Article

Evaluation of Attackers' Skill Levels in Multi-Stage Attacks [†]

Terézia Mézešová *, Pavol Sokol (ID) **and Tomáš Bajtoš**

Faculty of Science, Pavol Jozef Šafárik University in Košice, 040 01 Košice, Slovakia; pavol.sokol@upjs.sk (P.S.); tomas.bajtos@student.upjs.sk (T.B.)

* Correspondence: terezia.mezesova@upjs.sk

† This paper is an extended version of our paper published in International Workshop on Systems Safety and Security—IWSSS 2019, Pitesti, Romania, 27–29 June 2019.

check for updates

Abstract: The rapid move to digitalization and usage of online information systems brings new and evolving threats that organizations must protect themselves from and respond to. Monitoring an organization's network for malicious activity has become a standard practice together with event and log collection from network hosts. Security operation centers deal with a growing number of alerts raised by intrusion detection systems that process the collected data and monitor networks. The alerts must be processed so that the relevant stakeholders can make informed decisions when responding to situations. Correlation of alerts into more expressive intrusion scenarios is an important tool in reducing false-positive and noisy alerts. In this paper, we propose correlation rules for identifying multi-stage attacks. Another contribution of this paper is a methodology for inferring from an alert the values needed to evaluate the attack in terms of the attacker's skill level. We present our results on the CSE-CIC-IDS2018 data set.

Keywords: alert correlation; attack evaluation; attacker skill level

1. Introduction

The increasing number of systems connected to the Internet presents a new set of risks for organizations as they become an interesting target not only for opportunistic attacks but targeted multi-stage attacks as well. Multi-stage attacks consist of several steps and are executed in logical follow-up steps.

Security operation centers monitor the activity within an organization's network for various threats and employ a wide range of tools to provide situational awareness to responsible asset owners. Large organizations especially grapple with a lot of legitimate network traffic, and they experience a massive number of alerts that are generated by intrusion detection systems. In such an environment, it is difficult for the analysts to filter out the noise and to discover logical relations between the alert and construct attack scenarios on a higher abstract level that the asset owners will be able to process.

One of the most common practices is to correlate events from intrusion detection systems into an attack path, a so-called multi-stage attack. These attacks are further prioritized, and the aim is to minimize the number of attacks that analysts must investigate. In threat and risk analysis, risks associated with vulnerabilities that are considered difficult to exploit are often given low priority for treatment. Therefore, analysts should be monitoring such risks and checking for any attacks targeting such vulnerabilities. They should have a comprehensive framework available so that they are able to evaluate how difficult a detected attack is to execute and treat it with the appropriate priority.

Information about attacks can be forwarded to risk management, and the appropriate countermeasures should be given a higher priority. To foster the feedback between operations and

security monitoring, there should be a common understanding of the terms difficulty of an attack and difficulty of vulnerability exploitation.

Our objective in this research is to propose a set of correlation rules that will connect alerts from intrusion detection systems into more meaningful attack paths that reflect multi-stage attacks. Furthermore, a contribution of this paper is also a revised methodology first presented in [1]. It provides a common framework for inferring values needed for the evaluation of an attacker's skill level needed to exploit a vulnerability and an attacker's skill level needed or demonstrated that leads to a particular alert being raised by an intrusion detection system. It creates a relationship between what kinds of properties are evaluated for a vulnerability and how alerts can be evaluated with regard to such properties. The contribution of this paper is that we can determine the skill level needed to execute multi-stage attacks, especially the enhanced determination of how to establish the appropriate metric values on which the skill level evaluation is based.

To formalize the scope of this paper, our research objectives were:

- Define alert correlation rules to identify multi-stage attacks.
- Revise the framework for evaluating an attacker's skill level with regard to alerts.

This paper is organized into five sections. In Section 2, we discuss the related works on attacker profiles and the behaviors of attackers. Section 3 presents the rules used for alert correlation and a revised method for evaluating the skill level needed to raise alerts by an intrusion detection system. Finally, we present the results and discuss on example cases on data from a public data set in Section 4.

2. Related Works

In this section, we discuss similar works in the area of attacker skill modeling with a focus on the attackers' capabilities. In terms of quantitative analysis, Paulauskas and Garsva [2] suggested using the mean time to compromise within a normally-distributed interval as an evaluation of attacker's skill. Hu, Liu, Zhang and Zhang [3] used an absorbing Markov chain to extract various properties of attack scenarios and attackers, such as the estimated probability of reaching each attack target and the estimated occurrence number of each alert in the attack scenario. A risk estimation method that incorporates attackers' capabilities in estimating the likelihood of threats as conditions for using means and opportunities is presented by Othmane et al. [4]. They demonstrated the use of the proposed risk estimation method on video conferencing systems and connected vehicles. Additionally, they focused on evaluating whether incorporating attacker capabilities reduces the uncertainty in the experts' opinions and showed that changing attackers' capabilities changes the risks of the threats.

Rensburg et al. [5] proposed a method of generating a set of attack graphs, parameterized by attacker profiles. Vertices correspond to states of the network and an attacker, and edges correspond to actions that the attacker can take. They defined profiles as collections of capabilities and generated complete sets of profiles. This ensures analysis is not only about common types of attackers. An attacker model in networked embedded systems is presented in [6], which is used to sense, actuate and control physical processes. The authors defined an attacker framework with mapping 23 attacker profiles from related work into that framework. They defined a distance metric that allows computing overlap and discrepancies between attacker models in related work. GAMFIS [7] defines attacker types taking into account all of an attacker's attributes—motivation, skills and resources. Attacker models are further extended in [8] using types of attacker manipulation with a node, monitoring capabilities and movement strategies. Within node manipulation, they distinguish (I) attackers who are able to extract a part of the key material from the compromised node, (II) attackers who compromise the node, extracting all key material and installing their malware and (III) attackers who compromise the node and actively influence the behavior of the node. According to the attackers' capabilities, they distinguish global and local attackers.

