Design and Implementation of an Availability Scoring System for Cyber Defence Exercises

Mauno Pihelgas

This paper was accepted to the 14th International Conference on Cyber Warfare and Security: ICCWS 2019 and the final version of the paper is included in the Conference Proceedings (ISBN: 978-1-912764-11-2; ISSN: 2048-9870).

Design and Implementation of an Availability Scoring System for Cyber Defence Exercises

Mauno Pihelgas
NATO Cooperative Cyber Defence Centre of Excellence, Tallinn, Estonia
Tallinn University of Technology, Tallinn, Estonia
firstname.lastname@ccdcoe.org

Abstract: Cyber defence exercises are crucial for training readiness and awareness within the *cyber domain*. This new domain is acknowledged by NATO, alongside land, sea, air and space. Alliance nations are endorsing the development of both defensive and responsive cyber capabilities. This paper discusses designing and building a reliable availability scoring system for a large international cyber defence exercise, Locked Shields. The system provides essential input for scoring the exercise participants to spark some friendly competition and motivate players. The previous solution was replaced by a modular setup that is built around a well-known open-source IT monitoring software called Nagios Core. Before embarking to develop a new system, we studied available research and looked at various other CDXs for similar implementations. Unfortunately, we did not find such full-blown scoring systems in use at the time. At least not according to the information that was provided to us. We therefore relied on best practices and prior experience to develop an automated availability scoring system. The paper provides some background information on the exercise, describes the requirements, design process and implementation of the scoring solution. The current system has been under continuous improvement since 2014 and has successfully provided the automated scoring checks for the past five exercises. In addition to success stories, several issues and problem workarounds are addressed. As such, this paper serves as a valuable resource for cyber defence exercise managers and practitioners looking to implement similar scoring solutions.

Keywords: availability, cyber exercise, monitoring, Nagios, scoring, Selenium

1. Introduction

The field of cyber defence has become increasingly important over the years and numerous cyber defence exercises (CDX) are being organised all over the world. One of the largest is Locked Shields (LS), which is a game-based real-time network defence exercise that has been organised by the NATO Cooperative Cyber Defence Centre of Excellence since 2010 (NATO Cooperative Cyber Defence Centre of Excellence 2018). Measuring and scoring the performance of the training audience (Blue Teams) is essential to ensure that everyone gives their best during the competitive exercise.

The primary focus of this paper is on the process of measuring the availability of individual services provided by Blue Team (BT) systems and passing the data to the overall scoreboard (see Figure 1). This *scoreboard* is another system, which combines all the different sub-scores into a combined score for a BT; however, this does not fall within the scope of this paper and is not to be confused with the availability scoring system. Although this paper focuses only on the LS implementation, the same scoring system could be easily implemented for other similar CDXs.

1.1 Exercise overview

The LS exercise spans two days and is built as a competitive game featuring a fictional scenario in which the defending BTs are scored on their performance in several different interdisciplinary categories, such as defending against technical cyber attacks, incident reporting, situation reporting, responding to scenario injects and keeping their systems available to the users. Scoring is an integral part of the game, because participants need to know how well they performed in the tasks set for them and compared to other teams. The final scoreboard ranking decides which team wins the annual LS exercise.

Figure 1: Example scoreboard combining all different score types

Participants represent one of five different roles: Blue Team (defence), Red Team (*bad guys*), Yellow Team (situation awareness), Green Team (exercise infrastructure), and White Team (exercise management, strategic gameplay, media, etc.). According to the scenario, the BTs assume the role of rapid-reaction teams assisting the fictional country of Berylia, which is in conflict with another fictional country, Crimsonia, represented by the Red Team (RT). The exercise takes place over just two days (8 hours per day) of intense game-play.

In 2018, each of the 22 BTs was responsible for maintaining the continuous and secure operation of 140 hosts with several monitored services on each host, amounting to 1,425 monitored services per team to calculate the availability of each required service. The notion of availability is represented as an uptime value between 0 and 1 for a particular service. This value could be represented as a Service Level Agreement (SLA) percentage for easier interpretation. The scoring system keeps track of all check results and calculates the uptime for each service.

