TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science

ITV70LT
Erki Naumanis 121996IVCMM

CENTRALLY MANAGED NETWORK TRAFFIC GENERATION FOR CYBER SECURITY EXERCISES

Master's thesis

Supervisor: Margus Ernits
MSc

Tallinn 2014

Author's Declaration

I declare that this thesis is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Erki Naumanis

June 3, 2014 ……………………

(Signature)

Annotatsioon - Keskhallatav võrguliikluse genereerimise keskkond küber-õppuste tarbeks

Käesolev magistritöö keskendub küber-õppustel esinevale pakilisele probleemile, piisava liikluse olemasolule suletud süsteemis. Töö käigus loodud rakendus lahendas probleemi: suletud keskkonnas on vähe liiklust ning ründav võrguliiklus on kergesti tuvastatav.

Probleemi lahendamiseks analüüsiti olemasolevaid võimalusi liikluse genereerimiseks vabavaraliste vahenditega. Püstitati nõuded liikluse genereerimiseks, mis oleks kooskõlas õppusel oodatud mahtudega. Seejärel pakuti välja lahendus, mida kasutades on võimalik võrguliiklust tekitada.

Töö tulemusena valmis töövahendi prototüüp, mida on võimalik küber-õppustel kasutada võrguliikluse genereerimiseks. Piisava hulga liikluse korral on kaitsvatel meeskondadel keerulisem tuvastada ründava meeskonna poolt algatatud liiklust. Lisaks võimaldab loodud lahendus liikluse profiile töötamise ajal dünaamiliselt uuendada ja koormust võrgule suurendada või vähendada.

Lisaks hinnati prototüübi kasutamisvalmidust küber-õppuse "Locked Shields 2014"raames. Hinnati lahenduse töökindlust kui ka pakutavat funktsionaalsust. Testi tulemuste ja osalenute tagasiside põhjal anti soovitused edasise funktsionaalsuse arendamiseks.

Annotation

This thesis focuses on analysing the issues that Cyber Defence Exercises (CDX) are experiencing in closed environments, Gamenets. The main issue of having insufficient legitimate traffic is explored. This thesis provided a solution to mitigate the problem of having insufficient legitimate traffic in the network, thus making attacks more visible.

While solving these issues the requirements for traffic generation in CDX environments were determined. A feasible proof-of-concept solution was proposed after the requirements were set and the development ideas presented. Furthermore, designing the architecture and developing the tool has been accomplished.

The completed tool was tested during the annual CDX named "Locked Shields 2014". The results acquired from the test run were analysed and evaluated. Thereafter, the solutions performance was validated according to the requirements. Additional features to be implemented were suggested based on the validation results and the feedback from the participants of the exercise.

Contents

List of Figures

Glossary

Blondes A person in each BT, whose task was to simulate the users of Blue systems. 14

Blue Team (BT) The side that is defending their infrastructure and services. 12, 16

bug A fault in the program or system, that caused it to return an incorrect or unexpected value. 39, 41

Capture The Flag A game based scenario where the teams try to get a token from the opposing teams systems.. 12

CCD COE NATO CCD COE. 12

CDX Cyber Defence Exercise. 3, 8, 11–16, 18, 20–23, 26, 31, 33, 35, 38, 42–44, 46, 47

CTF Capture The Flag. 12

Cyber In this thesis Cyber used as Cyber-Space Security field. 9, 11, 12, 16, 17, 22, 23

DDoS Distributed Denial of Service. 12

DMZ Demilitarized Zone. 24, 25

DNS Domain Name System. 12, 25, 40, 41, 43–45

DTLS Datagram Transport Layer Security. 25

EITC Estonian Information Technology College. 11, 16, 30, 46

FTP File Transfer Protocol. 25

Gamenet A virtual or simulated closed game environment. 3, 13, 14, 18, 21, 23, 36–39, 66

git Git is a free and open source distributed version control system. 30

Green Team (GT) Team responsible for infrastructure and management in Gamenet. 17

HTTP Hypertext Transfer Protocol. 23, 25, 28, 41

HTTPS Hypertext Transfer Protocol Secure. 23, 25, 41

human firewall a situation, where the operator is manually checking every incoming connection, which is feasible during a CDX but impossible in real-world environments. 13

ICMP Internet Control Message Protocol. 25, 41

IMAP Internet Message Access Protocol. 25

IMO International Military Organization. 12

IPT Inter Packet Timing. 19

IRC Internet Relay Chat. 9, 25, 28–30, 34, 42, 44

ISP Internet Service Provider. 40

LAB Laboratory, development environment. 24

LDAP Lightweight Directory Access Protocol. 25

LS13 Locked Shields 2013. 13, 14, 16, 23, 25

LS14 Locked Shields 2014. 12, 16, 17, 24, 25, 47

MAPI Messaging Application Program Interface. 25

NATO CCD COE NATO Cooperative Cyber Defence Centre of Excellence. 7, 11, 16

NFS Network File System. 25

ngIRCd Next Generation IRC Daemon. 30, 34

NTP Network Time Protocol. 25

P2P Peer to Peer. 28

POP3 Post Office Protocol version 3. 23, 25

PS Packet Size. 19

RDP Remote Desktop Protocol. 25

Red Team (RT) The side that is performing the attacks against other teams infrastructure and services. 12, 16

RPC Remote Procedure Call. 25

RRT Rapid Response Team. 17

SINET Simulated Internet. 25, 46

SITREP SITuation REPort, a periodic report of the current missions situation. 66

Skype Free voice over internet protocol (VoIP) service that allows users to communicate across an internet connection. 25

SMTP Simple Mail Transfer Protocol. 23, 25

SNMP Simple Network Management Protocol. 25

SSH Secure Shell. 25, 46

SSL Secure Socket Layer. 30

TCP Transmission Control Protocol. 19

UAV Unmanned Aerial Vehicle. 24

VoIP Voice over IP. 9, 23, 25

White Team (WT) Team responsible for the whole game play coordination, rules, legal play and communications. 12

XMPP Extensible Messaging and Presence Protocol. 25

1 Introduction

The aim of this thesis will be to develop a suitable and practical traffic generator that could be the underlying basis for current and future Cyber Defence exercises (CDX). The resulting solution will help to effectively conduct testing and execution of various tasks related to network traffic simulation. The main aim of the solution will be the use of the tool for Cyber exercises planned by NATO Cooperative Cyber Defence Centre of Excellence (NATO CCD COE). Furthermore, the tool will be of use in the *Estonian Information Technology College* (EITC) while developing new courses and curriculum modules for Practical Cyber Defence for IT System Administrators.

This thesis will be focusing on analysing and comparing different solutions available for Cyber Defence Exercises held all over the world. The author will bring out benefits and limitations of different systems while executing the exercises and creating a solution to make the experience more realistic for the participants.

> "Against organized Cyber attacks there should be organized defense."
>
> Ü. Jaaksoo[1]

This was the idea of Ülo Jaaksoo, while proposing to the Minister of Defence his idea of creating an organization to unite the elite of IT specialist to tackle Cyber attacks in the future. He proposed to conduct Cyber Defence Exercises to demonstrate how to deliver an attack and teach participants how to defend their systems against such attacks. The participants would experience first hand how to secure their systems and bring the knowledge back to their organizations. Jaaksoo's idea has since inspired many organizations to take part of the Cyber Defence Exercises. These ideas have emerged after Estonian government and private sector web sites were under Cyber attacks.

[1]Free translation of idea: Aaviksood vaimustas mõte küberkaitseliidu loomisest - http://epl.delfi.ee/news/eesti/aaviksood-vaimustas-mote-kuberkaitseliidu-loomisest.d?id=51103195, EPL, 2 Oct 2007

”Following the events in 2007, when Estonian government and private sector web sites were under Cyber attack, Distributed Denial of Service attacks (DDoS), Estonian government adopted the first Cyber security strategy for the period of 2008-2013.” [11, p. 38]

CCD COE (Cooperative Cyber Defence Centre of Excellence) was activated as an International Military Organization (IMO) by the decision of the North-Atlantic Council, making it the first IMO hosted by Estonia [6]. Soon after, CCD COE was called to existence, in December 2008 the first joint CDX was held.

