An Experimental Approach for Estimating Cyber Risk: a Proposal Building upon Cyber Ranges and Capture the Flags

Giorgio Di Tizio*, Fabio Massacci*, Luca Allodi†, Stanislav Dashevskyi‡ and Jelena Mirkovic§
*University of Trento, Trento, IT
Emails: giorgio.ditizio@unitn.it, fabio.massacci@unitn.it
†Technical University of Eindhoven, Eindhoven, NL
Email: l.allodi@tue.nl
‡Forescout Technologies
Email: stanislav.dashevskyi@forescout.com
§University of Southern California, Marina del Rey, USA
Email: mirkovic@isi.edu

***Abstract*—Current approaches to estimate the risk of compromise are based on either historical data or pure technical assessments, such as the number and severity of vulnerabilities in the target network. We propose a novel experimental approach for estimating the risk of compromise based on experimental data, as opposed to observational data, by leveraging on cyber ranges and capture the flag exercises. We identify the key design principles in terms of response and explanatory variables, specification of how they can be measured, and the overall block design from related experiments and approaches as well as assess their suitability and limitations.**

***Index Terms*—Cyber ranges, Risk assessment, CTFs**

1. Introduction

Quantitative Risk Analysis (QRA) has been increasingly adopted in most industry sectors from nuclear energy to insurance for natural catastrophes [9], [52]. This approach, in contrast with a qualitative analysis, assesses risk based on the use of numbers with an associated meaning. Yet, this adoption is not the standard approach in cyber-security [26], withheld by both technical and organizational difficulties.[1]

Indeed, risk assessment standards and best practices are often qualitative. For example, NIST's Information Security Handbook recommends 'risk matrices' to estimate the likelihood of security events [17]. Unfortunately, qualitative estimations strongly depend on the experts making them [27] and may cause risk miscategorization [48], and even wrong risk prioritization [8], [33].

This problem is worsened by the fact that, whilst the estimation of a vulnerability's technical impact utilizing the Common Vulnerability Scoring System (CVSS) [73] is well understood, its use for *exploitation likelihood* have been empirically questioned both in terms of prediction accuracy and risk reduction. For example, Bozorgi et al. [18] showed that base CVSS technical metrics are not a good predictor for the eventuality that exploits will be engineered. Allodi and Massacci [3] showed that patching 100+ vulnerabilities bundled in exploit kits yields a risk reduction of 40% of attempted attacks in the wild whilst the recommendation of the Payment Card Industry [62], based on the qualitative indicator of CVSS [37], only yields a 4% risk reduction whilst looking at 10K+ vulnerabilities.

How to progress further towards *quantitative estimation of the likelihood of a compromise?* We propose a *methodology for laboratory and testbed experiments* to estimate the likelihood of a compromise. In this respect, the setting of 'Class Capture The Flag' (CCTF) [54] may provide the clean set-up and the large pool of experimental data needed to yield insight into the quantitative estimation of compromise likelihood. This can be later scaled to large and possibly automated [79] testbed experiments, playing the role of large scale clinical trials [69].

Next, we present the related works (§2), we propose our experimental approach (§3) and identify both response (§4) and explanatory variables (§5), discuss the block design (§6), present possible extensions (§7), the limitations of this approach and the conclusions (§8).

2. Related Works

We now describe the related works on risk assessment and class capture the flags.

2.1. CTFs competitions

Computer security CTFs like Defcon CTF [30], UCSB iCTF [80], Build it Break it Fix it [65], and the DARPA CGC [79] are international competitions for security enthusiasts. Smaller cyber-security challenges are now widely used for educational purposes. Different designs of CCTF exists in the literature with varying degrees of complexity of the tasks and time available to participating students. We summarize some of them in Table 1. However, CTFs are not limited to educational purposes and can be used for the analysis of many security aspects [70]. To the best of our knowledge, no one has proposed a methodology for risk estimation based on CTF exercises.

[1] See for example the debate on the Verizon breach report [60]. Further, big datasets (e.g. reporting perimeter alarms from an *Intrusion Detection System* [24]) are often unstructured [13] and the statistical link between alarm data to actual breaches is hard to make [76].

2.2. Risk Assessment

The academic literature has produced a rich set of methodologies for risk estimation based on attack surfaces, attack graphs, game-theoretic models, and time-to-compromise [4], [27]. However, they often rely on assumptions based on expert judgments instead of real data [4]. On the other hand, several papers addressed cyber risk estimation by using security data exhausts (data about attacks and exploits in the wild) employing machine learning [18], [43], [86], statistical analysis, regression using big data analytics [4], [5], [24] and case-control studies [3]. More recent approaches employs ML algorithms on data from Twitter and dark web forums [6], [25], [66]. Most of the empirical studies regarding cyber risk assessment are based on case-control studies where attack traces in the wild are analyzed to determine, for example, which people are more likely to be a victim of targeted attacks [50], [75], the web server characteristics associated with a higher rate of compromise [77] and the behaviors and applications positively correlated with the probability of being infected by malware [22], [83]. However, case-control studies have some limitations as the causal relation between attack phases and the attack measurement can be only approximated [64]. Instead, CTF exercises carried in a controlled environment allow us to better isolate the factors related to the likelihood of compromise and assess their impact through multiple experiments. Compared to the narrowed point of view offered by data exhausts we instead have a complete view of the attack procedure, the attacker's assets, and intentions.

3. An Experimental Approach

Several definitions of risk exist based on probability and impact, uncertainty and (expected) consequence [11]. In practice they ultimately collapse to the intuitive relation

$$\begin{aligned} Risk =& Impact \times Likelihood \\ =& Impact \times Pr(Attack) \times Pr(Compr|Attack) \end{aligned} \tag{1}$$

Whilst estimating $Impact$ is a well understood process [7], [23], [44], [45] and $Pr(Attacks)$ is a matter of threat intelligence, the focus is estimating the probability of compromise given an attack ($Pr(Compr|Attack)$).