The behavior of attackers is considered in [9]. The authors strive for a more refined attacker model introducing the attacker's view of a system. This view drives the actions of the attacker, depending on

the knowledge and resources the attacker possesses. A Markov decision process models the behavior of the attacker as the method for the selection of attack steps. In [10] the attacker's skill level is classified based on the level of unauthorized access that an attacker has reached in a network. Using a multilayer fuzzy logic, attackers are divided into three categories (high, medium and low). Hassan and Guha deployed deception to characterize further the relationship between an attacker and their target [11]. They aimed to develop reliable capabilities. Their results demonstrate an association between average skill level and overall effectiveness and success of deception in unique ways.

3. Methodology

3.1. Data Set

In this paper, we worked with the data set CSE-CIC-IDS2018 published by Sharafaldin, Lashkari and Ghorbani [12]. The data set contains seven different attack scenarios: brute-force, heartbleed, botnet, denial of service, distributed denial of service, Web attacks and infiltration of the network from the inside. Infrastructure includes 50 machines belonging to the attackers, and the victim organization includes 420 machines and 30 servers. All the machines were hosted by Amazon Web Services. The data set includes the captured network traffic and system logs of each machine from several days over the course of weeks over February and March 2018.

The methodology that the authors of the data set present makes the data set suitable for various research purposes. It was used in many research papers, most recently in comparative studies on machine learning and deep learning methods for intrusion detection [13,14] or to verify the intrusion detection method for the Internet of Things environment [15].

3.2. Processing Data and Identification of Attacks

In this section, we present the methodology on processing data into security alerts and a process of attack identification. An overview of the full process is illustrated in Figure 1. Firstly, the raw data set that is available as a network capture is processed through an intrusion detection system (IDS) which generates alerts based on its available rules. Then, the alerts are input for our aggregation and correlation rules, and we are left with the identified multi-stage attacks. Finally, we evaluate each hyper-alert with the appropriate metric values and give a final evaluation of the attacker's skill level.

For data set processing, we used the intrusion detection system (IDS) Suricata (https://suricata-ids.org) with two sets of alert rules. The first rule set is one that is provided by Suricata by default. Company Proofpoint provides the second open rule set called Emerging Threats (https://rules.emergingthreats.net/). Rules from the first rule set are marked as "GPL" and rules from the second rule set are marked as "ET". The outputs from Suricata IDS were event logs, consisting of security alerts, which were further processed.

Alert processing can be divided into three stages: alert pre-processing, alert aggregation and alert correlation. At the end of the process, the results are grouped into hyper-alerts with some level of the relation between them.

At first, we selected 8 days with multiple interesting attacks, in particular Wednesdays, Thursdays and Fridays. Raw alerts from those days were normalized into the format expected by the next stages. From the original raw alert returned by Suricata, we kept only the parameters below. Further, we filtered out noisy alerts and alerts with low severity, which we expect to be benign, and therefore, they represent false positives. It is possible that not all false positives were removed from the data, but it significantly reduced the number of noisy alerts. At this point, the alert pre-processing stage is finished.

- Date and time;
- Source and destination IP addresses;
- Source and destination network ports;
- Protocol;

- Identifier of the rule (sid);
- Rule name or message;
- Alert severity.

The second stage is alert aggregation. The network intrusion detection systems generate big amounts of log data that might or might not be of interest for security-related purposes. Aggregation and therewith the reduction of the number of the alerts are necessary steps. We split the data set into distinct days, and for each day, we ran the aggregation process separately.
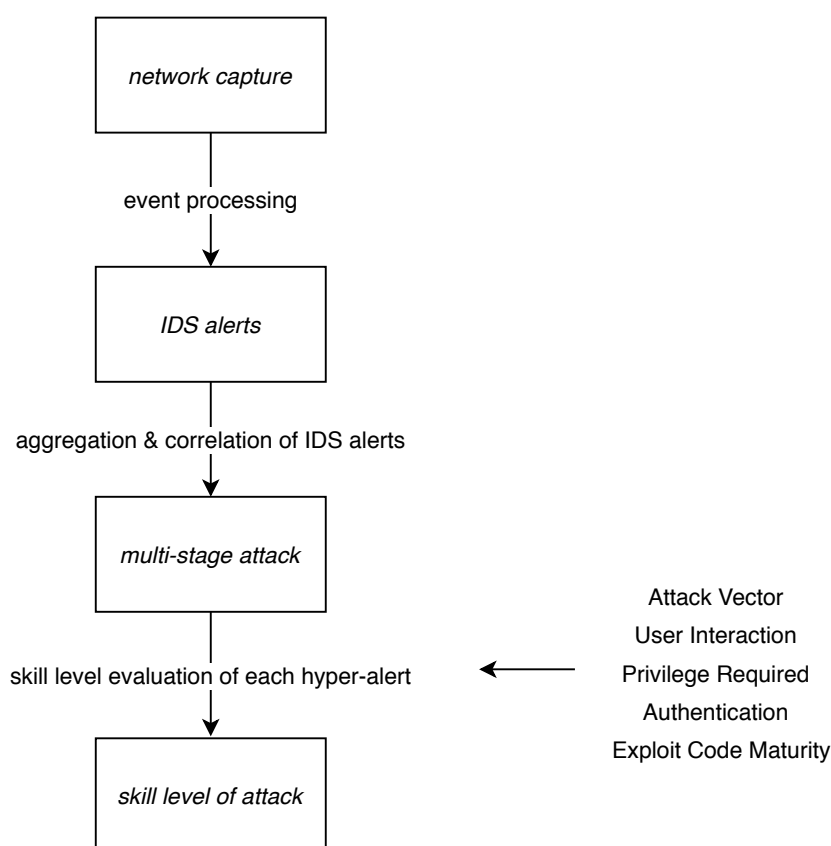

Figure 1. Overview of the processing and evaluation stages.