> ”The exercise was a cooperative effort of the scientists from the CCD COE, Swedbank Estonia, Tallinn Technological University and the Swedish counterparts. During the exercise the students showed their abilities to tackle with the Cyber Attacks against important websites, e-mail accounts and DNS.” [7]

And continuous CDX-s followed: Baltic Cyber Shield in 2010, Locked Shields 2012, Locked Shields 2013[2].

To be successful the exercises need to have participants who are skilled in both attack and defence techniques. The attacking side is titled the Red Teams (RT) and defenders are titled the Blue Teams (BT). The definition for the BT and RT is as follows:

- The BT-s - are the main training audience of LS14. They are composed of IT specialists and legal advisers.

- The RT-s - The Red Team’s mission is to reckon, attack, compromise and degrade the performance of the BT systems. The phases and objectives will be pre-defined. During the execution of the exercise RT acts closely together with the *White Team* (WT).[3]

As Locked Shields is not a *Capture The Flag* (CTF) event, the focus of the exercises is to train the BT-s.

[2]Cyber Defence Exercises - https://www.ccdcoe.org/353.html (06.05.2014)
[3]Description of teams for LS14 - https://collab.mil.ee/collab/ls14/Teams

1.1 Main Problems

The exercises are conducted in a closed environment called Gamenet, where the currently available solution allows easily to identify attackers networks traffic among other. There is not enough traffic masking the RT-s efforts. In real world environment, there would be a lot of other network traffic originating from several users other than the attackers and masking the attempts. This situation lets the BT-s defend their assets more easily and does not simulate the real conditions. Currently, there is not a better solution to have more diverse and unique traffic in the network. Thus the BT-s are in advantage against the RT-s efforts. This exact situation played out during *Locked Shields 2013* (LS13) exercise:

> ”A very common activity was to block any IP address which seemed to be a source of suspicious actions. Detecting malicious traffic was relatively easy as the simulation system generally failed to create the expected amount of legitimate traffic.” [9, p. 34]

During LS13 execution, a human firewall[4][5] was enforced to check all incoming traffic during the CDX. In a simulated environment this will help to gather points, but it is not feasible in real environments and has nothing to do with reality. Furthermore, this behaviour would not benefit the experience and learning curve of the participants.

BT-s are graded by their performance to uphold their systems against attacks from RT. The score will be higher if the BT can mitigate attacks, update their systems and repair any damages done. Therefore, if the RT has no chance to perform their objectives infiltrating Blue systems, the BT would be left without valuable experience. Furthermore, this could leave them with impression of false security - the Gamenet is not real world alike and its purpose is to let the teams gather knowledge how to defend their systems more effectively.

There are many CDX-s held in many different countries and among many different organizations. They all have custom built Gamenets, set of rules and objectives. In current situation all the different executions employ systems that are not compatible with each

[4]Scripts from a participant of LS13 - `https://github.com/tarko/ls13blue8-scripts`

[5]Comment: ”Used these to monitor all incoming and outgoing traffic and pick out connections that haven’t been seen before. Outgoing based on src/dst/dstport tuple and incoming based on payload only (to get rid of all noise and scoring connections). These enabled us to very quickly spot malicious downloads, callbacks to red team machines etc. Scripts were fed from SPAN destination that aggregated all ingress and egress traffic from all our machines.”[18]

other - they are custom tailored. Customizing the solutions for different execution would be too expensive regarding the manpower and working hours required.

1.2 Main Objectives

To enhance the compatibility of CDX-s and prevent situations like during LS13 from happening again, there should be a simple but effective way to conjure up some network traffic to mask the activities of RT actions. In a closed environment this is not the easiest of tasks. The Gamenet has a predefined amount of traffic that is originating from within the network. Thus volunteers are invited to play the role of internal users (Blondes). Despite the Blondes' effort to comply with the tasks provided, it is still a hurdle to get enough unique traffic on the wire. This issue led to the idea of creating one such solution, that would be capable and able to provide sufficient legitimate traffic to create a ”smoke screen” to hide RT-s actions.

The thesis is focusing on enhancing CDX-s to make them more realistic. The main problem with current exercises has been that the attackers traffic can be determined far too easily. With this thesis the author is building a solution that will help make the identification of attackers traffic more cumbersome for the defenders. The solution will be a centrally managed modular framework to make the handling and generating of network traffic more effective.

The primary outcome of this thesis will be:

Objective 1. List of requirements for traffic generation,

Objective 2. Architecture of the solution,

Objective 3. System that generates network traffic, allows to modify traffic profiles and rate dynamically during execution,

Objective 4. Evaluation report based on live CDX named ”Locked Shields 2014”.

1.3 Outline of The Thesis

This thesis is organised into chapters. In Chapter 2 the current situation of conducting CDX-s is described and what difficulties are related to traffic generation during these events. Moreover the different methods of generated traffic are viewed. The third chapter is the main part of this thesis. Here we are analysing our requirements for a solution. Requirements for generating traffic during the CDX are described. Selecting tools and describing a solution is conducted in the end of this chapter. Next in the fourth chapter a solution is described. Herein the solution has already set up to work. In chapter five the evaluation of the developed tool is reviewed based on live CDX. We also describe what complications have risen and what lessons were learned. Chapter 6 is devoted to additional features that should be implemented in the coming events. And our conclusion we disclose in Chapter 7.

1.4 Acknowledgements

The author would like to show gratitude towards Margus Ernits for his support making this thesis happen. The ideas and insights provided were inspiring. Author also thanks Kaur Kasak, Risto Vaarandi, Rain Ottis, Artjom Lind for courses in TUT and UT which helped write this thesis.

2 Current Situation

This thesis is aimed on traffic generation for *Cyber Defence Exercises* (CDX). The main exercise in focus is Locked Shields organized by NATO CCD COE. The same functionality will be used in Estonian Information Technology College's (EITC) virtual laboratory system, where the students act as *Blue Team* (BT) members and protect their systems against a simulated *Red Team* (RT).

Locked Shields 2014 (LS14) is the third in series. The previous events have determined requirements that would help to enhance the exercise. The essence of Locked Shields CDX is to train BT-s in technical aspects to defend systems that they are given to maintain. It is a friendly competition and defence is the focus of the training. The scenario is predetermined and BT members are playing fictional roles. During the *Locked Shields 2013* (LS13) exercise there were 10 BT-s participating. In LS14 already 12 BT-s participated. For 2015 it is assumed that 20 BT-s will be participating in the exercise. That means the scale of the exercise is going to change. It has been brought out that one of the challenges during the execution is to have enough legitimate user traffic in the network[9].

Additional high level objectives are set for the execution and evaluating are the BT ready to face full scale and full-speed Cyber attacks. It is an essential part of the training while the same experience cannot be gained during every day routine. The work load on BT-s members is immense as there is massive amount of different events happening simultaneously and there is a lot of stress present - to keep communications up not only with other teams but also with the press. Furthermore, leadership skills and teamwork becomes essential. The teams are provided the opportunity to try out new tools and technology that is helping to mitigate issues in their everyday work.

In everyday work the Blue team members are specialists, who defend their own networks and systems. These are usually already secured and fall under strict management guidelines. Patching and fixing vulnerabilities is a must and done with care and precision.

The Cyber Defence Exercise is trying to simulate the real-world situations. Although during the execution the scenario (the scenario of Locked Shields) puts the experts in a role of a *Rapid Response Team* (RRT), which is tasked to take on the challenge to defend a specific infrastructure and services. They must fix the vulnerabilities in the systems and implementations, find and remove malicious code, patch unrestricted network access issues and many more tasks what is required of them by the scenario.

