; while all other paths are exploitable by moderately skilled. This is due to the required user interaction for many of the vulnerabilities in the system and this skill was assigned to moderately skilled attackers.

For the purpose of a cyber exercise it is clear that majority of the paths are within the capability of the same attacker category. Therefore, we suggest to remove the path F from the graph, or add additional step to the path to make it more difficult and shift it to the *moderately skilled category*, like the other paths.

The numerical values of the difficulty rating show that there are slight differences among the remaining paths A-E. In this example, they are all leading to different attack goals. Were some of them to have the same goal, the difficulty score expressed via numerical value would show the subtle differences among the paths.

Figure 2. Example attack graph, paths A-blue, B-green, D-orange.
Edited for clarity and length [24].

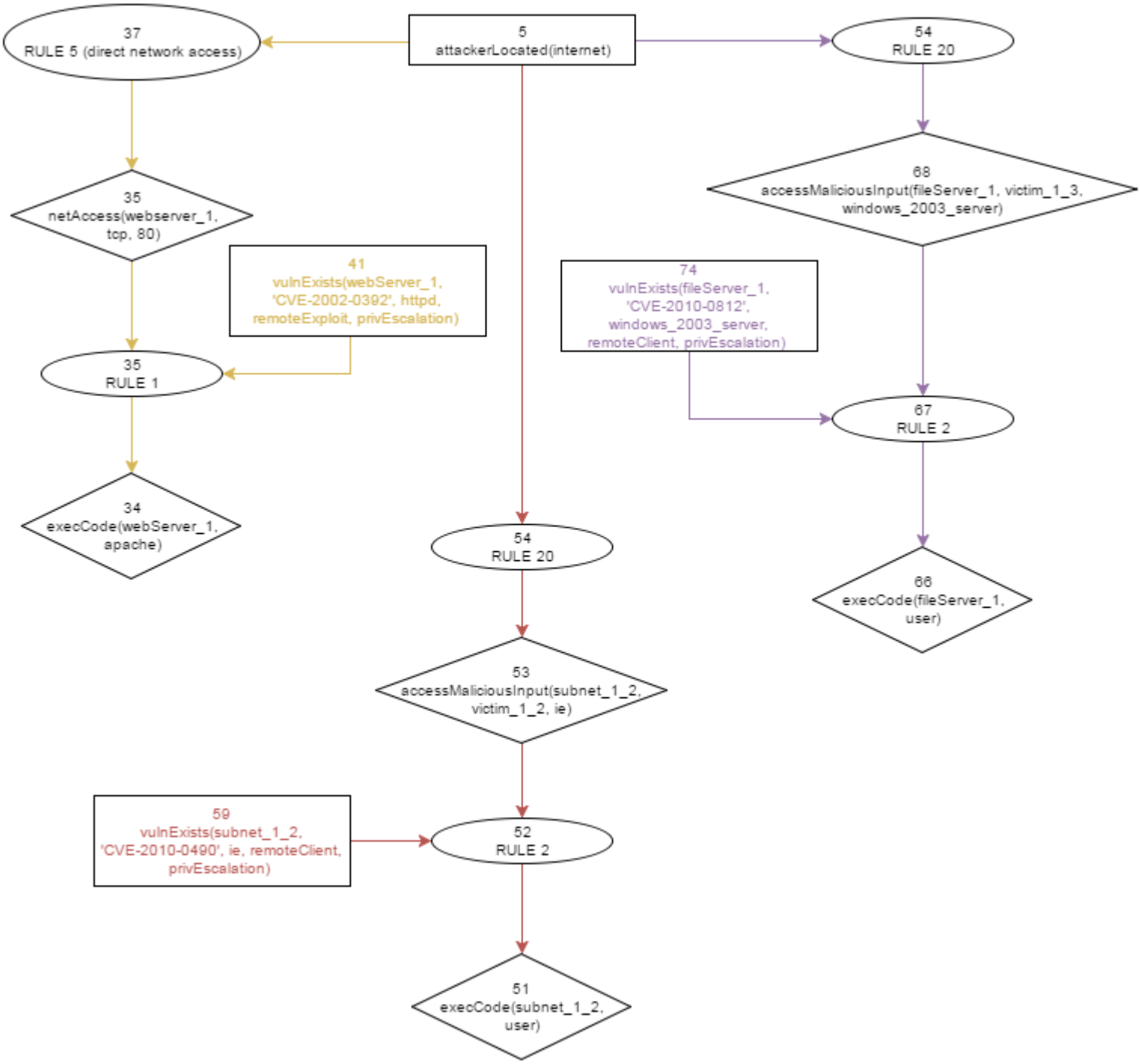

Figure 3. Example attack graph, paths C-yellow, E-red, F-purple.
Edited for clarity and length [24].

7. Conclusion and future work

An attack graph is a tool representing exploitable paths in a computer network. It shows network conditions and exploitable vulnerabilities that allow an attacker to reach a particular set of goals within the system. Attack graph-based security metric aims to aid prioritizing risks and mitigation strategies. Attack graphs show relationships between vulnerabilities and order in which they must be exploited for a successful attack. Existing security metrics analyze the overall security level of a system based on length of the path or mean time to compromise or recover from a compromise. Related works group vulnerabilities around assets for better readability or use age of vulnerability for security assessment. CVSS is a common framework used for scoring criticality of vulnerability based on exploitation and impact factors.