We use a sliding time window of size t, in which we aggregate alerts if they have the same alert sid, the same source IP address and the same destination IP address. The aggregated alerts with the time difference between the last seen alert and a new incoming alert of the same type larger than the size of the aggregation time window are excluded from the aggregation window, and a new instance of hyper-alert is created. The appropriate size of a time window t is between 5 and 30 min. With a bigger size of the time window, alerts which originate from different attack vectors may be aggregated together, and even more importantly, the reduction of alerts is not as significant anymore.

At the end of this stage, we have aggregated hyper-alerts that share the same features. In the structure of hyper-alerts, we left original alerts as a separate parameter, but in further processing, the whole aggregated hyper-alerts are used. The individual parameters of the alerts are united into the sets of source IP addresses, destination IP addresses and sids. The time parameter is represented in hyper-alerts as start time, which represents an occurrence of the first alert, and end time, which represents an occurrence of the last alert aggregated into the same hyper-alert. This is an example of output after the aggregation phase. In the Results section, we present only the message for clarity.

- ```
  2018-03-02 17:50:16.251431, 114.86.88.5, 172.31.69.26, 2010939
  ET SCAN Suspicious inbound to PostgreSQL port 5432
  ```

In the third stage, the aggregated hyper-alerts were correlated according to predefined correlation rules based on an alert similarity. Each of the six predefined rules uses its own similarity function. The similarity functions take sid, source IP address and destination IP address as inputs and return a value true if the appropriate rule is fulfilled, and else return false. The rules are applied in a defined order, and two hyper-alerts are compared if they occurred in the same time window. For the size of the time window, the same applies as for the aggregation window. The rules in exact order are:

1. Each of two compared hyper-alerts has at least one identical source IP address and at least one identical destination IP address.
2. Each of two compared hyper-alerts has at least one identical sid and at least one identical destination IP address.
3. Each of two compared hyper-alerts has at least one identical sid and at least one identical source IP address.
4. Each of two compared hyper-alerts has at least one identical sid and at least one of the destination IP addresses of the first alert; and one of the source IP addresses of the second alert is the same for both.
5. Each of two compared hyper-alerts has at least one source IP address in common with the other.
6. In each of two compared hyper-alerts, at least one of the destination IP addresses of the first alert and one of the source IP addresses of the second alert are identical.

The correlation stage uses the same structure for alerts as the aggregation stage, so the alert outputs are also called the hyper-alerts. Each rule represents some similarity between alerts. For example, the fourth and the sixth rule represent a hopping of the attack from one IP address to another. The first rule represents different kinds of alerts, but between the same IP addresses. All of those rules are applied within the relatively short time window so that we can expect those individual alerts have some similarity. Therefore, we assume that the correlated hyper-alerts are sequences of alerts that belong to the same attack.

3.3. Evaluation of Attacker Skill Level of the Hyper-Alert

In this section, we present a revised methodology from [1] on evaluating the necessary skill level of an attacker from the detected alerts raised by intrusion detection systems (IDS). The methodology aims to be a corresponding partner to determining a skill level needed to exploit a vulnerability as presented in [16], so that both vulnerabilities and IDS alerts can be evaluated under the same framework. Reference [16] presents the results of an expert survey on determining an attacker's skill level needed to exploit a vulnerability or a sequence of vulnerabilities successfully. Three skill levels were defined. Parts of their definitions are also more concrete actions that attackers with said skill levels are capable of performing. These were then in the second round of survey mapped to individual values of metrics in the common vulnerability scoring system (CVSS), as defined in [17,18]. In this paper, we are using the mappings between the CVSS metric value and skill level category, as presented in the referenced paper.

The available skill level categories TM: Confirming the changes. are from least (minimal) to most skilled (maximum): script kiddies, moderately skilled attackers and highly skilled attackers. The script kiddie category represents attackers with basic IT knowledge, and their capabilities are limited to running publicly available tools or exploits. Moderately skilled attackers can customize their toolbox to fit their attack target better, and combine various components to perform an attack. Highly skilled attackers are described as attackers with in-depth, often professional technical knowledge, and can create functional exploitative codes, which are custom-fit towards their target networks.

That paper was adapted to reflect the skill level needed to raise an IDS alert in subsequent paper [1]. Firstly, individual hyper-alerts were evaluated. In this paper, we present revised keywords and observations of an IDS alert or the alert definition and the values of the individual CVSS metrics.

Equation (1) shows that to determine the skill level of an individual hyper-alert sl_{alert}, the maximum skill level of five metrics is taken. The maximum of the evaluated skill level values for individual metrics is suitable because an attacker must demonstrate a more mature level to trigger an IDS alarm successfully.

$$sl_{alert} = max(m^{AV}, m^{UI}, m^{PR}, m^{Au}, m^{EC}) \tag{1}$$

After each hyper-alert is evaluated individually, we can evaluate the entire correlated path of hyper-alerts by choosing the maximum skill level of the present individual evaluations. We assign the lowest possible skill level to the whole path— this represents the minimum level of skill to raise the alerts detected by the IDS.

For each hyper-alert, we trace to the original alert rule according to the sid parameter, which is a unique identifier of each rule. Rules might include a reference to a vulnerability's identifier, in which case the skill level needed to trigger the alert in the network is the same as the skill level needed to exploit the originating vulnerability. IDS raise alerts based on observations of events, and they can be arbitrarily written by security analysts and often do not correspond to any particular vulnerabilities at all. A good example is reconnaissance scanning. Scanning tools such as nmap can, when configured, be used to determine whether any known vulnerabilities are present in the target network. The values of the individual metrics can be determined for a hyper-alert from the alert rules' definitions, positions of the source and destination IP addresses, how quickly the alerts are triggered and from analyst's manual determination. The values are then mapped to the skill level category, as shown in Table 1.