During the planning phase, the RRT-s are given a few weeks before the real execution to get to know their future systems. They can toy around and try to fix all vulnerabilities and issues they find. They have no knowledge in what state the systems are running or what malware persists already until the execution. Furthermore the systems are reverted to their initial state one week before the execution, leaving all fixed issues open again.

The RT members are usually professionals, who do penetration testing, or persons who usually are performing Red Teaming tasks during different exercises. All of them bring a valuable set of skills and knowledge to the arena. LS14 RT-s had 55[6] different objectives to execute against the blue systems. That means, the BT-s were under constant attack by RT-s, which in turn made the situation for the defending teams very stressful.

During the preparation phase the RT works closely together with the events organizers, the *Green Team* (GT). The idea behind is to build a system that is full of vulnerabilities and weaknesses the RT-s could exploit while training the BT-s to respond to different attack vectors. Thus opposite to BT-s the RT-s have full knowledge about the existing system and infrastructure beneath. This is described as a *"White box"* approach where all enumeration of targets, systems and weaknesses is already known. This helps to give momentum to the RT attacks during the exercise execution - the reconnaissance phase is already done and no time needs to be spent for *"Phase 1"* [7].

As the GT is trying to create a network and infrastructure as world alike as possible (trying to simulate Internet), it still comes down to a situation, where the only traffic on the wire is generated by the BT-s and scoring agents checking the availability of the systems. Noticing this network pattern lets the BT-s enforce white lists[8] and everything else will be blocked on the perimeters edge. Despite having knowledge about all the

[6]Red Team Objectives - `https://collab.mil.ee/collab/ls14red/Objectives`
[7]Phase 1 - Reconnaissance: Information Gathering before the Attack
[8]A white list is a list of items that is granted access to resources.

Gamenet, the RT-s are prohibited from achieving their objectives.

Now the real problem has been faced: how to hide the traffic, that the RT-s are creating while performing the attacks on BT-s networks and systems? Solving this problem allows to conduct more realistic future CDX-s.

2.1 Related Problems

Every CDX is unique. The scenarios vary every year and the organizing team is investing lot of effort to make the exercises interesting for all the participants.

The hurdle becomes apparent with the tools to be used. As everyone is aiming for getting better, the trend for longevity of the tools is decreasing. The tools are being built and evaluated during the exercises. It happens that the intent of the tool was higher than the real outcome. This can be accounted for the lack of manpower developing such tools.

The developing teams are often small - 1-6 people, and the expectations for the developed systems are set high. More than usual, the developers are volunteers who sacrifice their personal time to make something that will help conduct the exercise. It is not rare an exception, when the tool(s) are ready in the development environment, they will not work out of the box in the Gamenet. This in turn requires endless testing of the software and making sure that everything is performing as intended.

There shouldn't be a need to develop a new system for every CDX, but the current situation leaves no other option. Information on the performance of the tools used elsewhere is lacking and hard to acquire. The tools are a good kept trade secret of the organizers and developers. Thereafter necessity for tools to be used by a wider audience exists.

2.2 State of Art

In this section the author is describing the many different ways traffic could be generated.

Molnar et al. [15] are analysing the way traffic generators are categorized and generated traffic validated. They present, that in their sample they are reviewing, the traffic generators that can be found in their present literature are uniformly difficult to validate. The report dissects all five different categories identified and brings out the strengths and weaknesses traffic generating profile, measuring characteristics and ways of validation. These five categories are: replay engines, maximum throughput generators, model based generators, high-level and auto-configurable generators, special scenario generators. In Table 1 we have hand-picked a few of the traffic generators that were also in Molnar's samples. The criteria of choosing was:

- Supporting multiple platforms (Window, Linux, OSx, etc). It is crucial for medium-to-large deployments that there is an unison code-base for all the tools used. Therefore some of the tools from Appendix A are not suitable, they are designed for a specific Operating System.
- Their licensing is adequate and the work can be reused and customized.
- Script based, tools can be deployed separately, but with modified integration they are interacted with.

Table 1: *Table taken from [15, p.1361]*

Generator Category	**Traffic Generator**	**Description**
Replay Engines	TCPreplay	User-level application for replaying *libpcap* file
Maximum Throughput Generators	Iperf	User-level application for bandwidth, packet loss ratio and jitter testing
Model Based Generators	TG	Packet-level generator supporting various distributions for IPT and PS values
	MGEN	Packet-level generator supporting various distributions for IPT and PS values
High-Level and Auto-Configurable Generators	TMIX	Traffic emulator for ns-2 based on source-level characterization of TCP connections
	D-ITG	Extensive workload generation framework that can produce traffic for wide range of network scenarios

The full list of traffic generators sampled can be found in Appendix A. The tools we selected, are the ones that can be run on several different operating systems. They have a suitable licence (eg. GPLv3, New BSD License or similar license approved by *Open Source Initiative* (OSI)), and are open source. With these options we are able to build our own handles to interact with the software. To be noted is that none of the *Special Scenario Generators* made it into our pick. The reason is that they did not satisfy our criteria or there was not sufficient information available.

Vishwanath and Vahdat [20] have built the Swing traffic generator. It is a structural model that is accounting for interaction across multiple layers of protocol stack. The model takes into account end users who determine the communication characteristics. The main characteristics are how active the user has been, what site he/she visits and when, and the *delay* between visits. Secondly the model takes into consideration the individual session information, downloading multiple files from the server simultaneously, getting parts of the same audio file from different servers (the authors have file-sharing capabilities envisioned). Third, characteristics of single connections within a session is measured. What is the connection source and destination, the number of request and response pairs created, size of request and wait time before connection is established. Finally they convey the underlying network characteristics to their work. They are extracting the link loss-rates, capabilities and latencies for interconnected paths from the original trace file.

> ”We have found that model parameters that attempt to reproduce human/-machine think time are the most difficult to accurately extract and reproduce.” [20]

In [12], Guerber explains why in recent years virtual environments are more favourable over physical ones. Despite the costs involved to acquire a simulation environment, this investment will pay off immensely. Guerber reasons that the existing environment can be used to run CDX-s and other smaller scale exercises like table-tops. He also stresses that organizations, having such an environment in place, should conduct possible small-scale exercises and walk through possible scenarios. It is noted that there might be some shortcomings in test environments which are used for simulations. Namely they might lack user and/or traffic simulation equipment. In this Article Guerber also points to another shortcoming, that in virtual and simulated environments, where low traffic traverses the network, attacker traffic can easily detected and mitigated. He is suggesting similar techniques to mitigate this issue as we have laid out in subsection 3.2.1.

To generate legitimate SSH traffic, paper [2], presents us with a solution that is able to synthesize traffic based on real traffic samples. The process takes action in two phases. The first phase takes the sample traffic capture and filters the traffic characteristics that are required for phase two. The second phase is putting the initial information back together and generating the traffic. This method is also taking into account the diurnal patterns to get more accurate results.

As there are many different tools available for network simulation and network traffic generation, one could simply ask, why not do we simulate the internet in our closed environment. The answer might lie within the Internet's wast heterogeneity, many different devices connected, rapidly changing nature and many supported protocols involved [16], [5]. Furthermore, if we would start to simulate on such grand scale, we would have to take into consideration the longevity of our research. Reason is, if we model something today, we have to be prepared that it is still in use several years from now. This in turn means that generating legitimate and reliable traffic is really hard[5].

Considering the fact that a CDX is usually taking place over a few days, is performed in a closed Gamenet and has many participating teams, then simulating the Internet with all the diurnal activity patterns does not seem feasible. These characteristics are limiting the traffic generation to a certain specific model. There is need not for all available protocols and applications, but for a particular few, what the RT is using for their attack vector.

3 Analysis

In this chapter we will identify the requirements to be met for traffic generation during a CDX event. These requirements are then tied to goals that the author is pursuing to achieve with this thesis. Having the goals in focus, a system architecture will be suggested. Thereafter the components required to build the system are chosen and their usage explained.