The availability score accounts for approximately ⅓ of the total positive score points in the exercise, which is roughly the same as the amount of negative score that teams may get for successful attacks carried out by the RT. Therefore, teams need to balance their strategy between security and availability of services wisely, because the function for calculation of points lost due to downtime is exponential, not linear (discussed in section 3).

The gamenet infrastructure features a wide variety of traditional IT infrastructure and special-purpose industrial systems: Windows, Linux, and FreeBSD hosts, Siemens Programmable Logic Controllers (PLCs), Threod Flight Controllers and Ericsson Virtual EPC Packet Gateways for 4G/LTE connectivity. In addition to system administration and hardening tasks, teams are also faced with forensic and legal challenges and various injects from the game's media team. This means that the defending BTs must include specialists with very different skills to be able to field all the required expertise.

1.2 Problem and motivation

The development of this availability scoring system was motivated by several deciding factors. At the beginning of 2014 the author of the previous system left the LS organising team. The old system turned out to be a monolithic Perl script that contained all the information on what services to check and the logic for every single availability check. We have nothing against implementing solutions in Perl language, but the script was lacking some of the widely accepted best practices in software engineering (e.g., it is recommended to have a modular

design where smaller and simpler interconnected subsystems communicate) (Seacord 2018). Thus, the old system was difficult to understand and develop.

Primary concerns were the stability and scalability of the system. There were issues with high memory consumption with all the various libraries required for checking the BT services constantly loaded in the process. We decided to replace the previous system with a new and improved solution that, among other aspects, would provide a modular design and increased stability.

1.3 **Outline**

The remainder of this paper is organised as follows, section 2 describes the requirements of the new system, section 3 introduces the basic functionality the monitoring solution, section 4 presents the current implementation of the availability scoring, section 5 provides a brief insight into future work, section 6 gives an overview of related work, and section 7 concludes the paper.

2. Requirements

When we initially set out to design the new scoring system, we had several requirements due to the nature of the LS exercise and several of our own ideas and best practices on what is required of an exercise scoring system. The following list briefly describes the requirements and the reasoning behind the specific item.

- Stable and reliable: remain operational even in situations where the underlying network or related systems may temporarily fail.
- Active checks: the queries towards BT systems have to be initiated by the scoring server to avoid any manipulation of passive check results.
- Predictable: each BT should receive the same number of requests (checks) from the scoring engine.
- Modular: easy to add new functionality or modify existing modules, also availability of existing modules in the community.
- Customisable: since the CDX is not a regular IT environment, we do not need all the functionality offered by many monitoring tools (e.g., problem escalation, help-desk functionality, etc.).
- Integration: ability to easily integrate the solution with other pre-existing systems.
- Scalable: ability to monitor thousands of hosts and services with a short check interval.
- Well-known and documented: avoid relying solely on a single person to configure the system.
- Evasive tactics: implement various tactics to avoid being easily identified as the scoring engine and thus allowing the system to be whitelisted by the BTs.
  - Randomise IP addresses: periodically change IP addresses on selected network interfaces.
  - Randomise check intervals: periodically change check intervals to avoid creating a recognisable pattern in log files.
  - Mimic regular systems: remove any identifiers of the monitoring software.
- Preferably open-source software: we might need to modify the source code of various components (especially to implement the evasive tactics mentioned above).

It was evident that most of the first level requirements listed above could be addressed by using some well-known monitoring solutions that are already available. We tested several different solutions, such as Zabbix, Shinken, Opsview, Nagios XI, Nagios Core, op5 and Centreon. Most of the solutions offered similar functionality and were able to satisfy many or even most of the requirements but were often too overloaded with complex functionality (e.g., helpdesk and incident handling functionality) which we did not need in our situation. After testing several strong contenders in our lab environment, we decided to use Nagios Core as a central element of the new availability scoring system for LS.