3.1. Research Questions

The overall goal is to set up an *experimental methodology* that estimates the *empirical hardness* of exploiting the vulnerabilities in a network, $Pr(Compr|Attack)$, against the actual configuration of the network and the related skills needed to take advantage of such flaws. Several factors can determine such probability but to define an experiment we propose to distill two of them:

- Which network configuration has the highest likelihood of being compromised by attacks perpetrated by attackers with a given set of skills?
- Which defenders' or attackers' skills for a given network configuration yield a higher compromise likelihood?

3.2. Experimental Methodology

We need an experimental set-up in which the experimenter can at least realize whether an attack has been successfully carried on. Security data exhaust cannot address these issues because:

> "with data exhaust we don't know if are measuring dim attackers getting caught instead of smart attackers getting through." (R. Clayton)

Our key idea is to monitor the outcomes of *Capture The Flag* exercises carried inside a *cyber range*, since it is easier to control and more amenable to replication studies.

The experiment set-up would then follow the protocol reported by [49]:

- **Pre-assessment** Before the execution of any activity, subjects are given a questionnaire to collect information on their background and knowledge of attack techniques.
- **Training**. A scenario description is administered to subjects by either an individual reading or by an introductory presentation. Then, a training phase follows in which the expert in the testbed introduces its functionalities[2] through a tutorial.
- **Application**. The subjects apply their attacking skills to the scenario. Part of the design decisions would also include the presence (or absence) of defenders. We discuss this later in §6.
- **Evaluation**. In this phase, the outcome of the CTFs is analyzed to identify for example if and how a red team has successfully compromised the system. If an automatic assessment is not possible, then external evaluators assess the results of the CTFs. These evaluators should possibly *not* be the experimenters but rather external experts contracted for the purpose. If human defenders are included there should be an assessment also of their activities to be used as a controlling factor of the possible mitigation effect that this might have had. It is important to underline that the external evaluators determine the outcome of the exercises (network compromised/not compromised) and do *not* assess the risk of compromise of the network.
- **Post-assessment**. A post-task questionnaire is conducted to gather subjects' perception of the tools they used and on the experiment as a whole.

If it is possible, the post-measurement phase could be organized into focus groups to discuss the drawbacks and benefits of various methods. A list of questions is used to guide the discussion. This qualitative analysis attempts to throw light on the features affecting the actual and perceived efficacy of attack techniques to further improve the assessment of these factors. For example, two red teams can be graded with the same level of knowledge on a certain programming language but one team successfully compromised the network while the other one failed. In the post-assessment phase it is possible to know that, for example, some functions or libraries utilized by the first team to compromise the target are not known by the second team. This allows improving the questionnaire to

[2] E.g. lectures on particular attacking tools or attack techniques.

TABLE 1: CCTF Experimental Approaches

Paper	Duration	Preparation	Instrumentation
Mirkovic and Peterson [54]	2 hours	Defense and attack scenarios (2/3 weeks preparation before CTF)	Linux Machine, LAMP servers
Werther et al. [82]	18 hours	Defense and attack scenarios (access VM 1 month before, evening lect.)	VMs, Wordpress and LAMP stack
Wagner and Wudi [81]	2 days	Defense and attack scenarios (24h hardening, 24h attack)	Linux, Windows unpatched machines
Brustoloni [20]	2-3 weeks	Only defense scenarios (lectures about attacks and countermeasures)	Cluster of machines w/ VPN and firewall
Chothia and Novakovic [28]	2-3 weeks	Only attack scenarios (lectures, application of attacks)	Offline Linux VM, web and DB server
Bock et al. [16]	3 hours	Defence and attack scenarios (machines divided in *territories*)	Linux,Windows unpatched with subnets

better evaluate the skills of the subsequent exercises. It is important to underline that this last phase is not used for any risk assessment in our methodology but to improve the overall experiment structure.

In terms of actual realization the experiments should take place in structured testbed (e.g., DETER [15], ViSe [10], and vGrounds [46]). The use of a structured testbed can help in achieving greater control over the execution environment, isolation among experiments, and reproducibility. A comparison of network-based experimental security testbeds can be found in the Master's thesis by Stoner [71]. Table 2 shows some available testbeds and experimentation tools from the academic literature.

3.3. Students, Ethical Hacker or Black Hacker?

It is important to notice that a CCTF is *not* restricted to BSc/MSc students. The same approach can be used for professionals to whom an advanced scenario must be presented. For example, the training-execution-evaluation has been used for testing the efficacy of industry security catalogs for risk assessment with professionals with 10+ years of experience [32]. The student vs professional issue is well debated in software engineering but less in the security field.

There is another issue that requires attention. The relevant population to consider when assessing the risk of compromise is not necessarily students *or* professionals, but rather ”criminals”. The population of cybercriminals may differ from students and professionals in significant ways [40], [41]. It's not really clear whether the distribution of skill levels is the same as students or professionals [19] albeit there is some preliminary evidence that it may be very similar. It is a popular opinion to consider malicious hackers more skilled than penetration testers. However, as pointed out by Shim et al. [67], highly skilled hackers may prefer legal activities with respect to criminal actions due to the lowest risk and higher profit.

Another aspect that requires attention is the level of motivation between these groups: students or professionals playing a CTF have plenty of motivation to continue to attack a target, at least for the duration of the competition, while real-world attackers may quickly give up and move on to lower-hanging fruit. For example, it is rare for attackers in the real world to find bugs on their own and develop exploits for them (i.e., zero-day vulnerabilities), but it is extremely common in CTFs. This criticism can be representative of a subset of black hackers, the so-called script-kiddies. However, it is not representative of Advanced Persistent Threats (APTs) [40] that are often driven by economic factors and therefore can have motivations comparable to professional penetration testers.

In conclusion, even if the population of students and professionals could not describe *exactly* the nature of malicious attackers, it can be used to generate reliable attack scenarios. Furthermore, current red teaming approaches are based on the simulation of Tactics, Techniques, and Procedures (TTPs) from known threat actors to reproduce realistic scenarios of risk [36]. The simulation is made easier thanks to the presence of open-source Offensive Security Tools (e.g. PowerSploit, Meterpreter, and Mimikatz) that are widely used by malicious actors [74].

4. Response Variables

From Eq. (1) the factor of interest is the likelihood of compromise $Pr(Compr|Attack)$. We now describe two possible approaches to compute this factor.