The Information Warfare exercise ”Eligible Receiver” has been noted to be the first of its kind. The exercise was played out in U.S. in 1997. The ”RED TEAM” consisted of thirty-five people and the campaign lasted over 90 days. The team used publicly available tools and software to perform the attacks. A pre-set scenario was introduced and the exercise was spun-off. The findings of the exercise did reveal how vulnerable U.S. information systems were [13].

Since its execution back in 1997, ”Eligible Receiver” has inspired many other Cyber Security Exercises.

”Locked Shields” CDX series was inspired by the United States service academies annual CDX held since 2001[9], which in turn was inspired by ”Eligible Receiver”.

3.1 Different CDX formats

Many different kinds of exercises exist, each has a different format, different benefits, challenges and costs.

The format of the exercise will be chosen by the scope what is aimed for. Small scale

[9]What the CDX Challenge is - http://www.usna.edu/Cyber/_files/documents/CDX/What%20the%20CDX%20Challenge%20is.pdf

exercises like desk check[10], comms check[11], walk through[12], workshop[13][14], table-top[14] are relatively easy to organize as resources required are also small.

More teamwork and procedural effort is required by larger scope exercises as distributed table-top[15], command post[16] and full simulation[17][19]. In the *"Methods for Enhances Cyber Exercises"* some more specific exercise formats are noted aside the already mentioned [12].

3.2 Requirements

> "One of the challenges we face when conducting the exercises is lack of realistic amount of legitimate traffic. We have used custom solution able to generate few protocols (HTTP, HTTPS, SMTP, POP3, VoIP). However, during LS13 the changes done by the Blue Teams caused majority of the traffic agents to stop. Therefore it was fairly easy for the defenders to differentiate between malicious and legitimate traffic"[8]

To build a full-scale simulation, all aspects of the real-world likeness should be accounted for. The virtual network environment (Gamenet) is different for every exercise, although the building blocks are similar.

3.2.1 Requirements for Traffic Simulation

As previously identified, all the CDX-s are in need of legitimate network traffic and user emulation, which is not the easiest tasks in planning the exercise. Previous CDX-s

[10]Desk check - Early stage and validation of new plans. Usually held within a small group, and procedures regarding a simple scenario are discussed.

[11]Comms check - is organized to validate systems or infrastructure

[12]Walk through - When a response team is planning to take action and is going over the steps beforehand.

[13]Workshop - A previously agreed scenario is being rehearsed. Usually in open forum format to allow discussion of activities to be performed.

[14]Table-top - Simulates a situation in an informal and stress free environment.

[15]Distributed table-top - A scenario-based exercise where participants test plans and activities according to routine.

[16]Command post - Gives a team the possibility to rehearse action plans using their own facilities. Often is played out on the management level.

[17]Full simulation - Requires a real time environment simulation. All aspects of execution are aimed as close to reality as possible.

have provided some useful insight on how to build systems and networks, software and simulation. But we are still lacking solutions that will provide enough traffic to conceal the RT attack attempts.

This situation was acknowledged also by the organizers of LS14 and they have put together a list of requirements for networks and simulated/generated traffic. The requirements paper actually focuses on a commercial product, is it feasible to purchase it or not. Despite the intent, the report gives a good overview of what is expected [8].

The LS14 scenario foresaw that the BT systems were divided into five segments: Internal, DMZ, LAB, Expo and UAV. The connectivity was designed using best practices in mind. Although some segments were intentionally interconnected, UAV and LAB to be specific. This set-up was prepared to make RT attacks more reliable on interconnected segments. Moreover, the BT should have identified this situation and address it accordingly.

From this thesis perspective, we are addressing the requirements that have been set for clients (workstations) and services that are available to them. That cover the whole internal segment of blue systems and services in the DMZ (see Appendix B Figure 7 for clarification). In addition we had the opportunity to install agents on the LAB workstations.

The requirements are applied for all BT-s, and are as following:

Requirement 1 Generate traffic from the workstations in internal segment against other systems and services available to them,

- the intent is to simulate typical user behaviour,
- if traffic is originating from not within, the BT-s could create white lists to filter the generated traffic.

Requirement 2 The workstations will be running different versions of Windows and Linux operating systems (Windows XP, Windows 7, Windows 8, Ubuntu Linux etc.)

- the provided solution should be working with all before mentioned operating systems.

Requirement 3 Central management of the traffic agents,

- should be able to stop/start traffic during the execution,
- ability to change traffic rate and profile on the fly.

Requirement 4 If workstation has access not only to own systems, but other BT-s systems too, then the agent should be able to generate traffic towards the other BT systems,

Requirement 5 If possible, have background noise generated in the internal network segment,

- simulating multicast or broadcast traffic from network devices,
- if possible network related L2\L3 background traffic.

Requirement 6 Traffic from real/simulated clients toward the DMZ,

- various application layer protocols - HTTP, HTTPS, DNS, SSH, SMTP, POP3, IMAP, MAPI/RPC, RDP, FTP, NTP, LDAP, ICMP, VoIP, Skype, XMPP, IRC, DTLS, NFS, SNMP, etc.

Requirement 7 Typical traffic that an Internet connected system would see for background noise,

Requirement 8 Traffic content should be customizable,

Requirement 9 Simulate user sending attachments or downloading files from the WEB,

Requirement 10 Possibility to provide credentials for services logins,

Requirement 11 Filling of forms, and defining which fields to fill in,

Requirement 12 Inbound (from SINET to DMZ) traffic should be originating from many various legitimate IP addresses from same range that also RT is using.

Requirement 13 Simulate known attack vectors do create a smoke screen and trigger many false positive alarms.

Requirement 14 Be able to re-play the attacks mentioned previously and integrate that with grading.

3.3 Development Decisions

The organizes were short on time and gave out the requirements what are the features, they would like to see done by the solution. Furthermore we were left to decide what requirements we shall implement in our work. This meant, in the short time-frame given, implementing all requirements would have turned out to be really hard. Thus we had to determine the most important characteristics for the solution we had to develop. We decided, that we are not going to generate protocols, that were generated during LS13 by an existing tool that was intended to be used also during LS14. Although the existing tool was going to be overhauled and improved, it would still lack some key protocols like DNS, SAMBA, FTP, ICMP. The tool was able to generate network traffic with HTTP, HTTPS, SMTP, POP3 and IMAP protocols.

Thus we set out to create a framework, we could use to deliver additional features, as they ripen. We set goals, what the emerging solution should be able to do:

Goal 1 From **Requirement 1** and **Requirement 2** the most benefiting way to continue would be to select a cross platform solution. One could select the possibility to develop a separate package for all the platforms used during the CDX, but that would require much more time. Moreover maintaining the different code bases would be overwhelming. Therefore **cross-platform implementation** is our selected path.

Goal 2 Referring to **Requirement 1**, **Requirement 2** and **Goal 1** the system ought to be **open source**. Otherwise we would have complications finding supporting developers in the future. And closed source solutions are prone to be discarded, if it is not possible to develop additional functionality.

Goal 3 As the solution is required to be managed from a central entity, we have decided to implement **central management**. An alternative would be to command all the agents in single sessions, but the overhead in management would set limitation on a medium-to-large installation scale. This goal is closely tied to **Requirement 3** and **Requirement 8**.

Goal 4 To be able to satisfy **Requirement 4**, **Requirement 5**, **Requirement 6**, **Requirement 8** and **Requirement 9**, we have to develop the system to be **modular by design**. Having to implement all the required functionality into a single piece of software would be counter productive. Instead we will be implementing a system where the modules can be loaded dynamically when required.

Goal 5 **Requirement 8**, **Requirement 10**, **Requirement 11** are describing a tool that is **customizable throughout the execution**. If the traffic profiles and behaviours would be hard-coded[18], the software is required to be restarted after every modification.