However, there was a narrower set of requirements regarding various evasive tactics that none of the compared solutions was able to solve *out of the box*. Typical monitoring solutions are not evasive and work in a very recognisable and predictable manner. For instance, checks usually come from a single known IP address that is always granted access through the firewall, the time between the checks (check intervals) is usually fixed at regular intervals (e.g., 1 minute or 5 minutes), and by default the system would identify itself (e.g., in the web browser user-agent string) as the monitoring system. To avoid this issue, additional functionality had to be developed separately.

3. Basics of availability scoring

Typical monitoring systems actively check and report the state – (*OK*, *Warning*, *Critical*, or *Unknown*) – of the monitored services. There is a predefined list of services BTs are supposed to keep up and running for the entire duration of the two-day exercise. Any disruptions in the services will result in lost uptime and valuable points.

Active checks are implemented by executing various user-defined monitoring scripts that perform the necessary steps. For example, checking the availability of a web page requires the script to establish a connection to the web server hosting the site and download the content of the page for any further inspection (e.g., verifying the presence of correct content). Active checks are initiated by the scoring server and BTs have no control over when the check is executed or what is being checked within the logic of the script (see Figure 2).

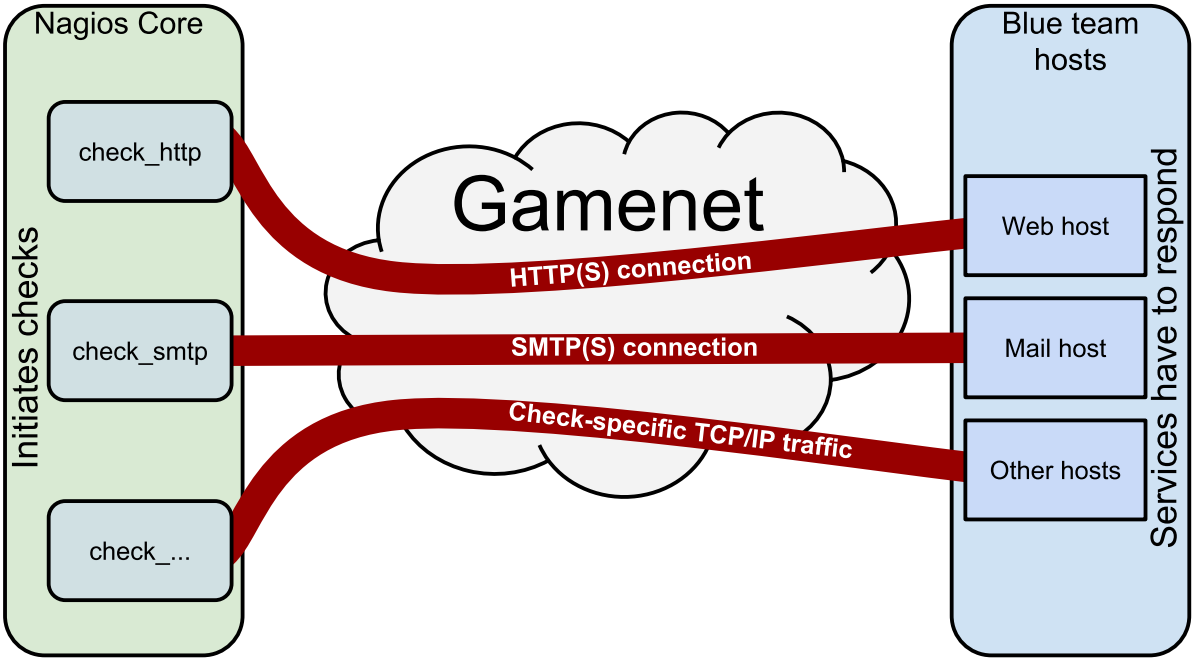

Figure 2: Basic active check process

Alternatively, passive service checks could be used in monitored systems to report the current state of the system back to the central monitoring server. For instance, in our previous example the web server would periodically execute a self-check and send the result to the scoring server. Although this is the preferred method in many standard monitoring solutions, it does not fit well with our exercise scenario, because it is in the interest of the competing BTs to portray as good an uptime as possible. It can be tempting for teams to stop sending genuine check results and send forged passive check results to our scoring server instead. We therefore prefer active checks, because they are more reliable in the adverse situations of the competitive exercise.