A hyper-alert's evaluation of script kiddies means that the least skilled attackers are able to perform the actions detected by the hyper-alert. Network hosts where such hyper-alerts are occurring in high volumes should be prioritized for patching or selected for closer monitoring. On the other hand, the occurrence of alerts evaluated with a highly skilled attacker should notify the security analysts to report a security incident and execute a response plan. The skill level metric can provide further context for the statistics reported by security teams to the stakeholders that are not familiar with the cybersecurity domain.

Table 1. Keywords or observations that distinguish skill level categories.

Keywords or Observations	Corresponding Metric Value	Skill Level
external source IP address	m^{AV}: Network	*script kiddies*
internal source & *external* destination IP address	m^{AV}: Network	*script kiddies*
internal source & destination IP address	m^{AV}: Adjacent	*moderately skilled*
small time differences in timestamps or regularity	m^{UI}: None	*script kiddies*
none - explicitly stated so references	m^{UI}: Required	*moderately skilled*
default	m^{PR}: None/Low	*script kiddies*
classtype "*successful-admin*" or "*Admin*" in message	m^{PR}: High	*moderately skilled*
default	m^{Au}: None	*script kiddies*
"*Authentication Success*" in message	m^{Au}: Single	*script kiddies*
more alerts with "*Authentication Success*"	m^{Au}: Multiple	*moderately skilled*
scanning or *Metasploit* in message or classtype	m^{EC}: High	*script kiddies*
malware names in message or classtype	m^{EC}: High	*script kiddies*
references in rule to ready-to-use payloads	m^{EC}: High	*script kiddies*
tagged with *SQL injection* or *Cross-site scripting*	m^{EC}: High	*script kiddies*
classtypes *rootkit* or *backdoor* (or in message)	m^{EC}: Functional/Proof of Concept	*moderately skilled*
classtype *Web-application-attack*	m^{EC}: Functional/Proof of Concept	*moderately skilled*
default	m^{EC}: Unproven	*highly skilled*

The attack vector metric m^{AV} defines the position of an attacker relative to the target or defended network. It can distinguish between script kiddies and moderately skilled attackers, respectively, if an attack can be performed from a remote network (e.g., Internet), or if a local connection is required.

It can be determined from hyper-alert by checking the source and destination IP addresses. If the alert is originating from an external network, which is the definition for attack vector metric value network, the skill level script kiddies is assigned. Commonly, the targeted host will respond, and this might also cause IDS to raise an alert. In the response scenario, the source IP address is internal, and the destination IP address is external. This is still within the understanding of the (m^{AV}) value network definition, and so the mapped skill level remains script kiddies. If the attack is originating from and is also targeting the internal network, an attacker has previously gained access to an internal machine in some way, and we assign the skill level moderately skilled. For the scope of this paper, we do not distinguish between an insider attacker and a truly external attacker. Outside methodologies ought to be used to recognize an insider, e.g., reference [19], and the skill level can be lowered or increased accordingly by an analyst. The attack vector metric value "local" need not be considered, as it is assigned to vulnerable components that do not have network connections [18]. Thus, no incoming or outgoing network traffic will be present for IDS to monitor.

The user interaction metric m^{UI} reflects whether a legitimate user's interaction is necessary or not to trigger the alert under evaluation. The determined value can be stored in the alert rule's metadata, so it can be adjusted if there are any changes to the alert rule conditions.

By default, the value none is taken for all alerts, thereby assuming the categorization of script kiddies. This is because scanning, automated brute-force tools or further lateral movements are executed without the need for a legitimate user's action. If there is a very small time difference in the timestamps or when we observe the alert occurring at regular intervals, this is very likely an automated tool that works on its own.

However, many initial infiltration techniques, such as phishing or malware, do require it. By keeping the skill level low by default and then searching for evidence that would support us in declaring that a legitimate user interacts with the system before IDS triggers an alert with the particular message, we stay closer to reality with the skill level evaluation. Some work has been done on automatically determining the value for the user interaction metric for vulnerabilities in [20,21]. Namely, [21] state that the word "file" was an important factor in predicting the value "required" for user interaction. IDS alert messages, however, are not as verbose as vulnerability descriptions from which the authors infer the metric value. Such alerts where user interaction is required can be identified by the security analyst manually when creating new rules, and adjusted accordingly for existing rules when the final evaluation of the skill level for a hyper-alert or correlated path seems insufficient.

Another important feature to consider is whether two-factor authentication is set up for any parts of the system monitored by the IDS. Two-factor authentication can help remedy many breaches of personal accounts and is a recommended best practice. At this point, it is necessary to look also at the human aspect of two-factor authentication and take into account the existence of the several issues (e.g., the vulnerabilities of using a third-party authentication provider) [22]. Security analysts should verify whether two-factor authentication is set up for the accounts and systems within their constituency. In such a case, they can evaluate the user interaction metric m^{UI} for the relevant rules with value required. If two-factor authentication is not enforced in the system, m^{UI} remains of the default value "none".

The privilege required metric m^{PR} aids in further evaluating the attacker's skill level. CVSS specification document defines this metric as "*the level of privilege an attacker must possess before successfully exploiting the vulnerability*" [18]. We aim to answer whether the attacker needs admin privilege to raise the observed IDS alerts. Possible research directions for integrating role-based/context-aware access control solutions can be found in [23]. That must not be confused with alerts classified as attempting to gain or successfully gaining user or administrator privilege.

The default value for this metric is considered none, meaning that the attacker can raise the alert under evaluation without any user privileges. It maps to the lowest skill level—script kiddies. The majority of scanning, malware and Web activities can be performed out of the box like this. A moderately skilled attacker is able to obtain administrative privileges to the system.

We, therefore, observe alerts whose rule definitions contain the classtype "successful-admin" or keywords "admin access" in their messages [24]. An example, although not directly present in the data set we worked on, is an alert "ET MISC HP Web JetAdmin ExecuteFile admin access".