4.1. Frequency-based approach

A simple measure might be the number of teams that were able to successfully attack (within the duration of the experiment) divided by the total number of teams in the experiment, assuming all teams were at the same skill level and have access to the same tools. However, this would suffer from several limitations in terms of transferability of the results. Another possibility is to define the likelihood as the inverse of the *time to compromise* τ from the moment in which the attack started to the moment in which a pre-defined compromise is achieved (i.e. attacked network's logs reveal the malicious behavior). This requires to have a pre-defined notion of what is a successful attack:

- denial of service on the defenders' machines;
- compromise of the integrity of operations;
- data exfiltration from a database.

We believe that each attack type should be measured separately and be subject to different experiments rather than collapsed in a single experiment.

At first, in practice most companies associate a different impact evaluations to a different type of attack [38] and therefore Eq. (1) should actually be weighted sum by attack type (in terms of confidentiality (C), integrity (I), and availability (A)).

$$Risk = \sum_{type \in \{C,I,A\}} Impact_{type} \times Pr(Attack_{type}) \times Pr(Compr|Attack_{type}) \qquad (2)$$

Second, attackers are specializing themselves (e.g. booter services aka DoS-for-hire [41]) and it might be appropriate for the analysis to distinguish them.

Eq. (1) can be also extended considering data from threat intelligence to get a fine-grained risk estimation in terms of the threat actors of interest. If we consider any

TABLE 2: Security testing and experimentation tools

Tool	Description	Exploit types and Structure
BugBox [56]	A corpus and exploit simulation environment for PHP web application (WordPress, CuteFlow, Drupal, etc.).	Selenium and Metasploit scripts in Python that exploit PHP application vulnerabilities.
TestREx [31]	A corpus and exploit simulation environment for Web application (WordPress, CoreApp, JS-YAML ODataApp, etc.).	Uses Docker and Selenium as well as provide exploits scripts.
vGround [46]	A simulation environment for Linux worms analysis.	Uses UML virtual machines, provides scripts to set-up, delete, and collect data from the nodes.
MalwareLab [2]	Controlled environment for experimenting with EK.	Runs Drive-by Download attacks against different software configurations.
MINESTRONE [35]	A software vulnerability testing framework for C/C++ programs.	Deployed in Linux Containers. Detect vulnerabilities like memory corruption, null pointer, and resource leak.
SecuBat [47]	Web vulnerability scanner based on a web crawler.	Crafted HTTP requests that exploit SQLi and XSS vulns.
DETER [15]	A shared testbed facility for cybersecurity experiments.	Based on Emulab and containers, provide scripts to generate and manage the experiment infrastructure.
ViSe [10]	A virtual testbed for digital forensic based on VMWare.	A set of exploits for O.S, browsers, and web applications.
Labtainers [42]	A simulation environment for Linux-based software exploitation (e.g. BoF, format string, and crypto attacks).	Based on Docker. Provides scripts to generate environments and automatically assess experiments.

attack on the system we have that $P(Attack) \approx 1$ for a sufficiently long interval of time [14], [34]. However, the probability of an attack can vary depending on the threat actor's skills. It is more common to be targeted by script-kiddies or criminal groups than nation-state actors [78], which traduces in different skill levels and assets. We can, for example, classify threat actors following the STIX notation[3] in minimal (m), intermediate (i), advanced (a), expert (e), innovator (i), and strategic (s). Depending on the threat intelligence information, we can associate a different $P(Attack_{skills})$, that describes the probability that an attacker with this level of skills will target the network. The different skill levels will likely traduce in different impacts ($Impact_{skills}$) on the network[4].

$$Risk = \sum_{skills \in \{m,i,a,e,s\}} Impact_{skills} \times Pr(Attack_{skills}) \times Pr(Compr|Attack_{skills}) \tag{3}$$

A fine-grained description of the $P(Attack)$ based on the level of the attackers can also be useful to tune the experiment. For example, if $P(Attack_{skills}) = 0.9$ with $skills = minimal$, it is not of practical interest to consider several teams with an advanced level of skills, instead, it is more useful to consider several different groups with a minimal or intermediate skills level. We are not claiming that the data obtained from threat intelligence gives an exact description of the skills level of the adversaries a company is facing. However, this information can be helpful for a raw classification.

4.2. Level-based approach

An alternative solution would be to compare the *Attack Score* α that the participants to the experiment would be able to gauge. Several metrics could be used depending on the type of attack, based on OWASP or NIST guidelines (e.g. employing the CWE vulnerability classification). For example, privilege acquisition can be measured by defining the level of authorization achieved by the attack (e.g. on a three-level ordinal scale $< admin, system, local >$). Similarly, the overall level of privilege can be measured in terms of the 'fraction' of system resources that the acquired level allows access to. Likewise attacks exfiltrating data can be measured in terms of the relative fraction of data extracted (e.g. in a database). Code execution can be measured by the reliability with which the payload is successfully executed by multiple instances of the attack (e.g. considering uncertainty introduced by the O.S. scheduler if a race condition must be won, etc.). However, measuring exfiltration as the fraction of data that is leaked may not be optimal since, in many cases, exfiltration is an all-or-nothing proposition. The same applies for privileges, it is not important the fraction of a system available to the attackers, but the actual privileges to sensitive data that they have gained.

Thus, given the choice between the frequency and level-based approach, we believe the former is more reliable than the latter for a first experiment since the overlap of the scores mentioned above could make collecting the evidence of their effects extremely difficult [29], [59]. Further, as the choice of attack scores is essentially expert dependent, this would re-introduce the serious problems behind the subjectivity of likelihood estimation [8], [33] that we were trying to eliminate. For example, in the case of exfiltration, some data can be considered more important than another depending on the situation. Making the definition of scores dependent on the specific scenario.

5. Explanatory Variables

We now describe how can we measure the explanatory variables that we identified in the experiment structure.

5.1. Measuring Attacker Skills

The type of skills has clearly an impact on the likelihood of compromise. The difficult question is how to correctly assess them.