Listing 1 illustrates the output produced by these service checks. The output format was largely taken from the previous scoring system and it fits the purpose nicely since the vertical bar symbol is a special character in Nagios to distinguish the message from check performance data and thus cannot accidentally appear in our output.

Listing 1: Example output from active service checks

```
mail.blue05.ex|http|OK|HTTP OK: HTTP/1.1 200 OK - 322 bytes in 0.005 second response time
|1524721248|1524721248

hmi.pgc.blue13.ex|http.ipv6|OK|HTTP OK: HTTP/1.1 200 OK - 954 bytes in 0.014 second response
time|1524721248|1524721249

ws4-02.int.blue15.ex|ssh|CRITICAL|No route to host|1524721244|1524721249
```

The BTs can gather up to 20,000 points (about ⅓ of total positive score) for the availability of services. Individual services are distributed into various sub-groups and are assigned an arbitrary weight value (W) based on their significance. Any loss of uptime (U) will affect the score coefficient (SC) exponentially:

$$SC=W^{U-1},\ 0\leq U\leq 1$$

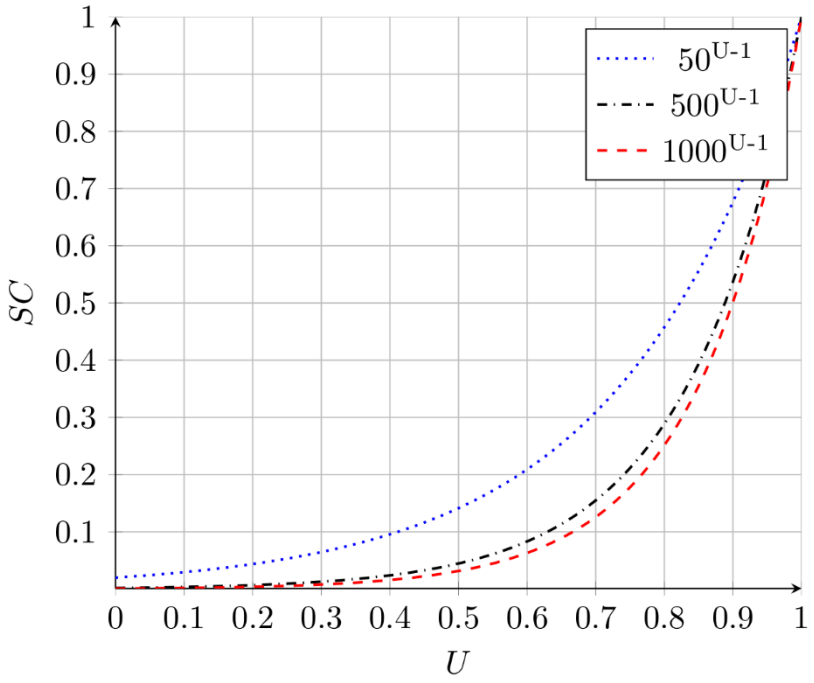

Figure 3: Score coefficient (SC) with regards to uptime (U) and three example service weight values (W)

The rationale for this harsh score penalty is that in the IT field, production system uptime is commonly expected to be in the order of 99.9% or in some cases even higher, which, for example, means that a service can be down for just about 10 minutes per week or 1.44 minutes per day. Evidently, the higher the service weight value, the steeper the score penalty for lost uptime (see Figure 3). We calculate the resulting score points according to W×SC. For instance, a service with a relatively high weight value (W) of 1,000 points will only yield 500 score points if its uptime is 90%. This is because its score coefficient (SC) quickly decreases to approximately 0.5 if the uptime (U) drops to 0.9.