Contribution of this thesis is the path difficulty security metric for attack graphs. Difficulty-based security metric is useful for preparation of equally challenging intrusion methods for participants of technical cyber exercises, or challenges of variously difficult levels to evaluate their technical skills. Penetration testers and CSIRT members provided their knowledge in a survey to distinguish attacker categories by their skill set. Survey showed that three categories are sufficient and that these categories are in fact, supersets of each other as attackers gain more technical and cyber security knowledge. The difficulty rating is derived from filtered CVSS values that reflect specific actions done by an attacker to exploit the vulnerability and are mapped to attacker categories. The path difficulty-based security metric determines the minimal category into which an attacker belongs to in a particular domain if when they successfully exploit a given path.

Future work

Open research questions for future work include:

- collecting operating system and application specific information from the corresponding nodes in attack graphs and concluding that particular path requires sophisticated Windows, Linux, or networking technical knowledge
- integrating the attack graph-based metric with solutions that follow the attacker steps during an attack execution and score the observed actions in real time.

References

[1] I. Paul, "PC World," 21 December 2016. [Online]. Available: http://www.pcworld.com/article/3152367/security/the-10-biggest-hacks-breaches-and-security-stories-of-2016.html. [Accessed 27 December 2016].

[2] E. Dubois, P. Heymans, N. Mayer and R. Matulevičius, "A Systematic Approach to Define the Domain of Information System Security Risk Management," *Intentional Perspectives on Information Systems Engineering*, pp. 289-306, 2010.

[3] "CVE Details," [Online]. Available: http://www.cvedetails.com/. [Accessed 1 December 2016].

[4] S. Samtani, K. Chinn, C. Larson and H. Chen, "AZ Secure Hacker Assets Portal: Cyber Threat Intelligence and Malware Analysis," 2016.

[5] A. Singhal and X. Ou, "Security Risk Analysis of Enterprise Networks Using Probabilistic Attack Graphs," National Institute of Standards and Technology, 2011.

[6] S. Abraham and S. Nair, "A Predictive Framework for Cyber Security Analytics Using Attack Graphs," *International Journal of Computer Networks & Communications*, vol. 7, 2015.

[7] S. Yi, Y. Peng, Q. Xiong, T. Wang, Z. Dai, H. Gao, J. Xu, J. Wang and L. Xu, "Overview on Attack Graph Generation and Visualization Technology," in *IEEE International Conference on Anti-Counterfeiting, Security and Identification*, 2013.

[8] X. Ou, S. Govindavajhala and A. W. Appel, "MulVAL: A Logic-Based Network Security Analyzer," 2005.

[9] "National Vulnerability Database," [Online]. Available: https://nvd.nist.gov/. [Accessed 1 December 2016].

[10] M. A. Alhomidi and M. J. Reed, "Attack Graph Representations," in *4th Computer Science and Electronic Engineering Conference*, 2012.

[11] N. C. Idika, *Characterizing and Aggregating Attack Graph-based Security Metrics*, Purdue University West Lafayette, 2010.

[12] S. Zhang, X. S. A. Ou and J. Homer, *An empirical study of a vulnerability metric aggregation method*, 2011.

[13] CVSS Special Interest Group, "Common Vulnerability Scoring System v3.0 Specification Document," 2015. [Online]. Available: https://www.first.org/cvss/specification-document. [Accessed 3 November 2016].

[14] P. Cheng, L. Wang, S. Jajodia and A. Singhal, "Aggregating CVSS Base Scores for Semantics-Rich Network Security Metrics," in *31st International Symposium on Reliable Distributed Systems*, 2012.

[15] M. Keramati, A. Akbari and M. Keramati, "CVSS-based Security Metrics for Quantitative Analysis of Attack Graphs," in *3rd International Conference on Computer and Knowledge Engineering*, 2013.

[16] Enisa, "Alerts, Warnings and Announcements: Best Practices Guide," 2013.

[17] J.-F. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems," 2000. [Online]. Available: http://freehaven.net/anonbib/cache/raymond00.pdf. [Accessed 3 November 2016].

[18] P. M. Fonash, "Identifying Cyber Ecosystem Security Capabilities," 2012. [Online]. Available: http://m.crosstalkonline.org/issues/3/14/. [Accessed 3 November 2016].

[19] C. Meyers, S. Powers and D. Faissol, "Taxonomies of Cyber Adversaries and Attacks," 2009. [Online]. Available: https://e-reports-ext.llnl.gov/pdf/379498.pdf. [Accessed 3 November 2016].

[20] G. Salles-Loustau, R. Berthier, E. Collange and M. Cukier, "Characterizing Attackers and Attacks: An Empirical Study," in *17th IEEE Pacific Rim International Symposium on Dependable Computing*, 2011.

[21] C. Okoli and S. D. Pawlowski, "The Delphi method as a research tool: an example, design considerations and applications," *Information and Management,* vol. 42, pp. 15-29, 2004.

[22] G. J. Skulmoski, F. T. Hartman and J. Krahn, "The Delphi Method for Graduate Research," *Journal of Information Technology Education,* vol. 6, 2007.

[23] P. Mell, K. Scarfone and S. Romanosky, "Common Vulnerability Scoring System v2.0 Specification Document," 2007. [Online]. Available: https://www.first.org/cvss/v2/guide. [Accessed 1 December 2016].

[24] S. Zhang, "Example MulVAL Network scenarios," [Online]. Available: https://github.com/westlifezs/MulVAL/tree/master/testcases/networkScenarios. [Accessed 30 November 2016].

[25] "Exploit Database," [Online]. Available: http://www.exploit-db.com/. [Accessed 1 December 2016].

[26] "Metasploit: Penetration Testing Software," [Online]. Available: https://www.metasploit.com/. [Accessed 1 December 2016].