Some studies analyzed which cognitive skills help to perform well in computer programming. In particular, Harvey et al. [39] showed that a high score in the systemizing quotient is positively related to the ability to solve hacking challenges. Nevertheless, these results are not related to the experience of the subjects. To evaluate the skills level of the participants, an assessment of computer security topics must be performed.

[3] https://stixproject.github.io/

[4] Whose value is not necessarily positively correlated with the skill level. For example, a wiper malware deployed by script-kiddies could have a higher impact than the exfiltration of sensitive data carried by an APT.

SANS has developed a web-based skills assessment tool [5] to assess the skill set of information security experts. It provides both general assessments regarding cyber-security as well as more technical tests on specific topics like penetration testing, cyber defense, and forensic. The questions for the entry-level test are broadly divided into four areas and are summarized in Table 3. Unfortunately, the SANS questionnaire is far too long for a simple experiment: it requires over an hour to assess the participant's skills. It might be useful for more advanced experiments. Further, we do not know of any published evidence that those questions are useful to assess the skills of attackers.

A possible alternative is based on the OWASP Top 10 [6], which represents a broad consensus about the most critical security risks to web applications. One of the limitations of OWASP is the narrow technical focus as well as the large emphasis on web application attacks. Also, there is no proper questionnaire and one would, therefore, need to recast the topics of the OWASP tutorial into a questionnaire.

Self-assessments provide a quick way to evaluate the skills but they might be misleading. On the opposite side of the spectrum is the evaluation through small wargames (simple cyber-security challenges) that produces a more reliable evaluation but it is too expensive in terms of time.

Therefore, a combination of self-assessment and a few questions SANS-style to control for the accuracy of the former could be the right strategy. In the course of our experience, we found out that it is always better to "instantiate" the semantics of classical terms such as "Familiar", and "Expert" to something more precise when asking participants to perform a self-assessment. For example for programming languages using "Attended some classes" and "Developed a large project" would provide a semantics that is less prone to interpretation.

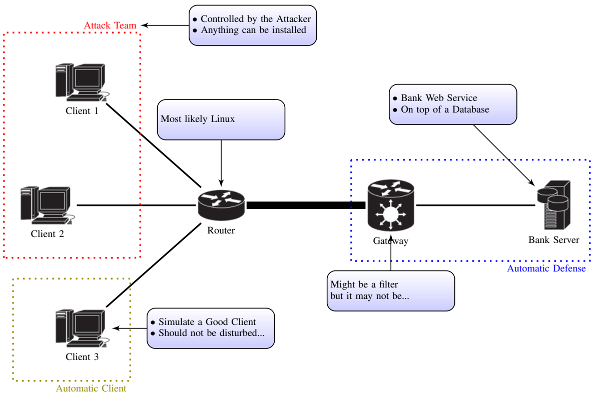
The assessment of the attacker skills should depend on the information collected from the threat intelligence. The knowledge about which threat actors are active in the specific fields and their TTPs allows one to identify the skills needed to emulate these attacks.

Defender skills can be assessed in the same way. Additionally, the NISTs NICE Framework [55] can be used to identify the skills and abilities required for specific roles in the Blue team.

5.2. Designing and Measuring Vulnerability Profiles

For a first experiment, we might want to eliminate all confounding factors due to the network topology (for example the presence of firewalls partly segregating part of the network). Hence we propose to adopt the simple network layout used in some CCTFs [54]. The network topology is shown in Fig. 1.[7]

When deciding the trials it is possible to identify two alternatives *Vulnerability Profiles* $\mathcal{V}$:



(a) The experiment emulates an online bank where a user can register, deposit some money, withdraw it and check the balance. The defender has access to the gateway and the server, meanwhile the *attacker* has access to the clients Client 1, and Client 2. Client 3 is used by the experimenter to implement a legitimate client.

Figure 1: The baseline experimental topology from [54]

- **Single**: a configuration with a single vulnerability (e.g. SQL injection, Buffer overflow) so that the attack should be carried by attacking it.
- **Global**: a configuration with a set of vulnerabilities so that the attackers can choose the exploit.

Several configurations are then tested in round each determining a new experimental trial. Having a learning phase is important in this set-up, to avoid learning effects during the trials that might unfavorably bias the experiments. These two setups are complementary in that they allow variance by controlling the two key variables identified by our research questions: variability of the configuration, and variability of the attack choices. Whereas the interplay between these two aspects can be difficult to be controlled in a single experimental instance, running separate experiments can shed light on the mutual impact of each factor. Some configurations may influence the probability of success of certain types of attacks, or favor a particular type of attacker skills.

In general, the first set-up may represent an easier feat as varying configurations are straightforward to measure (as they are imposed by the experimenter); in this setup, attackers are asked to deploy only one (set of) attack(s), whose success can be easily assessed: the successful execution of, say, an SQL injection can be measured by the extraction of specific values from a dataset, or the execution of custom code by having it 'calling home' to a default location, proving execution as in [2]. If teams find "unintended" vulnerabilities this might be an interesting deviation. The only way to mitigate this would be to provide the knowledge of where the vulnerabilities are to the teams and require that only those vulnerabilities be attacked, which removes the reconnaissance phase of attack from the experiment and reduces its realism. Alternatively, teams are required to provide proof of concepts (PoCs) to verify that the right vulnerability has been exploited.

The second setup allows for greater variability as it may be hard to foresee all ways attackers will proceed. By sacrificing control it allows for greater realism as, in reality, the attacker can choose. If needed, such scenarios may be accurately measured by performing some level

[5] https://www.sans.org/cybertalent/assessment-products

[6] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[7] To have a slightly better realism it might be useful to add a simple Windows node on the side of the defender corresponding to the machine of the system administrator or the internal user of the company.