In the context of evaluating IDS alerts, the privilege required metric can be adapted as the level of privilege an attacker must possess to successfully execute the behavior leading an IDS to raise an alert. The specification document distinguishes between no authorization needed, authorization for basic tasks and authorization for significant control in the system. As such, it works with many access control models and can sufficiently reflect the deployment of role-based or context-aware access control solutions in the monitored network.

The authentication metric m^{Au} can distinguish between script kiddies and the moderately skilled category by counting the number of times any authentication takes place. By default an alert is assigned the value none, giving the lowest category—script kiddie. If in the correlated path of hyper-alerts, we have one alert containing the key phrase "authentication success" or similar, we will assign it with authentication metric value "single". It will still give us the script kiddies category. When we have two or more alerts with this message, it means that the attacker passed two or more authentication gates, and we assign the authentication metric to multiple. In such a case, the skill level raises to moderately skilled. An example alert with such keyword would be "ET POLICY VNC Authentication Successful". However, this alert was not raised in the processing of the data set we worked on for this paper.

The exploit code maturity metric m^{EC}, a temporal metric, is not part of the standard vulnerability score evaluation by a vendor when a vulnerability is publicly disclosed. It is left to the analyst to determine its value when needed and should be periodically checked. The metric has four possible values—high, functional, proof of concept, unproven—each representing the likelihood of the vulnerability being attacked, and is typically based on the current state of exploit techniques, exploit code availability or active "in-the-wild" exploitation [18].

In the context of the IDS alerts, we ought to determine whether the activity causing those alerts to be raised is a result or part of a publicly well-known exploit, implementation of a published proof-of-concept attack or evidence of an exploit for a vulnerability that was thought theoretical until now. There is common agreement that the more technical details are available about how to exploit a vulnerability, the less skilled of an attacker can successfully execute the exploit, and that increases the number of potential attackers. It is also possible that the highly skilled attackers will perform their operations in such a way that their attack steps will be obscured by legitimate traffic in the network or they will be spread throughout several days or weeks; they will obscure their IP address, thereby avoiding any correlation between the raised IDS alerts.

To evaluate the m^{EC} metric for a hyper-alert, let us assume that by default, all activity is a result of a theoretical exploit and assign the value unproven. This will yield us the highest skill level category—highly skilled attackers. Then, we are searching for evidence that will support us in deciding that the skill level category is objectively lower. The following sentences also reflect the order in which it should be executed, and once we reach the lowest skill set, we stop and do not consider further factors. References in the alert rule can link to websites describing attack steps and their detection, or explanation of the rule's logic, or a website providing the payloads which the detection rule is aiming for. They are evidence that we should evaluate at most with value proof of concept, so we are lowering the category to moderately skilled. The classtype of the IDS alerts can be useful here as well. Ready for use scanning tools, metasploit modules or malicious software do not require any additional modification, and so their exploit code maturity is high—mapping to the script kiddies category. Rootkits and backdoors might need to be compiled for the target host or even programmed from scratch by following a published proof of concept, so this maps to value functional or proof of concept and establishes the skill level as moderately skilled. Web application attacks are a broad class, and we cannot determine just one value for all of them with a reasonable degree of certainty. In general, any available tools to execute Web application attacks must be further modified to the

target specifics, which is beyond the capabilities of script kiddies as defined in [16]; therefore we still are at the level of moderately skilled attackers. However, SQL or OS command injection attacks or cross-site scripting attacks are attacks will well-documented execution steps and fully autonomous tools are available, or a list of payloads to try is available. The value for m^{EC} metric for alerts with SQL injection, or cross site scripting (XSS) is high—which maps to the script kiddies level.

In any case, as it is a temporary metric, each alert with value unproven or proof of concept ought to be regularly checked if there were any exploits created and made public since the initial evaluation. The metric value should be changed accordingly.

4. Results and Discussion

In this section, we present the results of the correlation of alerts and the evaluation on four attacks from the data set.

4.1. Denial of Service Attack

The GoldenEye denial of service attack is well visible in the processed data. It was detected five times in a 30 min window from one external IP address targeting an internal machine. Data set description does not provide further details on the attack execution.

1. 5x `ET DOS Inbound GoldenEye DoS attack`

Let us present the evaluation flow of the skill level for this alert. The attack vector metric gets value network because the source IP address is external. There is a reference to the GitHub repository with the GoldenEye Python script; therefore, exploit code maturity is high. All other metrics assume the default value none. They all map to the skill level script kiddies—the maximum of all values. The mapping to the skill level is shown in Equation below. The skill level needed to raise the hyper-alert `ET DOS Inbound GoldenEye DoS attack` is script kiddies.

$$
\begin{aligned}
sl_{GoldenEye} \quad &= max(m^{AV} = Network, m^{UI} = None, m^{PR} = None, m^{Au} = None, m^{EC} = High) \\
&= max(script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies) \\
&= script\_kiddies
\end{aligned}
$$

4.2. Brute-Force Attacks

The presence of two kinds of a brute-force attack is well visible in the correlated alerts: SSH and cross-site scripting. SSH brute-force attack happened in 6 min and manifested with one occurrence of a potential SSH scan alert and one occurrence suggesting a likely brute-force attack. Both these alerts originated from the same external IP address and targeted the same internal machine.

1. `ET SCAN Potential SSH Scan`
2. `ET SCAN LibSSH Based Frequent SSH Connections Likely BruteForce Attack`

Skill level evaluation of the first alert follows the same as in the GoldenEye example. m^{EC} is high because of the keyword scan in the message. For the second alert, all the metrics apart from m^{EC} are the same as there are no indications in the alert's message or the alert's rule that a user interaction must happen, or higher privileges are required, or any authentication was successful. However, there are no references to exploit codes, so automatically it would be assigned the value unproven—meaning that only a highly skilled attacker can execute such attack. This is, in fact, not so. There are SSH brute-force tools readily available, and so we re-evaluated the metric with value high. As a result, both alerts can be raised by script kiddies.

$$
\begin{aligned}
sl_{SSHScan}, sl_{LibSSH} \quad &= max(m^{AV} = Network, m^{UI} = None, m^{PR} = None, m^{Au} = None, m^{EC} = High) \\
&= max(script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies) \\
&= script\_kiddies
\end{aligned}
$$

A cross-site scripting brute-force attack was detected within a 40-min window with 3724 alerts, each within milliseconds of each other, indicating nothing else but the execution of an automated script to check a vulnerable website for the presence of a cross-site scripting vulnerability. The data set description of the attack states they implemented script in selenium framework to attack the vulnerable application. The attack was executed from one external IP address and targeted one internal IP address.