Throughout the game execution, runtime scores are added up and sent to the live scoreboard. An example scoreboard can be seen in Figure 1. The availability score discussed in this section is represented as *sla score* in the figure.

4. Implementation

This section describes the hardware and software specification that the solution was deployed on in 2018, followed by a description of how the evasive techniques mentioned in section 2 have been implemented.

4.1 System description

There were five virtual machines (VM) assigned for the availability scoring system. Each VM was assigned 12 logical processors, 8GB of memory, and 60GB of disk space. We could have managed with fewer VMs, but the main requirement was to attain different vantage points (e.g., BT internal or external) in the gamenet infrastructure. For instance, depending on the type of BT service (e.g., public or private), the availability checks must be performed from a BT internal network, from an external network, or in some cases from both.

Thus, on each scoring machine, two network interfaces provided Green Team management and generic internet access, while the third network interface provided the unique network vantage point: 4G/LTE network, RT network, BT internal networks, or BT management networks.

To monitor all the required internal BT assets, we had to connect the scoring server to five different /24 subnets for every BT. This added up to 132 virtual network interfaces connected on one of the scoring servers, which is quite uncommon and created some issues that had to be overcome in the operating system configuration.

4.2 **Nagios**

In 2018, the scoring system was built on a Debian GNU/Linux 9.4 operating system using Nagios Core v4.3.4 (latest stable version at that time) which we compiled from source code. We also modified and compiled the standard nagios-plugins package (version 2.2.1) containing the check scripts provided by Nagios Developers (section 4.4 discusses the motivation behind these modifications).

We applied several of the steps outlined in the Nagios Performance Tuning guide (Nagios Enterprises 2018a). We made extensive use of the RAM Disk setup recommended by Nagios Enterprises (2018b) to minimise the number of slow disk I/O operations and keep frequently accessed files in memory. In addition to Nagios-specific files, we also used it to store other temporary files required by scripts and plugins.

4.3 **Performance assessment**

The virtualised systems worked well. System load was mostly below the critical threshold (i.e., equal to the number of CPU threads), and although there were critical peaks when restarting Nagios, these passed momentarily. In terms of memory use, the 8GB assigned to the hosts was plenty; on average the systems used about 3-4GB.

For each BT, we performed scoring checks for 140 hosts providing a total of 1,425 services on both IPv4 and IPv6. Multiply that by the number participating teams (22), and we get an impressive 3,080 hosts with 31,350 services being checked at least once per minute. During the 16 hours of game-play, 34,025,305 individual scoring checks were performed and logged. This averages at about 35,443 checks per minute.

4.4 **Evasive techniques**

The requirements introduced several situations where we needed to avoid the availability scoring system being identified by BTs. They could still attempt to identify the scoring server by making an educated guess based on the requests made by the scripts checking the services, but we tried to make it more difficult for them. To clarify why this is important, consider a situation in which the BTs were able to successfully identify the IP addresses from which the scoring server is accessing their services. They could use a whitelisting approach in the firewall and always allow access for the scoring server, but block everyone else, including the RT, from accessing the service. The scoring server would see that everything is up and functional, while no other users would be able to use the service. We describe some of the more important evasive techniques below.

4.4.1 *Randomising IP addresses*

The primary way to evade identification was to regularly change the IP addresses of the scoring server. We configured the server to change all its public IP addresses that were used to communicate with BT services from the RT networks. The change was scheduled at a random interval ranging between 5 to 10 minutes. To safeguard against IP conflicts, we used the arping utility for IPv4 and ndisc6 utility for IPv6 to verify that an IP address was available before assigning it to the interface.

Since changing an IP address could disrupt an ongoing service check, we needed to shut down Nagios before replacing any addresses. Typically, the entire process took about 7-8 seconds. This was a critical moment, because the system can end up in a state where it no longer has connectivity to the gamenet. To make sure that the new configuration was fully correct and operational, we introduced a series of connectivity tests. If the tests fail, the system reverts to its default static IP addresses, which should always work.