TABLE 3: CyberSecurity Skill Assessment - SANS

Admin.	Type	Description
System	Access logging, Incident handling, IDS	Maintenance, configuration, and reliable operation of computer systems; especially, multi-user computers, such as servers.
Network	Network topologies, TTL, DNS configuration, Subnets, IR Logging Access, Switch, Proxy, Packet Filters	Maintenance of network infrastructures such as switches and routers, and diagnosis of related problems or the behavior of network-attached computers.
Web	HTTP, network protocols (TCP/IP, UDP), SQL injection	Maintenance of web server services (such as Apache or IIS) that allow for internal or external access to web sites. Management of multiple sites, administration of security, configuration of necessary components and software.
Security	Encryption, ARP spoofing, ICMP, ping, traceroute, DDL injection	Computer and network security, including the administration, of security devices such as firewalls, as well as consulting on general security measures.

of 'live' dynamic analysis, or by requiring participants to script their attacks to allow for an (automated) follow-up assessment on dynamic analysis platforms such as Cuckoo. From a global perspective, it is still possible to cast them as a one-bit answer (for example for data exfiltration one could simply expect a dataset to be uploaded in a suitable location). However, to be able to assess the second type of trial we need to assign a score to the overall network configuration to capture the whole set of vulnerabilities. In this case, a further design decision is to define a metric for the network.

In this respect, several metrics have been provided in the academic literature based on attacks, among which the percentage of compromised hosts [51], the weakest adversary to compromise a network [61], and attack likelihood [58]. For example, [57] proposed to use 'existence' (the relative fraction of vulnerable network services), 'exploitability' (the average value of the CVSS exploitability score across all available network services), or 'impact' (the average value of the CVSS impact score across all available network services). More sophisticated measures use attack graphs and vulnerability graphs [1] or a biodiversity-inspired metric, that models the network diversity to capture the resilience to zero-day attacks [85]. They would not be appropriate for a simple scenario as the one envisaged in Figure 1. For example Zhang's metrics about diversity [85] would always be constant as everything is based on Linux in the scenario on Figure 1.

A possible solution is to use the same simple measures currently used by industry and that constitutes the backbone behind the Payment Card Industry Certification [62]. Such scoring systems are close to the ones proposed in [57] and has the advantage to make the result of the experiments immediately understandable by industry. Also in terms of the analysis of the result, it provides two individual values for the statistical analysis.

We assume that the *network score* $\mathcal{N}$ is a global metric that is described by the following equation:

$$\mathcal{N}_{net} = \max_{v \in Net} \{\text{CVSS Env. Score}(v)\} \qquad (4)$$

$$\mathcal{N}_{loc} = \max_{v \in Loc} \{\text{CVSS Env. Score}(v)\} \qquad (5)$$

where $v \in Net$ are the network vulnerabilities belonging to the network analyzed, $v \in Loc$ are the vulnerabilities that have been detected with a vulnerability assessment system, and *CVSS Env. Score* is a function computing the vulnerability severity by accounting for 'environmental' (in the language adopted by NIST) factors as defined by the CVSS SIG team [37]. The outcome of the experiments might also be that they do *not* work (i.e. correlate with the resulting $P(Compr|Attack)$).

6. Block Design

The last steps of the design must consider the impact of the defender and the overall set up of the testbed environment.

6.1. Controlling for the effectiveness of Defense

The final design decision is whether we should consider a defense at all. In this respect we identify two *Defense scenarios* $\mathcal{D}$:

- **in vitro** where the attacker has to perform against a system that is automatically managed by the experimenter. Such a system is not necessarily static but may have an automated patch evolution as in [2].
- **in vivo** in this scenario the attacked system is managed by a defender who may upgrade the system defenses, detect the attack, and arbitrarily respond to it.

For the latter solution, in a CCTF scenario, one normally starts with a set of balanced groups that should all be considered attackers. It is, therefore, necessary to have a staggered assignment of the same groups to defenders so that no group has to defend against more than one attacker at the same time and vice versa. For example one could have Group i attack Group $i+1$. After the first trial, such staggering should be suitably permuted in order to have a balanced design.

For the in vitro scenario, we don't need to check the outcome of the defense as it is essentially determined by the vulnerability profile $\mathcal{V}$ and the initial network score $\mathcal{N}$ chose by the experimenter.

The in vivo scenario requires to control the result against the quality of the defense team. The simple solution is, of course, to control for their self-assessed skills when performing the final statistical analysis (for example a logit regression analysis considering them as a factor). However, a better solution exists which includes using the same metrics used to rate the initial system in the first place. This can be achieved by performing a standard vulnerability assessment from the network to evaluate the system configuration in terms of active services and local configurations as well as known vulnerabilities on the system. This can be performed *before* and *after* the operation of the defenders to evaluate the changes that their actions have on the system. These changes can be quantified in terms of the relative change in the overall score of the system ($\mathcal{N}_{after} - \mathcal{N}_{before}$). This evaluation has the drawback to recognize only patches and configuration changes and it does not address live mitigations.

A solution to address this limitation is to compare the number of successful exploitations carried by a team in the ”in vitro” and the ”in vivo” scenario on the same network configuration. In this case, the only difference is given by the defender actions that can be evaluated as the decrement of successful exploits. The drawback is that two experiments must be carried (one ”in vitro” and one ”in vivo”) to compute the quality of the defense.

The evaluation of the effectiveness of defense must take into account not only the security of the service but also its availability and performance. Indeed, perfect security can be achieved with no functionality [63]. It is necessary to check the availability of the services as well as their performance. This can be done using bots that simulate clients' activities and evaluate the availability and the performance of the defended network as done by Client 3 in Fig. 1.

6.2. Setting the environment

A key difficulty in this experimental set-up is the implementation of the application scenarios. Indeed web applications are deployed and run in many different execution environments, consisting of operating systems, web servers, database engines, and other sorts of supporting applications in the backend, as well as different configurations in the frontend [53]. This difficulty can be illustrated with typical exploits for the two types of web application security vulnerabilities: SQL injection exploits, where the success depends on the capabilities of the underlying database and the authorizations of the user who runs it [72, Chapter 9], and Cross-site Scripting (XSS) exploits, where the success depends on a specific web browser being used and its rules for executing or blocking JavaScript code [84, Chapter 14]. Small differences in software environments may transform failed exploitation attempts into successful ones, and vice versa.