1. `3724x ET WEB_SERVER Script tag in URI Possible Cross Site Scripting Attempt`

This is an interesting case for an evaluation of the cross-site scripting (XSS) attempt, especially in terms of the determination of the exploit code maturity metric and user interaction metric. Let us start with the values that are straightforward to assess: The attack vector is network because of the external source IP address. It was a Web application attack, and there are no indications that a different privilege than none was needed or that any authentication had to be successful before the alert was raised. For many types of cross-site scripting, an interaction from a legitimate user in the form of clicking on a link is required. However, here we observed such small differences in the times and the number of XSS attempts was so high, that it could not be anything else but an automated brute-force attack. Therefore, user interaction was set to none. Concerning the exploit code maturity metric, the alert's rule included a reference to a list of XSS payloads, thereby providing evidence of a highly independent exploit code. The resulting skill level required to raise this alert is, therefore, script kiddies.

$$
\begin{aligned}
sl_{XSS} \quad &= max(m^{AV} = Network, m^{UI} = None, m^{PR} = None, m^{Au} = None, m^{EC} = High) \\
&= max(script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies) \\
&= script\_kiddies
\end{aligned}
$$

4.3. SQL Injection Attack

Execution of an SQL injection attack was detected with altogether five different types of alerts within a 9-min window. The first alert came from an external IP address targeting an internal Web server. Responses from the Web server were sent to the external IP address as well. The nature of the alert messages and the execution in less than 10 min suggests usage of an automated tool, such as `sqlmap`. In the first steps, we can see an attempt to get an error in the result to verify the possible presence of a vulnerability. Then there were various SQL injection payloads tried again, resulting in an error response. In the end, we see a clear success with the last of the payloads and retrieving data from the database.

1. `2x ET WEB_SERVER SQL Errors in HTTP 200 Response (error in your SQL syntax)`
2. `4x ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT`
3. `2x ET WEB_SERVER Possible Attempt to Get SQL Server Version in URI using SELECT VERSION`
4. `2x ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT`
5. `2x ET WEB_SERVER SQL Errors in HTTP 200 Response (error in your SQL syntax)`
6. `8x ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT`
7. `2x ET WEB_SERVER Possible MySQL SQLi Attempt Information Schema Access`
8. `2x ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM`
9. `2x ET WEB_SERVER SQL Errors in HTTP 200 Response (error in your SQL syntax)`
10. `2x ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT`
11. `2x ET WEB_SERVER Possible MySQL SQLi Attempt Information Schema Access`
12. `4x ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT`
13. `2x ET WEB_SERVER Possible MySQL SQLi Attempt Information Schema Access`
14. `2x ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM`
15. `2x ET WEB_SERVER ATTACKER SQLi - SELECT and Schema Columns`

All of these alerts have the same attributes when it comes to determining the values for the metrics. They all are communications between an external IP address and an internal IP address, so there is no

need to choose differently than the attack vector value network. There is also no evidence showing that a user interaction must take place, or that any user privilege must be acquired. The number of alerts and the short time window and regularity with which they were observed supports nothing else but the usage of an automated tool. As it is a Web application attack, we do not have enough evidence to support that any authentication must have taken place before the attack was executed. Therefore, we keep the default values for the metrics m^{UI}, m^{PR} and m^{Au}, and set m^{EC} to high to reflect that for SQL injections there are publicly available lists of payloads to try and tools that automate these checks. The final skill level evaluation needed to raise any of the observed SQL injection hyper-alerts is script kiddies.

$$
\begin{aligned}
sl_{SQL} \quad &= max(m^{AV} = Network, m^{UI} = None, m^{PR} = None, m^{Au} = None, m^{EC} = High) \\
&= max(script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies) \\
&= script\_kiddies
\end{aligned}
$$

4.4. Infiltration Attacks

There were three different infiltration techniques detected in the data: usage of a file via Dropbox, usage of Eternalblue exploit and usage of a trojan via Metasploit.

The first attack was detected within a 1-min window with two occurrences of the hyper-alert `ET POLICY Dropbox.com Offsite File Backup in Use`. After the user downloaded a file via Dropbox to their machine, alerts were detecting various types of scans from the infiltrated IP address, indicating that an attacker scanned the network. This corresponds to the description of the executed attack. The infiltration and scanning were detected within an 11-min window. Correlated alerts of infiltration attack 1 are:

1. 2x `ET POLICY Dropbox.com Offsite File Backup in Use`
2. `ET TROJAN Windows dir Microsoft Windows DOS prompt command exit OUTBOUND`
3. `ET SCAN Behavioral Unusual Port 135 traffic Potential Scan or Infection`
4. `ET SCAN Behavioral Unusual Port 445 traffic Potential Scan or Infection`

Alert #1 notifies that twice, a file was downloaded from a host in an external network subnet to an internal machine via the Dropbox client. Thus, the attack vector metric was set to network. There is an URL linked in references with a description of a proof of concept attack. Therefore, we can set exploit code maturity metric to proof of concept (POC). In the alert, there are no keywords or elements that would suggest choosing differently than default values for the privilege required and authentication metrics. However, by extracting information from the referenced URL, it is understood that the circumstances in which that alert was raised, were such that there had to be at-least-once authenticated access of a legitimate user via the Dropbox client. The second alert was observed 19 s after the first one. This is a wide enough gap that it probably was not done automatically, but rather that a legitimate user by their actions triggered the synchronization of their Dropbox folder. That would suggest we should set the user interaction metric to required. This and the proof of concept value for m^{EC} introduce mappings to moderately skilled attackers in the inputs for the maximum function. The overall skill level needed to raise this policy violation alert is therefore moderately skilled.