As a small team and being short on time to deliver our product, we had chosen to go with a subset of *Agile Programming* method, which is suitable for projects big and small. What made the decision easier was, the model uses an iterative approach and it helps to deliver code that is working. The key deliverable of this concept is a solution that works. Having such a small team, only two persons, choosing a distinct method for programming was not feasible. Methods like Scrum and others need more persons in more roles. We over simplified the process of agile as there was no real need for the entire model to be applied. The best, the idea, was taken from the model. We present what was our work flow on Figure 1.

[18]Hard-coded - program behaviour written directly into programs code.

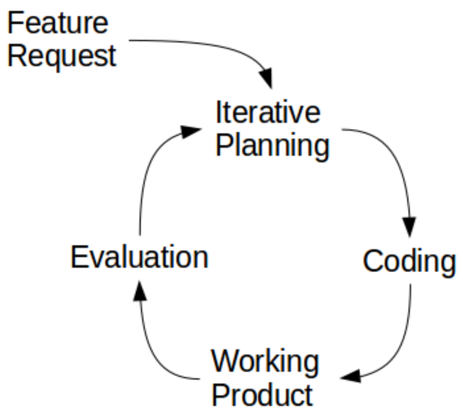


Figure 1: Our development cycle.

Having a team of two left the author of this thesis be the major contributor in programming the solution.
As the code-base we were developing was not that large, we were able to ship functionality pretty fast. Thus building the *"hello world"* sample for every additional feature was always a good starting point. This gave us the flexibility to get a working sample, and add additional functionality to it in the next iteration of requested features.

3.4 Architecture

To be able to satisfy Goal 1, the cross-platform dependency, a programming language is to be selected that is able to be run on all used operating systems (Windows and Linux). Programming languages that run on multiple platforms are Java, Ruby, C, Delphi, Perl and Python to name a few. Adding Goal 4 to the selection process, then the most modularity was shown by Python. Python was also used in various other network traffic generating tools like scapy, Ostinato, and several wireshark tools [22]. Selecting Python as the programming language of choice would enable the integration of other scripted tools. It is modular, runs on all the required platforms, updating the python code is easy, and simple enough to re-deploy. The later two statements are based on the the fact that python compiles the code previous to run, or uses previously compiled code. This gives the system the flexibility to update the scripts, stop the process and then start the process up again and Python is doing the rest.

Our Goal 3 requires a centralized approach, where we could control all traffic agents in one place and be able to do for all our operating systems. A similar approach has been used for the past decade - botnet[19].

Different ways to control a botnet can be identified:

(a) Centralized - the oldest known type of architecture used by botnets.
(b) Decentralized - no single point of failure, C&C server is no longer the single point of communication.
(c) Hybrid - in article [17] the more advanced characteristic of hybrid botnets are described (P2P).

In our endeavour we shall not need the malicious approach of the botnet, instead the central management, ability to be controlled, functionality to be modular and be updated as required [4]. This further expanded the opportunities to witch route to take, as several different examples of centralized structures could be found [23]- IRC and HTTP based.

The overall architecture of the slave-master system should be similar to Figure 2.

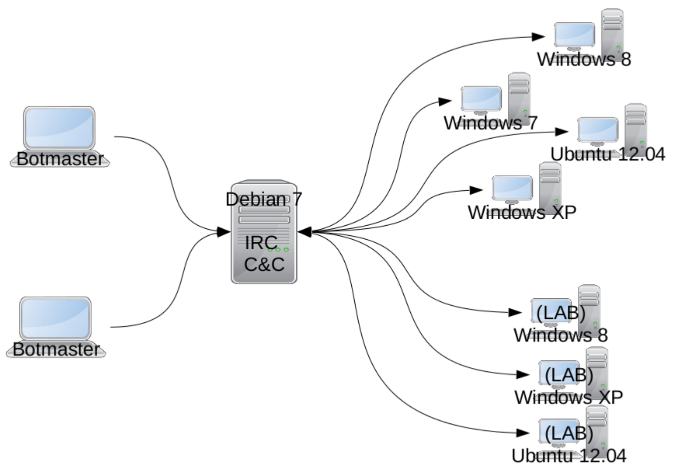


Figure 2: Initial planned architecture.

From Figure 2 it cannot be determined, but the C&C server and the bot bearing workstations are located in a virtual environment. Furthermore, the C&C is located in the RT_SINET[20] (see Appendix B Figure 9) and the workstations in the BT-s local network segment (see Appendix B Figure 7).

[19]Compromised or hijacked computers, which are obeying the commands of the owner of the botnet.
[20]Simulated Internet where also the RT is launching their attacks from, where the scoring and traffic generation system are located.

Satisfying **Goal 4** and **Goal 5** we must choose the right components and implement functionality that is allowing us to adjust the traffic profiles for services separately and be able to provide new target lists and traffic profiles throughout the exercise as the need arises. And this leads us to comply with **Goal 2** being open source as intention is to be able to pull in additional resources if required.

Leading from here we have to chose the components to be used. What components are used, and why, is explained in the next section of this thesis.

3.5 Selecting System Components

Initial thoughts were to build up the system from ground up, but eventually that seemed a task with a larger scope. Thereafter it was decided to use already existing components. This decision led to the next step, find the piece of software that would be suitable for the task of creating an IRC botnet. As there are many automated IRC chat bots already in existence, the choice of picking one had to be made.

Having the goals (see section 3.3) set, finding a suitable solution was the next logical step. During the research it was considered to turn to already existing malicious botnets in existence, because they have all the required functionality built in. Despite the lucrative idea, it was decided against such actions. We might have been able to alter some of the code of the malicious tool, but it still might inhibit functionality which the initial botmaster was able to activate remotely. Furthermore, if the program seemed to be clean, files that were required to run the instance might have backdoors[21] built in. And having such malicious tool in our arsenal was not in our intent.

This left the author with looking for a suitable package of components. An incomplete list of existing IRC based chat bots can be found here [21]. Previous searches did not turn up anything useful, so the list was re-checked. The selection had to be based on our earlier selection of goals, cross-platform, modular and easy to update. Own criteria was that the chat bot should have on-going development, and latest version should be quite recent. It was decided to go with an IRC chat bot named ”**Willie**”, authored by Edward Powell, Dimitri Molenaars, Elad Alfassa, Ari Koivula. In conjunction with willies own modularity, the additional modules to support functionality to meet **Goal 3-Goal 5** had to be developed.

[21]backdoor - method for bypassing authentication procedures.

Having a herd of chat bots alone was not sufficient for the task at hand. An IRC server had to be set up additionally. Having the Debian distribution installed, from the default repository one could download available packages of different suitable IRC servers. The choice was made easier by the fact that willie chat bot supports SSL, and in anticipation of the BT-s ingenuity, making sure the information were sending over the network does not traverse the network in plain-text as it would for the standard IRC protocol. The selection of IRC servers was made in favour of *Next Generation Internet Relay Chat Daemon* (**ngIRCd**[22]), a sophisticated enough server to satisfy our requirements. The configuration of the server was documented and based on the procedure an installation guide was compiled.

Wanting to have additional feedback from the bots, it was decided to use a log-server. During the course of "Linux System Administration" at EITC a student had prepared a good step-by-step guide to deploy a solution that would satisfy the need of a log-server [3]. Based on his work we put together or own instructions. The building blocs used to get the log-server running are **Apache2**[23], **Kibana**[24], **Logstash**[25] and **Elasticsearch**[26]. All of them are open source.

Not being able to build the whole system in one session meant that we need code version control. Here the distributed version control system named *git* came very handy. It was not rational to have the git server run on our own systems. There were several options available to host source code on public repositories. One that was selected to be used is Bitbucket[27]. Using Bitbucket left us to have more freedom to code andl not hassle with managing the availability of the server. For initial development purposes a local installation of ngIRCd was used in virtual environment. Later on it was possible to use the e-lab environment at EITC.
The first vision of how the outcome of the development should look like is depicted on Figure 3.