Currently, docker-based framework solutions like Labtainers [42] and TestREx [31] reduce the effort required to build and test cybersecurity labs. Labtainers allows to implement environments for cyber-challenge and provides an automatic collection of data as well as forensic indications of activities. TestREx allows one to automatically generate the configurations required to simulate a certain scenario. It combines packing applications and execution environments that can be easily and rapidly deployed, scripted exploits that can be automatically injected, and isolation between running instances of applications to provide a real “playground” and experimental setup.

An important point would be that every time a new trial is launched the participants would start with a clean configuration. The idea of automatically loading a series of clean configurations every time before an exploit is launched has been also proposed by Allodi et al. [2].

6.3. The Final Programme

The final experiment structure can be divided into two distinct but related experimental phases:

- **individual trials** are run first with the purpose to measure the correlations between the skills and the likelihood of compromise. They are performed on individual vulnerabilities in which the subjects are given a single system with a specific vulnerability and are told to exploit that particular vulnerability. They intuitively correspond to *in vitro* clinical pretrials according to the FDA[8].
- **team trials** are run after the individual trials to have a better realism and to gauge the potential impact of defenders. They correspond to *in vivo* pre-trials.

7. Moving toward Automated Risk Analysis

New directions can look to completely automated systems to detect vulnerabilities, apply countermeasures, and evaluate the risk. This approach can eliminate the uncertainty in the evaluation of attacker and defender skills. The first move in this direction was the DARPA Cyber Grand Challenge (CGC), where autonomous cyber reasoning systems (CRSs) were used in a special CTF without human interactions. The implementation and the performance of these systems have been discussed extensively [79]. In terms of risk estimation the CGC showed that through the application of rules and penalties for software replacement, automated systems can evaluate the risk of their actions. In particular, CRSs strategies to patch software is evaluated on the risk that a specific vulnerability could be exploited and by the evidence that this situation can occur. The competition showed that CRSs can often act more rationally than humans [12].

Unfortunately, these systems can be easily deceived using honeybugs [12], [21]. This kind of traps can be easily identified by humans but not by machines. A possible solution to overcome this limitation is a *human-assisted automated analysis* [68] where human actions are used as aids for automated vulnerability analysis. Even if this approach does not eliminate the uncertainty related to human skills, it reduces its impact on the assessment of risks.

8. Conclusions and Limitations

In this paper, we have described an experimental plan for the empirical assessment of the likelihood of compromise of a network based on the idea of using ‘Class Capture The Flag’ exercises. We have identified both response and explanatory variables, specified in detail how they can be measured, as well as the overall block design including how to control for eventual defense abilities during in vivo experiments. Yet several issues remain open and we discuss them to further encourage research in this direction.

At first, the CTF framework does not capture the timescales on real-world APTs. There is evidence that advanced attackers will often have control over portions of a target network for months, and will adapt to changing network conditions and slowly learn what valuable data may be on the network. This is challenging to be replicated in a CTF environment. This scenario could be described through a sequence of CTFs, where the

[8] https://www.fda.gov/ForPatients/Approvals/Drugs/ucm405658.htm

defender team has to implement network modifications on the environment of the previous CTF. The environment of the CTF is frozen and the new CTF maintains access to the assets obtained from the previous competition.

Another issue is how do we scale up to large networks. A possible idea would be to leverage on the data provided by the scan of corporate networks to create networks that are *artificial but realistic*. Starting from the data collected, an enumeration is performed concerning the different types of subnets, the connective components between them (e.g., gateway), and the different kinds of users. Then a new network is generated in which every node of a category is replaced by a randomly sampled node with the same configuration of a node in the original network. In this way, we have a full testbed that can be the object of more sophisticated experiments.

Another issue is the relation of these metrics with current industry practices of ”penetration testing” with internal and external teams. The scoring methods used by these industry engagements may vary and we need to find a way to map these metrics to the ones we are proposing.

Finally, there is the issue of consistency. For example, if the same experiment is conducted with a different set of people (with similar skills), will the results be consistent? This is one of the key issues with current practices of red-teaming where the results do not replicate well and depend on the team performing the assessment and the tools they bring. Specifically, what kind of instrumentation will be used to collect data about the attacks and defenses will be essential to understand whether the results are due to the superior knowledge of the participants or the latest version of Metasploit. The instrumentation might be also useful to understand how many attempts an attacker needed to create a working exploit, or how ”noisy” an attack is.

Acknowledgments

This research has been partly funded by the EU under the H2020 Programs H2020-EU.2.1.1-CyberSec4Europe (Grant No. 830929)

References

[1] M. Albanese and S. Jajodia. A graphical model to assess the impact of multi-step attacks. In *JDMS'17*, JDMS, 2017.

[2] L. Allodi, V. Kotov, and F. Massacci. Malwarelab: Experimentation with cybercrime attack tools. In *Proc. of CSET'13*, 2013.

[3] L. Allodi and F. Massacci. Comparing vulnerability severity and exploits using case-control studies. *ACM TOPS*, 2014.

[4] L. Allodi and F. Massacci. Security Events and Vulnerability Data for Cybersecurity Risk Estimation. *Risk Analysis*, 2017.

[5] L. Allodi, F. Massacci, and J. Williams. The work-averse cyber attacker model. evidence from two million attack signatures. In *Proc. of WEIS'17*, 2017.

[6] M. Almukaynizi, E. Nunes, K. Dharaiya, M. Senguttuvan, J. Shakarian, and P. Shakarian. Proactive identification of exploits in the wild through vulnerability mentions online. In *Proc. of CyCon-17*, 2017.

[7] R. Anderson, C. Barton, R. Bhme, R. Clayton, C. Ganan, T. Grasso, M. Levi, M. Vasek, and T. Moore. Measuring the changing cost of cybercrime. *Proc. of WEIS'19*, 2019.

[8] L. Anthony Tony Cox. What's wrong with risk matrices? *Risk analysis*, 2008.

[9] G. E. Apostolakis. How useful is quantitative risk assessment? *Risk Analysis*, 2004.

[10] A. Arnes, P. Haas, G. Vigna, and R. A. Kemmerer. Digital forensic reconstruction and the virtual security testbed vise. In *Proc. of DIMVA'06*, 2006.

[11] T. Aven and L. A. Cox. National and global risk studies: How can the field of risk analysis contribute? *Risk Analysis*, 2016.