Alert #2 notified us that from an internal IP address there was outgoing traffic with characteristics of a trojan. As this is an autonomous malicious software activity, the exploit code maturity was set to high. Other metrics take their default values because there is no evidence showing otherwise in the alert's rule or references. This alert can, therefore, be triggered by script kiddies.

Alerts #3 and #4 notified us of scanning behavior from an internal IP address; only the destination port differed. This means that attack vector was set to adjacent. There was no user interaction, authentication or higher required privilege, so these metrics assumed their default values of none. Due to the presence of the keyword scan in the message, exploit code maturity was set to high.

The maximum function will return an overall skill level of moderately skilled needed to raise these two alerts.

$$
\begin{aligned}
sl_{Dropbox} \quad &= max(m^{AV} = Network, m^{UI} = Requir., m^{PR} = User, m^{Au} = Single, m^{EC} = POC) \\
&= max(script\_kiddies, moderate, script\_kiddies, script\_kiddies, moderate) \\
&= moderate \\
sl_{TrojanDOS} \quad &= max(m^{AV} = Network, m^{UI} = None, m^{PR} = None, m^{Au} = None, m^{EC} = High) \\
&= max(script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies) \\
&= script\_kiddies \\
sl_{scan\#2}, sl_{scan\#3} \quad &= max(m^{AV} = Adjacent, m^{UI} = None, m^{PR} = None, m^{Au} = None, m^{EC} = High) \\
&= max(moderate, script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies) \\
&= moderate
\end{aligned}
$$

The second attack shows exploitation of the Eternalblue vulnerability in Windows. The attack was executed over 12 min from one external IP address and targeted 37 machines in the victim network altogether. Eternalblue usage is not specified in the infiltration attack description in the data set.

1. `37x ET EXPLOIT ETERNALBLUE Exploit M2 MS17-010`

The alert detecting usage of the Eternalblue exploit contains in its rule a reference to the vulnerability CVE-2017-0143 that is targeted by this exploit code. Therefore, we will use the same values for our metrics as are defined in the CVSS score for this vulnerability. We use the CVSS string vectors from the National Vulnerability Database for this vulnerability, which are:

- CVSS:3.0/**AV:N**/AC:H/**PR:N**/**UI:N**/S:U/C:H/I:H/A:H
- AV:N/AC:M/**Au:N**/C:C/I:C/A:C

From these vectors, we use emphasized metric values. The exploit code maturity metric is missing in the base score, and we must evaluate it ourselves. Company Rapid7, which maintains the exploits in the tool Metasploit, contains in their database an entry marked with the vulnerability's CVE identification. Therefore, there exists a functional exploit code, but there is no evidence that it is ready to use as-is; rather, it must be configured with respect to the targeted network or hosts. We will set the m^{EC} metric to functional, which maps to moderately skilled attackers. As moderately skilled is a higher level than script kiddies, the maximum function will return moderately skilled.

$$
\begin{aligned}
sl_{Eternalblue} \quad &= max(m^{AV} = Network, m^{UI} = None, m^{PR} = None, m^{Au} = None, m^{EC} = Functional) \\
&= max(script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies, moderate) \\
&= moderate
\end{aligned}
$$

The third attack was detected by alerts that aggregate to just one hyper-alert. The data set's description does not provide further details on what type of exploit was executed with the Metasploit framework. We also do not see whether the infiltration through this technique was successful. The same as after the first attack, here we also see scans originating from the infiltrated internal IP address. The attack was detected within a 10-min window. The infiltration payload originated from an external IP address. The infiltrated IP address then became the source from which the attacker scanned the network.

1. `ET TROJAN Possible Metasploit Payload Common Construct Bind_API (from server)`
2. `ET SCAN Behavioral Unusual Port 139 traffic Potential Scan or Infection`
3. `ET SCAN Behavioral Unusual Port 1434 traffic Potential Scan or Infection`

Alert #1 suggests a possible trojan payload coming through Metasploit tool from an external IP address. Due to the presence of that keyword in the message, the exploit code maturity was set to high. Other metrics take their default values, because there is no evidence showing otherwise in the alert's rule or references. The skill level needed to trigger this alert is therefore script kiddies. Alerts #2 and #3 are the same scanning alerts as in the first infiltration attack.

$$
\begin{aligned}
sl_{Trojan} \quad &= max(m^{AV} = Network, m^{UI} = None, m^{PR} = None, m^{Au} = None, m^{EC} = High) \\
&= max(script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies, script\_kiddies) \\
&= script\_kiddies
\end{aligned}
$$

5. Conclusions

Security operation centers monitor the activity in an organization's network and use correlation methods to create attack paths providing more comprehensive information for situational awareness. These multi-stage attacks are further evaluated and prioritized with respect to various properties.

In this paper, we presented a set of rules for correlating alerts from intrusion detection systems into more meaningful attack steps to help reduce noisy alerts that take away the focus of security analysts. Furthermore, we presented a revised methodology for evaluating the skill level of these IDS alerts. The evaluation of an attacker's skill level is a useful metric for prioritization and situational awareness. In addition to the impact-based metrics, analysts can focus on "breaking points," i.e., alerts with suddenly higher skill level evaluations. The appearance of such an alert signals that the attacker demonstrated a high-level skill set and analysts should focus their efforts on preparing appropriate countermeasures for other possible targets. In this paper, we evaluated each hyper-alert individually with such use cases in mind and did not summarize one skill level evaluation for the full correlated path.