[22]ngIRCd - http://ngircd.barton.de/documentation.php.en
[23]Apache HTTP server – http://projects.apache.org/projects/http_server.html
[24]Kibana - http://www.elasticsearch.org/overview/kibana/installation/
[25]Logstash - http://logstash.net/docs/1.3.2/tutorials/getting-started-centralized
[26]Elasticsearch - http://www.elasticsearch.org/guide/
[27]Bitbucket - https://bitbucket.org/features

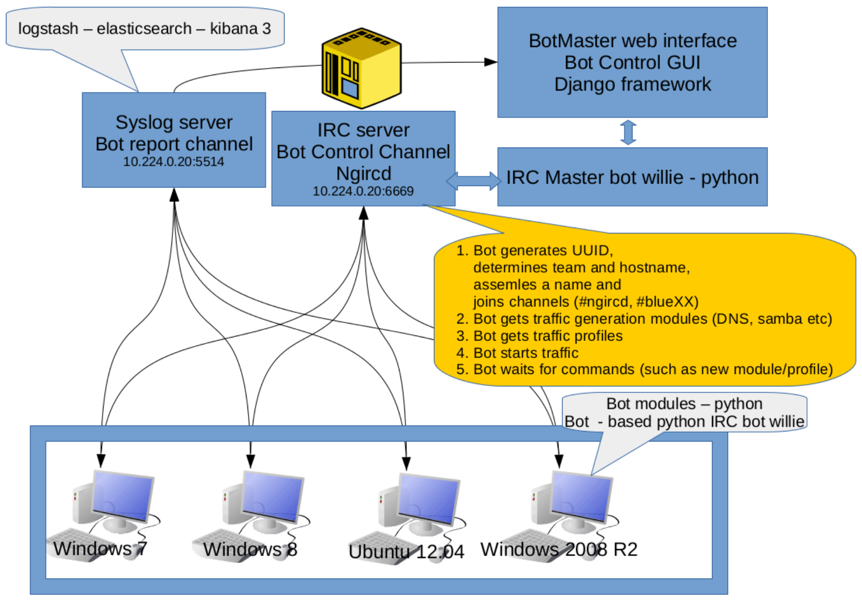

logstash – elasticsearch – kibana 3
BotMaster web interface Bot Control GUI Django framework
Syslog server Bot report channel 10.224.0.20:5514
IRC server Bot Control Channel Ngircd 10.224.0.20:6669
IRC Master bot willie - python
1. Bot generates UUID, determines team and hostname, assemles a name and joins channels (#ngircd, #blueXX)
2. Bot gets traffic generation modules (DNS, samba etc)
3. Bot gets traffic profiles
4. Bot starts traffic
5. Bot waits for commands (such as new module/profile)
Bot modules – python
Bot - based python IRC bot willie
Windows 7
Windows 8
Ubuntu 12.04
Windows 2008 R2

Figure 3: System architecture.

Using iterations to enhance the functionality of the modules, it was required to verify and evaluate the performance of the already written code. The test-environments were therefore a good solution. Furthermore the code had to be deployed on several test-workstations running all the different operating systems that would be present also in the up-coming event environment. Manually deploying the enhancements turned out to be too labour extensive, thus it was suited to build an automated deployment. This same functionality would be required to be available within **Goal 4** and **Goal 5**. Continuing to satisfy the last mentioned goals, re-deployment of the system would count towards functional availability and customization readiness. And cross-platform integration in mind, scripts to do the automatic re-deployment were written. This helped to extend the automated installation scripts given to the organizers prior to the CDX for deployment onto their systems.

For Windows environments the development encountered an issue. There was no good way to manage the services with available tools to Windows. The **service controller** (sc) with Windows is unable to restart a service. Starting and stopping a service is in sc's capabilities but nothing more. Preventing issues with exceptions and network problems, it was essential to be able to run the service ”indefinitely”. Thus an external program was introduced which turned out to be very beneficial. The **Non-Sucking Service Manager** (nssm)[28]. It did allow to set the service to be restarted. Furthermore, it was watching the

[28]nssm - http://nssm.cc/

service and if the service failed or was killed, nssm restarted it again. This issue did not turn up with Linux, as for Ubuntu, **upstart**[29] is initially coming with the installation and it behaves the exact same way as nssm would.

For time being the source code of the solution is not publicly available. It is currently shared per-request.

[29]Upstart - `http://upstart.ubuntu.com/`

4 Solution

This chapter will describe the installation process of components. The solution contains a component of a many bots and a component of central management infrastructure (see Figure 3).

A solution satisfying the requirements (Section 3.2) and goals (Section 3.3) set in the previous sections was built. The development proceeded following the previously determined architectural references (see Section 3.4). Source code of the solution is available on Bitbucket. And anyone interested to acquire the source code should contact the author.

In the following sections we will describe the processes required to get the systems running and operational.

4.1 Traffic Agent Installation

The solution that emerged was a software package. It was possible to deploy the solution based on simple instructions handed down by the author. The preparations for the CDX were done by a team, who prepared all the BT workstations and following the instructions the software was deployed. The installation instructions can be found in Appendix C.

The execution path for the solution was different for Windows and Linux operating systems. In Linux it had to be installed under **/root/scoring_traffic/willie**, whereas in Windows environment it was installed in **C:\Windows\scoring_traffic\willie**. This allowed the program to run in elevated privileges.

To have some persistence and fail-safes in place for sudden exceptions, the program was set to run as service. For Linux distributions this was achieved by invoking the functionality of **upstart**, with Ubuntu it is available out-of-the-box. For Windows environments **nssm** was picked. All of the service configuration was done by the installation scripts (see Appendix D.1 and Appendix D.2).

After installation is complete, the service "willie" should be started. In Linux it can be done by command-line with:

```
# In Linux
$# start willie

# In Windows from command-line
C:\textbackslash > sc start willie
```

While starting up, willie is generating a suitable name for itself and then it is trying to connect to the provisioned IRC server. While networks can be unreliable, willie is in an endless *trying-to-connect* loop, until a successful connection is made.

Now the user controlling the bots hsould connect to the IRC server and join the predefined command channel. When everything is lined up, and bots are ready, the user can control the generation of traffic by issuing the following command into the channel:

```
.mload all 100
```

The command seen, will adjust the load of all available network traffic modules to be 100. The wait time between generated requests will be ranging from 0.001 to 0.01 seconds. This translates to a full load. The load can be adjusted to be also 1000, then the delay between requests goes into milliseconds.

Additional commands to control the bots and traffic profiles exist and are disclosed in the source-code documentation.

4.2 C&C Infrastructure Installation

The previous instructions aimed to install the willie traffic bot. But the solution used some external server for logging and C&C communications. Getting logs from the traffic bots, a syslog server was installed in virtual environment. On the same host the C&C channel, IRC server was running. To be able to install and configure both services in an efficient way, the initial installation was documented and installation instructions compiled. A copy-paste approach was chosen, because it leaves less room to make critical errors. The installation manuals had additional section on what should be replaced if the system would be deployed elsewhere. The installation manual for the ngIRCd server can be found in Appendix E, and the installation manual for logging purposes can be found in Appendix F.

Further automation was implemented on into the build cycle. As it required many manual steps from committing the changes into git, to be able to deploy a new release (this was meant for re-deployment of the whole package of willie), the server side work was scripted. This meant, the job scheduler *cron*[30] was set up to execute re-build cycle every 5 minutes. The line in crontab file:

```
*/5 * * * * robot cd /home/robot/ls14/willie/utils \
        && git pull && bash makeistallzip.sh > \
        /home/robot/ls14/cronjob.txt 2>&1
```

Now all the components were installed and ready to run. Re-deployments were automated.

For willie management, the important directories with modules reside in willie home directory (in Linux and Windows specified earlier). The extra modules, that give willie the ability to adjust traffic load and use re-deployment functionality are hidden away in ”willie\modules”. The original modules that came with willie are also located in this directory. It was not deemed important to remove the original modules.