[12] T. Avgerinos, D. Brumley, J. Davis, R. Goulden, T. Nighswander, A. Rebert, and N. Williamson. The mayhem cyber reasoning system. *IEEE Security & Privacy*, 2018.

[13] S. Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM TOPS*, 2000.

[14] W. Baker, M. Howard, A. Hutton, and C. D. Hylender. 2012 data breach investigation report. Technical report, Verizon, 2012.

[15] T. Benzel. The science of cyber security experimentation: The DETER project. In *Proc. of ACSAC'11*, 2011.

[16] K. Bock, G. Hughey, and D. Levin. King of the hill: A novel cybersecurity competition for teaching penetration testing. In *Proc. of ASE'18*, 2018.

[17] P. Bowen, J. Hash, and M. Wilson. Information security handbook: a guide for managers. In *NIST Special Publication 800-100*. NIST, 2006.

[18] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker. Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In *Proc. of SIGKDD'10*, 2010.

[19] S. Bratus. What hackers learn that the rest of us don't: notes on hacker curriculum. *IEEE Security & Privacy*, 2007.

[20] J. C. Brustoloni. Laboratory experiments for network security instruction. *ACM JERIC*, 2006.

[21] T. Bryant and S. Davenport. A honeybug for automated cyber reasoning systems. *IEEE Security & Privacy*, 2018.

[22] Y. Carlinet, L. Mé, H. Debar, and Y. Gourhant. Analysis of computer infection risk factors based on customer network usage. In *Proc. of SECURWARE'08*, 2008.

[23] B. Cashell, W. D. Jackson, M. Jickling, and B. Webel. The economic impact of cyber-attacks. *CRS Documents*, 2004.

[24] H. Cavusoglu, B. Mishra, and S. Raghunathan. The value of intrusion detection systems in information technology security architecture. *Inf. Sys. Res.*, 2005.

[25] H. Chen, R. Liu, N. Park, and V. Subrahmanian. Using twitter to predict when vulnerabilities will be exploited. In *Proc. of SIGKDD-19*, 2019.

[26] P.-y. Chen, G. Kataria, and R. Krishnan. Correlated failures, diversification, and information security risk management. *MISQ*, 2011.

[27] Y. Cherdantseva, P. Burnap, A. Blyth, P. Eden, K. Jones, H. Soulsby, and K. Stoddart. A review of cyber security risk assessment methods for scada systems. *Computers & Security*, 2016.

[28] T. Chothia and C. Novakovic. An offline capture the flag-style virtual machine and an assessment of its value for cybersecurity education. In *Proc. of 3GSE'15*, 2015.

[29] C. Cowan. Defcon capture the flag: Defending vulnerable code from intense attack. In *DARPA ISCE'03*, 2003.

[30] C. Cowan, S. Arnold, S. Beattie, C. Wright, and J. Viega. Defcon capture the flag: Defending vulnerable code from intense attack. In *DISCEX-III*, 2003.

[31] S. Dashevskyi, D. R. Dos Santos, F. Massacci, and A. Sabetta. TestREx: a testbed for repeatable exploits. In *Proc. of CSET'13*, 2014.

[32] M. de Gramatica, K. Labunets, F. Massacci, F. Paci, and A. Tedeschi. The role of catalogues of threats and security controls in security risk assessment: An empirical study with ATM professionals. In *Proc. of REFSQ'15*, 2015.

[33] N. J. Duijm. Recommendations on the use and design of risk matrices. *Safety Science*, 2015.

[34] T. Dumitras and P. Efstathopoulos. Ask wine: are we safer today? evaluating operating system security through big data analysis. In *Proc. of LEET'12*, LEET'12, pages 11–11, 2012.

[35] N. S. Evans, A. Benameur, and M. Elder. Large-scale evaluation of a vulnerability analysis framework. In *Proc. of CSET'13*, 2014.

[36] FireEye. Red Team Assessment. [online] https://www.fireeye.com/services/red-team-assessments.html.

[37] First.org. Common vulnerability scoring system v3.0: Specification document. Technical report, FIRST, 2015.

[38] M. Giacalone, R. Mammoliti, F. Massacci, F. Paci, R. Perugino, and C. Selli. Security triage: A report of a lean security requirements methodology for cost-effective security analysis. In *Proc. of ACM ESEM'14*, 2014.

[39] I. Harvey, S. Bolgan, D. Mosca, C. McLean, and E. Rusconi. Systemizers are better code-breakers: Self-reported systemizing predicts code-breaking performance in expert hackers and nave participants. *Frontiers in Human Neuroscience*, 2016.

[40] A. Hutchings. Crime from the keyboard: organised cybercrime, co-offending, initiation and knowledge transmission. *Crime, Law and Social Change*, 2014.

[41] A. Hutchings and R. Clayton. Exploring the provision of online booter services. *Deviant Behavior*, 2016.

[42] C. E. Irvine, M. F. Thompson, M. McCarrin, and J. Khosalim. Live lesson: Labtainers: A docker-based framework for cybersecurity labs. In *Proc. of ASE'17*, 2017.

[43] J. Jacobs, S. Romanosky, I. Adjerid, and W. Baker. Improving vulnerability remediation through better exploit prediction. 2019.

[44] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams. Cauldron mission-centric cyber situational awareness with defense in depth. In *MILCOM*, 2011.

[45] G. Jakobson. Mission cyber security situation assessment using impact dependency graphs. In *FUSION'11*, 2011.

[46] X. Jiang, D. Xu, H. J. Wang, and E. H. Spafford. Virtual playgrounds for worm behavior investigation. In *Proc. of RAID'05*, 2006.

[47] S. Kals, E. Kirda, C. Krügel, and N. Jovanovic. Secubat: a web vulnerability scanner. In *Proc. of WWW'06*, 2006.

[48] H. Kunreuther. Risk analysis and risk management in an uncertain world. *Risk Analysis*, 2002.

[49] K. Labunets, F. Massacci, F. Paci, and L. M. Tran. An experimental comparison of two risk-based security methods. In *Proc. of ESEM '13*, 2013.