There are open questions as to how to design the methodology for such a summation of skill level for the entire correlated path. In a real-time scenario, the path is never fully complete, and it is more useful to evaluate individual alerts as they come and reconsider potential targets of the attack based on the newly detected actions. For example, in the example scenario where first an infiltration takes place, and then the attacker uses the compromised machine to execute scans of the internal network, the scanning itself is not inherently more complicated than the infiltration; therefore, it should not increase the skill level needed for the whole attack.

The CSE-CIC-IDS2018 data set [12] was intended for verifying anomaly-based detection methods and the executed attacks were single atomic attacks. Even though one or two show characteristics of multi-stage attacks, they are only the early stages of such attacks. Therefore a more comprehensive data set is needed for more robust verification of methods. It must be designed with the intention of capturing multi-stage attacks that simulate advanced persistent threats in the network in a more realistic manner.

References

1. Mézešová, T.; Sokol, P.; Bajtoš, T. Evaluation of Attacker Skill Level for Multi-stage Attacks. In Proceedings of the 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 27–29 June 2019; pp. 1–6.

2. Paulauskas, N.; Garsva, E. Attacker skill level distribution estimation in the system mean time-to-compromise. In Proceedings of the 2008 1st International Conference on Information Technology, Gdansk, Poland, 18–21 May 2008; pp. 1–4.

3. Hu, H.; Liu, Y.; Zhang, H.; Zhang, Y. Security metric methods for network multistep attacks using AMC and big data correlation analysis. *Secur. Commun. Netw.* **2018**, *2018*. [CrossRef]

4. ben Othmane, L.; Ranchal, R.; Fernando, R.; Bhargava, B.; Bodden, E. Incorporating attacker capabilities in risk estimation and mitigation. *Comput. Secur.* **2015**, *51*, 41–61. [CrossRef]

5. van Rensburg, A.J.; Nurse, J.R.; Goldsmith, M. Attacker-parametrised attack graphs. In Proceedings of the Tenth International Conference on Emerging Security Information, Systems and Technologies, Nice, France, 24–28 July 2016.

6. Rocchetto, M.; Tippenhauer, N.O. On attacker models and profiles for cyber-physical systems. In Proceedings of the European Symposium on Research in Computer Security, Heraklion, Greece, 26–30 September 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 427–449.

7. Fraunholz, D.; Anton, S.D.; Schotten, H.D. Introducing gamfis: A generic attacker model for information security. In Proceedings of the 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 21–23 September 2017; pp. 1–6.

8. Ošt'ádal, R.; Švenda, P.; Matyáš, V. Reconsidering attacker models in Ad-Hoc networks. In Proceedings of the Cambridge International Workshop on Security Protocols, Brno, Czech Republic, 7–8 April 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 219–227.

9. Krautsevich, L.; Martinelli, F.; Yautsiukhin, A. Towards modelling adaptive attacker's behaviour. In Proceedings of the International Symposium on Foundations and Practice of Security, Montreal, QC, Canada, 25–26 October 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 357–364.

10. Mohammadian, M. Intelligent security and risk analysis in network systems. In Proceedings of the 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), Dubai, UAE, 18–20 December 2017; pp. 826–830.

11. Hassan, S.; Guha, R. A probabilistic study on the relationship of deceptions and attacker skills. In Proceedings of the 2017 IEEE 15th International Conference on Dependable, Autonomic and Secure Computing, 15th International Conference on Pervasive Intelligence and Computing, 3rd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Orlando, FL, USA, 6–10 November 2017; pp. 693–698.

12. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), Funchal, Portugal, 22–24 January 2020; pp. 108–116.

13. Gamage, S.; Samarabandu, J. Deep learning methods in network intrusion detection: A survey and an objective comparison. *J. Netw. Comput. Appl.* **2020**, *169*, 102767. [CrossRef]

14. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]

15. de Souza, C.A.; Westphall, C.B.; Machado, R.B.; Sobral, J.B.M.; dos Santos Vieira, G. Hybrid approach to intrusion detection in fog-based IoT environments. *Comput. Netw.* **2020**, *180*, 107417. [CrossRef]

16. Mézešová, T.; Bahsi, H. Expert Knowledge Elicitation for Skill Level Categorization of Attack Paths. In Proceedings of the 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Oxford, UK, 3–4 June 2019; pp. 1–8.

17. Mell, P.; Scarfone, K.; Sasha, R. A Complete Guide to the Common Vulnerability Scoring System Version 2.0. Published by FIRST-Forum of Incident Response and Security Teams, 2007. Available online: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198 (accessed on 19 November 2020).

18. CVSS Special Interest Group. Common Vulnerability Scoring System v3.1: Specification Document; 2019. Available online: https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf (accessed on 19 November 2020).

19. Andel, T.R.; Yasinsac, A. Adaptive threat modeling for secure ad hoc routing protocols. *Electron. Notes Theor. Comput. Sci.* **2008**, *197*, 3–14. [CrossRef]

20. Allodi, L.; Banescu, S.; Femmer, H.; Beckers, K. Identifying relevant information cues for vulnerability assessment using CVSS. In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, Tempe, AZ, USA, 19–21 March 2018; pp. 119–126.

21. Elbaz, C.; Rilling, L.; Morin, C. Fighting N-day vulnerabilities with automated CVSS vector prediction at disclosure. In Proceedings of the 15th International Conference on Availability, Reliability and Security, Dublin, Ireland, 25–28 August 2020; pp. 1–10.

22. Watters, P.; Scolyer-Gray, P.; Kayes, A.; Chowdhury, M.J.M. This would work perfectly if it weren't for all the humans: Two factor authentication in late modern societies. *First Monday* **2019**, *24.*. [CrossRef]

23. Kayes, A.; Rahayu, W.; Watters, P.; Alazab, M.; Dillon, T.; Chang, E. Achieving security scalability and flexibility using Fog-Based Context-Aware Access Control. *Future Gener. Comput. Syst.* **2020**, *107*, 307–323. [CrossRef]

24. ET Labs. *Emerging Threats Rules*; ET Labs: Austin, TX, USA, 2020.

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.