4.4.2 *Randomising check intervals*

For normal monitoring applications, a fixed checking interval is a sensible idea. However, in our case the BTs could predict, with an accuracy of one second, when the scoring system would perform the next availability check. This look like a heartbeat on log monitoring tools and dashboards. Inspect the timestamps in Listing 2 to see how easy it is to detect a system with a fixed check interval (e.g., 60 seconds). To counter this, we developed a simple script that would randomise the check interval within a given range between 40 and 55 seconds. Since any changes in check interval would only take effect after restarting the Nagios software, we combined this procedure with changing IP addresses. Observe the access timestamps in Listing 3 after applying the randomised check interval.

Listing 2: Typical unmodified Nagios checks examined from a BT web server access log

```
10.0.228.111 - - [23/April/2018:06:39:58 +0000] "GET / HTTP/1.1" 200 437 "-"
"check_http/v2.2.1 (nagios-plugins 2.2.1)"

10.0.228.111 - - [23/April/2018:06:40:58 +0000] "GET / HTTP/1.1" 200 437 "-"
"check_http/v2.2.1 (nagios-plugins 2.2.1)"

10.0.228.111 - - [23/April/2018:06:41:58 +0000] "GET / HTTP/1.1" 200 437 "-"
"check_http/v2.2.1 (nagios-plugins 2.2.1)"
```

Listing 3: Example of a web server log entry created by a modified Nagios check_http script

```
10.0.228.111 - - [23/April/2018:06:45:43 +0000] "GET / HTTP/1.1" 200 437 "-" "Mozilla/5.0
(X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"

10.0.228.111 - - [23/April/2018:06:46:25 +0000] "GET / HTTP/1.1" 200 437 "-"
"Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
```

4.4.3 *Identification strings*

We discovered that default Nagios plugins such as *check_http, check_ssh, check_smtp* clearly identify themselves to the service they are communicating with. Log lines in Listing 2 reveal that the check_http script sends the BT web server the default user-agent string. Under normal circumstances this is good behaviour and helps anyone looking into the logs to quickly distinguish events created by the monitoring system. In our case, however, this would easily allow the BTs to identify and whitelist the scoring server's IP address. To address this, we modified several monitoring scripts to replace these identifiable strings with software versions that would exist in a regular Linux desktop computer. See Listing 3 for a log entry created by a modified version of a monitoring script using the Mozilla Firefox user-agent string.

4.4.4 *Countering BT tactics*

Over the years, we have seen BTs find ways to improve their score by trying to trick the scoring system into thinking their services are up and running when in fact they have just set up a dummy service or a similar-looking static website.

For instance, some teams have set up tools such as netcat or Portspoof that bring up fake services on ports and respond to any incoming requests. Therefore, it is essential that monitoring plugins perform more in-depth functionality checks to verify that there is in fact a correct service responding on the expected port.

Another questionable tactic has been to take an entire web site and convert it into static HTML pages. Visually the site seems normal, but any dynamic content and integration with a database has been eliminated. We have used the open-source Selenium WebDriver, a testing framework for web applications to counter this. Using Selenium allows us to write our test cases in Python for automating web browsing. For example, mimicking a customer visiting the page of an online shop: logging in, browsing a few items and adding some of them to the shopping cart, then finalising the purchase by checking out and verifying that the purchase was indeed saved under previous orders. Such test cases provide an excellent way of checking that essential functionality is present, but due to their complexity they are difficult to debug if the test case breaks and erroneously flags the site as broken.

5. Future work

Since we have been collecting exercise data such as scoreboard results, network PCAPs, scoring logs and RT activity logs for the past five years, we have gathered a substantial dataset for future research.