[50] M. Lee. Whos next? identifying risks factors for subjects of targeted attacks. In *Proc. Virus Bull. Conf*, 2012.

[51] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R. Cunningham. Validating and restoring defense in depth using attack graphs. In *IEEE MILCOM'06*, 2006.

[52] C. Lomnitz. *Global tectonics and earthquake risk*. Elsevier, 2013.

[53] G. A. D. Lucca and A. R. Fasolino. Testing web-based applications: The state of the art and future trends. *Inform. and Soft. Tech.*, 2006.

[54] J. Mirkovic and P. Peterson. Class capture-the-flag exercises. In *Proc. of 3GSE'14*, 2014.

[55] W. Newhouse, S. Keith, B. Scribner, and G. Witte. NICE Cybersecurity Workforce Framework, 2017. [online] https://doi.org/10.6028/NIST.SP.800-181.

[56] G. Nilson, K. Wills, J. Stuckman, and J. Purtilo. Bugbox: A vulnerability corpus for PHP web applications. In *Proc. of CSET'13*, 2013.

[57] S. Noel and S. Jajodia. Metrics suite for network attack graph analytics. In *CISR'14*, 2014.

[58] S. Noel, L. Wang, A. Singhal, and S. Jajodia. Measuring security risk of networks using attack graphs. *IJNGC*, 2010.

[59] E. Nunes, N. Kulkarni, P. Shakarian, A. Ruef, and J. Little. Cyber-deception and attribution in capture-the-flag exercises. In *Proc. of ASONAM'15*, 2015.

[60] OSVDB. A note on the Verizon DBIR 2016 vulnerabilities claims, 2016. [online] https://blog.osvdb.org/2016/04/27/a-note-on-the-verizon-dbir-2016-vulnerabilities-claims/.

[61] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup. A weakest-adversary security metric for network configuration security analysis. In *Proc. of ACM QP'06*, 2006.

[62] Payment Card Industry Security Standard Council. *PCI Data Security Standard (DSS): Requirements and Security Assessment Procedures*, 2010.

[63] B. Price, M. Zhivich, M. Thompson, and C. Eagle. House rules: Designing the scoring algorithm for cyber grand challenge. *IEEE Security & Privacy*, 2018.

[64] S. Ransbotham and S. Mitra. Choice and chance: A conceptual model of paths to information security compromise. *Inf. Sys. Res.*, 20, 2009.

[65] A. Ruef, M. W. Hicks, J. Parker, D. Levin, M. L. Mazurek, and P. Mardziel. Build it, break it, fix it: Contesting secure development. In *Proc. of CCS'16*, 2016.

[66] C. Sabottke, O. Suciu, and T. Dumitras. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *Proc. of USENIX'15*, 2015.

[67] W. Shim, L. Allodi, and F. Massacci. Crime pays if you are only an average hacker. In *Proc. of IEEE ASE CyberSec'12*, 2012.

[68] Y. Shoshitaishvili, M. Weissbacher, L. Dresel, C. Salls, R. Wang, C. Kruegel, and G. Vigna. Rise of the hacrs: Augmenting autonomous cyber reasoning systems with human assistance. In *Proc. of CCS'17*, 2017.

[69] A. Somayaji, Y. Li, H. Inoue, J. M. Fernandez, and R. Ford. Evaluating security products with clinical trials. In *Proc. of CSET'09*, 2009.

[70] T. Sommestad and J. Hallberg. Cyber security exercises and competitions as a platform for cyber security experiments. In *Secure IT Systems - NordSec 2012*, 2012.

[71] E. L. Stoner. A foundation for cyber experimentation. Master's thesis, Florida Institute of Technology, 2015.

[72] D. Stuttard and M. Pinto. *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*. John Wiley & Sons Inc, 2007.

[73] C. S. Team. Common vulnerability scoring system v3.0: Specification document. Technical report, First.org, 2015.

[74] A. Thompson. Misconceptions: Unrestricted Release of Offensive Security Tools, 2019. [online] https://medium.com/@QW5kcmV3/misconceptions-unrestricted-release-of-offensive-security-tools-789299c72afe.

[75] O. Thonnard, L. Bilge, A. Kashyap, and M. Lee. Are you at risk? profiling organizations and individuals subject to targeted attacks. In *Proc. of FCDS'15*, 2015.

[76] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer. Comprehensive approach to intrusion detection alert correlation. *IEEE Trans Dependable Secure Comput.*, 2004.

[77] M. Vasek, J. Wadleigh, and T. Moore. Hacking is not random: A case-control study of webserver-compromise risk. *IEEE Trans. Dependable Sec. Comput.*, 2016.

[78] Verizon. 2019 data breach investigations report, 2019. [online] https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf.

[79] T. Vidas, P. Larsen, H. Okhravi, and A. Sadeghi. Changing the game of software security. *IEEE Security & Privacy*, 2018.

[80] G. Vigna, K. Borgolte, J. Corbetta, A. Doupé, Y. Fratantonio, L. Invernizzi, D. Kirat, and Y. Shoshitaishvili. Ten years of ictf: The good, the bad, and the ugly. In *Proc. of 3GSE'14*, 2014.

[81] P. J. Wagner and J. M. Wudi. Designing and implementing a cyberwar laboratory exercise for a computer security course. In *Proc. of SIGCSE'04*, 2004.

[82] J. Werther, M. Zhivich, T. Leek, and N. Zeldovich. Experiences in cyber security education: The MIT lincoln laboratory capture-the-flag exercise. In *Proc. of CSET'11*, 2011.

[83] T. Yen, V. Heorhiadi, A. Oprea, M. K. Reiter, and A. Juels. An epidemiological study of malware encounters in a large enterprise. In *Proc. of SIGSAC'14*, 2014.

[84] M. Zalewski. *The Tangled Web: A Guide to Securing Modern Web Applications*. No Starch Press, 2011.

[85] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese. Network diversity: A security metric for evaluating the resilience of networks against zero-day attacks. *IEEE Trans. Information Forensics and Security*, 2016.

[86] S. Zhang, X. Ou, and D. Caragea. Predicting cyber risks through national vulnerability database. *Information Security Journal: A Global Perspective*, 2015.