All automatic build and re-deploy related scripts were re-located to willie’s home under *utils*. This directory is referred to all the automation scripts during build and re-deploy process.

[30]Crontab - http://manpages.debian.org/cgi-bin/man.cgi?query=crontab&sektion=5

Traffic generating modules were designed to reside somewhere more accessible. Thus they were located in a willie's home subdirectory named ”dne” (download-and-execute). This subdirectory was designed to be more accessible. All the traffic profiles and target-lists were saved in this directory. While giving command to download a new profile or target-list, this file was automatically saved into ”dne” directory.

The authors main contribution and most time consuming task was the programming and development of the solution. In the next chapter the solution will be tested in a live CDX named ”Locked Shields 2014”.

5 Evaluation of The Solution

The evaluation of the system was done in several steps which included a test run, live event execution, and questionnaire sent to participants.

5.1 The Test Run

The Test Run was held on March 28, 2014. The Test Run is a limited game-play between one RT and two BT-s. The BT members participating in the Test Run will not participate during the main Execution as BT members. The Test Run BT-s are assembled from the students of Tallinn University of Technology.

Objectives for test run were[31] pre set by the organizers. Identify problems in the infrastructure and Gamenet systems was the main objective for performing the test run. It was paying of as many issues were identified and mitigated during the test run. Another objective was to test communications means and information flow between all the teams participating and systems activated. Reporting and automated scoring were next on the list to be tested and turned out to be working with minor issues, which were fixed during the run. The rules of the event were presented and reviewed to be certain that every one has a clear overview and that the rules are being understandable to all.

As the test run was mainly meant to determine the capability of the infrastructure and fluidity of the underlying processes, the event was just one day long. Also the BT-s systems were not fully finalized. Only a small set was operational and it was planned that new functionality would be added after the test run. Also only part of the scenario and injects were tested. One aspect of the scenario that was left untested intentionally was the legal play.

[31] Test Run Objectives - https://collab.mil.ee/collab/ls14/Test_Run

There was a test run of the whole system one and a half months before the Live event. It was performed with only two BT-s active. So to say the load on the infrastructure had to be kept low. In fact, many a problem arose. The underlying Gamenet crumbled. It broke down and teams were unable to connect.

Problems were detected not only by the infrastructure team. The initial program code which was expected to run seamlessly, just failed. The underlying infrastructure had network time-out issues and the clients kept dropping out the command channel. The test run gave valuable insight on what should be worked on to be able to survive networking issues.

During and after the test run the code was partially rewritten to suit the situations encountered. The problems discovered during testing in Gamenet:

- Unreliable network connections killed the bots. A mechanism to prevent a total blackout had to be implemented.

- Unpredictable behaviour, the script was leaking processes, eventually eating up all available resources and thus killing the workstations it was running on.

- Fast re-deployment of the full code-set was missing from the arsenal of code.

This valuable information helped to prepare for the real event. While it is not possible to fix all the issues, mitigation is the next best approach. Quickly a roadmap for further developing was laid down:

1. Creating a reliable re-deployment procedure, rolling out new code needs to be tested and a lot of manual labour was involved previously. Automating the generation of deliverable packages reduces the time spent on deployments.

2. Producing an automated installation for all supported systems was the next essential task. Two main approaches were implemented - an installation script for Debian Linux based systems and the other based on Windows systems. Both approaches shared a similar execution path but implementation issues were different.

3. Implement fail-safes if there are network issues in the Gamenet or with workstations. Implementing the program to run as a service was mitigation enough for the workstations. If the service failed to start, it was re-scheduled to start indefinitely. Mitigating network failures was done by creating a connectivity check loop.

5.2 The Live Event

The Live event was taking place during 19-23 May 2014. The main timetable is given in Appendix G. This time frame was further split. The whole first day was dedicated for briefing the RT members in a workshop.

Day-0 was set to be May 20th. There were a lot of preparations still to be done in the facilities where the event was taking place. All network connectivity and access tests were planned to this day. Furthermore, all communication tests with BT-s had to be performed from ground up. After, testing, briefings and system touch ups were done, the Gamenet was closed and BT systems reverted back to initial snapshots.

Day-1, May 21st (Phase I and Phase II). We arrived at site early to make sure that all the expected bots were running. We were prepared to start troubleshooting if there would have been a requirement for that. For our surprise, we had almost all the clients already connected to our command channel. The default traffic profile provided to the bots was set to 0. Meaning they were just idling and waiting for commands to be sent to them. The initial count of bots in the channel was 176. That was the highest count noted during the CDX.

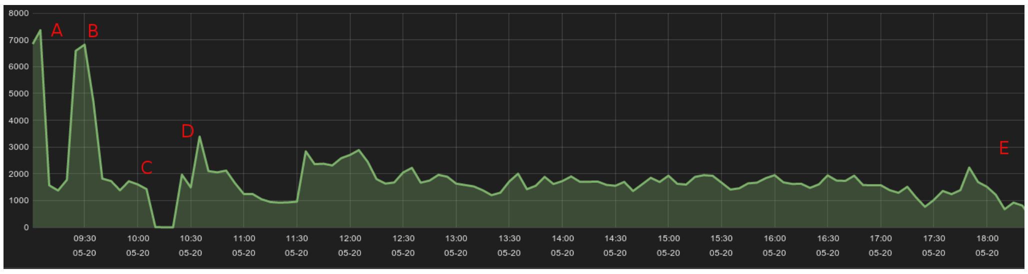


Figure 4: Day-1 traffic generation

Graph explained. We did not implement measuring of traffic generated. Instead, if a bot started a traffic profile or action, an event was logged to the logging server.

Phase I (07:30Z - 11:00Z / 10:30 - 14:00 GMT+3h).
As we arrived earlier, we started testing available bots. It was our aim to test how much the infrastructure could handle. The load was altered to be half of what we expected to be the maximum. On the graph that moment in time is marked with the letter A. We received word from the networking team that the underlying network is having issues. The load of traffic agents was reduced to 5 (on the figure 4, the low between peaks A and B). After getting confirmation of network being healthy, the load was increased back

to our initial test mark. Suddenly bot started to drop out of the command channel (On the figure 4, dipping of spike B). Further investigation revealed that the issue was again in networking. And then the network collapsed. This is a distinct low on the figure 4, marked with the letter C. The interruption lasted for a whole ten minutes (noted to be from 10:10 - 10:20 GMT+3). Eventually the network got fixed and the Live Event started. At 10:30 (GMT+3) the *STARTEX*[32] was given (on the figure 4 marked with D). As the bots gradually joined, the traffic profiles were modified to reflect the traffic load of 5, as it was tested initially.

Phase II (11:00Z - 14:00Z / 14:00 - 17:00 GMT+3h).
All together Phase II was uneventful regarding our traffic generating. During this phase we did troubleshooting of our code and tried to get our heads around why the traffic generator would kill the network. We were not the only ones having problems. Infrastructure team was battling on their own to have the network perform as planned. For the remainder of Phase II, the traffic generating profile was left to the 5 margin and not a lot of traffic was sent to the network. The time was used responsibly and we did discover some minor bugs. Fixing the bugs gave a boost to our traffic generating ability. Unfortunately this could not have been tested until the competition day was over (marked on the figure 4 with E, 17:00 GMT+3h), and the Gamenet closed for BT-s access.

After Gamenet closure, we were able to test our fix to the traffic agents. The results were satisfying. The results could be noted during Day-2.

Day-2, May 22nd (Phase III and Phase IV).
Phase III (07:15Z - 11:00Z / 10:15 - 14:00 GMT+3h).
Fixing the code the previous night the was promising to be able to generate more traffic for this day. After arriving, we tried to re-deploy the code to as many bots as possible. An average of 140 bots was available in the command channel. Despite the fact we had a fixed code, not all bots were able to download the new version. This time the problem lay with the BT-s who had restricted external Internet access. Till that moment we relied on our out-of-band test environment to get updates and download to the managed bots. Despite the fact that not all agents were updated, our initial test was successful (on the figure 5 between A and B). Phase III started at 10:15 (GMT+3) and we were asked to increase the traffic even further than we had dared already. On the figure 5, point B is indicating the start of phase III and also the increase in traffic events generated.