Our first idea is to analyse the success rate of different strategies used by BTs. For example, some teams focus more on providing excellent uptime at the expense of security, while others completely lock down their systems to protect them against the RT, but likely block the scoring system in the process. Having the training dataset, we could develop a model that would predict the score of the team based on their strategy. This could also reveal combinations of strategies that are more likely to succeed than others.

Additionally, we aim to develop a system that would better detect cheating among the participants. For instance, we should verify that the availability scoring results correspond to the RT reports. There should be a strong

correlation between the BT systems going offline after successful attacks and RT attacks failing when the BT system was already unavailable before the attack. Finding anomalies (e.g., change of expected values or hashes) in the responses from the BT systems would allow us to discover and sanction (negatively score) the excessive use of questionable tactics (see subsection 4.4.4 above) during the game. Currently all this requires manual analysis.

6. Related work

LS is by no means the only cyber defence exercise in the world. There are several different cyber exercises conducted around the globe. The *NSA Cyber Exercise* (2018) (formerly Annual Cyber Defense Exercise) hosted by the National Security Agency and *Cyber Shield 2018* (2018) hosted by the U.S. Army National Guard are probably the most similar to Locked Shields.

Other relevant exercises include *Cyber Security Awareness Week (CSAW)* (2018) challenges, *Cyber Security Challenge UK* (2018), *DEF CON Capture the Flag Contest* (2018), *Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC)* (2018), *National Collegiate Cyber Defense Competition (NCCDC)* (2018), *National Cyber League (NCL)* (2018) competition, and *International Capture The Flag (iCTF)* (2018) exercise. One of the most recent additions to this list are *off-the-shelf* technical exercises provided by CybExer Technologies (2018).

With such a variety of annual events, one should also distinguish different types of cyber exercises: security challenges, capture-the-flag contests, or exercises that focus on defence. Often, there is no clear classification, but it is still worthwhile to consider what is the primary goal of the exercise. In the case of LS, training to defend various systems and networks is the main objective and BTs are the training audience.

Unfortunately, there is a limited amount of material available on the subject. Most of the sources listed above are just limited to a homepage briefly describing the past exercises from a relatively high level. This is especially the case with military exercises. The remainder of this section discusses several more elaborate sources about previous work on cyber exercises, especially the ones that mention various scoring or assessment systems.

A paper by Patriciu & Furtuna (2009) presents practical guidelines to follow when designing a new cyber security exercise. The paper aims to assist CDX newcomers who oversee designing a cyber security exercise. The step-by-step instructions cover topics such as defining the objectives, designing the network topology, creating a scenario, establishing rules, choosing the appropriate metrics, and conducting a lessons-learned procedure. The guideline has a rather brief section on recommendations for the scoring engine, stating only that there must be a clear set of rules to express the way points can be obtained or lost. Moreover, the whole scoring process has to be transparent to all participants.

Papers from Hammervik et al. (2010) and Granåsen et al. (2011) try to capture and analyse the data from the Baltic Cyber Shield exercise, which is in fact a direct precursor of LS. They address whether the vulnerability of a host influences the time required to compromise it, and whether cyber security professionals can predict the success rates of arbitrary code execution attacks.

The work by Werther et al. (2011) shares the experience from conducting capture-the-flag exercises in the MIT Lincoln Laboratory. Regarding scoring, the solution is quite like ours in LS. The scores are calculated as a weighted average of availability, confidentiality, integrity, and offence. The exercise has a scoreboard where the overall score of all participating teams is displayed. Like LS, the first 30 minutes of the exercise is not scored to allow teams to apply patches and secure their machines. However, unlike LS, attacks are allowed during this period.

A paper by Henshel et al. (2016) describes using the Cyber Shield 2015 exercise to develop the assessment model and integrated evaluation of team proficiency metrics in CDXs. In addition to using exercise data, they also conducted an expertise survey before the event to determine potential relationships between prior expertise and performance in the exercise. For future work, they stress that near real-time analysis of the exercise data is required. They conclude that raw data collection is not an issue, but the capability to manually analyse the information does not scale with the huge amounts of incoming data. Their aims and observations closely coincide with ours, thus we have already contacted the authors of this paper and will engage in future cooperation.

7. Conclusion

There is a growing trend of organising cyber defence exercises, which is not an easy task. In addition to difficulties in finding people who can develop the scenario and build up the required infrastructure to provide an exercise

on this scale, motivating the participants to give of their best is a challenge. We have found that adding a competition to the exercise is beneficial and serves this purpose well. The competition can be scored based on the performance of trainees by checking how well they keep their systems up and functional.

This paper has presented an overview of the availability scoring system for the Locked Shields cyber defence exercise. It has discussed the design and practical implementation of the system and listed some key observations from the experience gained during the past five years of development.

8. Acknowledgements

This work has been supported by the Estonian IT Academy (StudyITin.ee).

References

Cyber Security Awareness Week (CSAW) (2018), Available: https://csaw.isis.poly.edu/.

Cyber Security Challenge UK (2018), Available: http://cybersecuritychallenge.org.uk/.

Cyber Shield 2018 (2018), Available: http://www.cs18.org/.

CybExer Technologies (2018), 'Technical Exercises', Available: https://cybexer.com/.

DEF CON Capture the Flag Contest (2018), Available: https://www.defcon.org/html/links/dc-ctf.html.

Granåsen, D., Granåsen, M., Sundmark, T., Holm, H. & Hallberg, J. (2011), Analysis of a Cyber Defense Exercise using Exploratory Sequential Data Analysis, The 16th International Command and Control Research and Technology Symposium (ICCRTS).

Hammervik, M., Granåsen, D. & Hallberg, J. (2010), Capturing a Cyber Defence Exercise, The 1st National Symposium on Technology and Methodology for Security and Crisis Management.

Henshel, D. S., Deckard, G. M., Lufkin, B., Buchler, N., Hoffman, B., Rajivan, P. & Collman, S. (2016), Predicting proficiency in cyber defense team exercises, *in* 'MILCOM 2016 - 2016 IEEE Military Communications Conference', pp. 776–781.

International Capture The Flag (iCTF) (2018), Available: http://ictf.cs.ucsb.edu/.

Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC) (2018), Available: http://maccdc.org/.

Nagios Enterprises (2018*a*), 'Tuning Nagios For Maximum Performance', Available: https://assets. nagios.com/downloads/nagioscore/docs/nagioscore/4/en/tuning.html.

Nagios Enterprises (2018*b*), 'Utilizing a RAM Disk in Nagios XI', Available: https://assets.nagios.com/ downloads/nagiosxi/docs/Utilizing A RAM Disk In NagiosXI.pdf.

National Collegiate Cyber Defense Competition (NCCDC) (2018), Available: http://www.nationalccdc. org/.

National Cyber League (NCL) (2018), Available: https://www.nationalcyberleague.org/.

NATO Cooperative Cyber Defence Centre of Excellence (2018), 'Locked Shields 2018', Available: https: //www.youtube.com/watch?v=meC8O9Mptz4.

NSA Cyber Exercise (2018), Available: https://www.nsa.gov/what-we-do/cybersecurity/ncx/.

Patriciu, V.-V. & Furtuna, A. C. (2009), Guide for Designing Cyber Security Exercises, *in* 'Proceedings of the 8th WSEAS International Conference on E-Activities and Information Security and Privacy', E-ACTIVITIES'09/ISP'09, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, pp. 172–177.

Seacord, R. (2018), 'Top 10 Secure Coding Practices', Available: https://wiki.sei.cmu.edu/confluence/ display/seccode/Top+10+Secure+Coding+Practices.

Werther, J., Zhivich, M., Leek, T. & Zeldovich, N. (2011), Experiences in Cyber Security Education: The MIT Lincoln Laboratory Capture-the-flag Exercise, *in* 'Proceedings of the 4th Conference on Cyber Security Experimentation and Test', CSET'11, USENIX Association, Berkeley, CA, USA, pp. 12–12.