[32]Start of Exercise

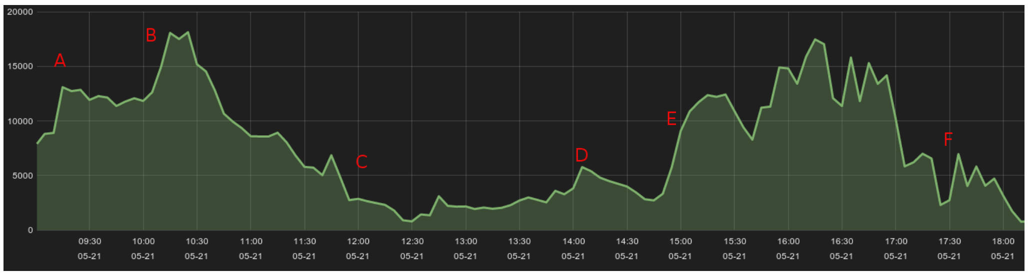


Figure 5: Day-2 traffic generation

The decline of traffic, near B on figure 5, represents the gradual decrease of bots on the command channel. The BT-s were closing required ports on their firewalls and hosts were rebooting too. The scenario foresaw an ISP incident for phase III. The replacement of a router was simulated and the connections to the existing connection was cut. On figure 5, this event can be seen at mark C. Two teams had set up resilience the day before and their connections failed over to the backup router. The other BT-s were given time to re-configure their routers to get the connections back up. To be fair, the traffic profile was downgraded on the active two teams.

Phase IV (11:00Z - 13:30Z / 14:00 - 16:30 GMT+3h).
As phase III came to an end at 14:00 (GMT+3), phase IV had just started. Many BT-s still had issues with their networking, which eventually got solved. Figure 5, point E marks the recovery of BT-s connections. The traffic load for all online traffic agents was set equally. This procedure was repeated several times, because the bots left and joined the channel and it was hard to track which of them already had the profile set. We had green light from the organizers to increase the traffic load for all teams. It was set that DNS traffic to be generated at a load of 100 (translated to requests sent from workstations every 0.1 - 0.01 seconds, minimal wait time between requests), and SAMBA load to 35-50 (we monitored our own SAMBA server, and if it started to choke, load was decreased accordingly). The extra traffic went not unnoticed by the BT-s. They started to reboot the workstations and tried to figure out ways to throttle the the traffic. Although, we had about 120-140 bots to command during phase IV, the traffic generated was sufficient. Varying traffic is represented on the figure 5 between point E and F, where the latter also represents the *ENDEX*[33] of the game.
The overall view of traffic events can be seen on figure 6. When we refer to load, we mean wait time between requests. If the load is 10, then the delay between requests will vary from 0.01 - 0.1s. If the load is set to 1000, the delay between requests randomly varies from 100μs - 1ms.

[33]End of Exercise

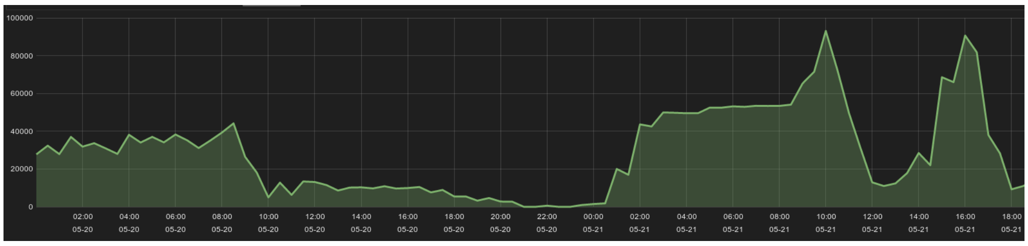


Figure 6: All event traffic generation

Figure 6 makes it really clear, that during the 1st half of the game, there were lots of issues present on the infrastructure side as well in the code (mentioned before as bugs). After the stumbling blocs were removed, the entire system started to work as expected.

5.3 Feedback from Participants

The BT-s point of contact persons were sent a short questionnaire regarding the traffic generator used during the exercise. The same questionnaire was submitted to Red team members. Questionnaire is added in Appendix H. And we are summarizing the answers.

Surprisingly few answers came back. But those, who took time and answered, gave valuable insight on what to improve in the future. One major goal was highlighted when RT members answered that generating on additional traffic was helping them to stay concealed. On the other hand it cannot be confirmed on full scale. Despite being aware of the traffic generated, they have requested for more DNS and ICMP traffic for the next execution. It was requested to generate more IPv4 and IPv6 traffic, not being clear what protocols in general are meant.
An eye opening request was that next execution should be managed to generate traffic between BT systems. Thus far the scenario has not seen it to be done. Hence the suggestion to administrative team will be made and the suggestion requested for further executions.
Other teams, other than red and blue, have answered that specific WEB servers, they were responsible for or had interaction with, were totally untouched by any traffic generation. The logs indicated action only, when the RT was doing its reconnaissance sweeps. This indicates misinformation in providing target lists. In addition, it was also mentioned to generate additional HTTP/HTTPS traffic.

The only BT, who answered in time, wrote that they noticed our traffic generator in the workstations, but did not tie that to the additional traffic coming from workstations.

More to our astonishment, the BT contact had written that they need to be informed, if any such agent is installed on the systems by organizers. Actually it was written in black-and-white in the rules that given ports shall be left open to servers named. Further more it was detailed what the agent was doing and what the aim of it was. This lets us conclude that the BT-s do not read the rules in depth, or just skim them through. That will be noted and improved in further preparations.

The fact that the BT-s did not respond back, might have something in common with our goals. If we are successful in generating lots of legitimate noise in the network, then they will be having more troubles identifying the real danger - meaning they have to look harder to score better. Which would be the case in a real-life situation anyway. The real-world-experience is not so forgiving as the CDX might make us think.

5.4 Lessons Learned

Despite the lack of a reporting interface, the command channel (IRC) was sufficient. Every team was assigned a private channel where all the interactions with bots could be monitored. And what was requested several times from the managerial team was to tune the load of traffic profiles throughout the exercise.

Phase I of the CDX was promising. 12 BT-s had all their workstations prepared and online. The highest count of bots concurrently in the command channel was 176. 3 additional users for management tasks are not counted for. The highest density of bots was observed only during a short period in beginning of the exercise. After that the numbers started to dwindle. The main hurdle for the teams was that they did not read all the rules presented to them.

The rules stated a few **important** guidelines:

- specific port towards the traffic servers should be open during the whole execution,
- the tools in a predetermined directory are for grading and traffic simulation only, and are off limits to the BT-s and RT-s,
- during the execution all pre-planted programs must remain fully functional on the workstations (*exception is malware, which can be removed).*

There were teams who did not follow the given rules. That meant the bots could not call in to receive new instructions. In terms of fairness, this sort of action gave some teams minor advantage, as the default traffic generation threshold was set low to begin with. On the contrary, the team managing the bots was able to pursue the teams to comply with the rules eventually. Despite the fact that the rules were eventually followed, some bots stayed missing. That was due to the fact that some workstations were rendered inoperable - low disk space was the reason that some bots could not download new modules or edit workloads. Another reason was the RT who was manipulating the workstations while completing their objectives.

During the 1st Live execution day, after all bots were active and traffic pattern thresholds set, the node, which was capturing and recording all network traffic, started to show symptoms of overload. The traffic pattern thresholds were adjusted and the capturing node recovered. This event gave a good indication how much traffic the online bots were able to generate. Main pressure point was DNS traffic, as RT-s was supposed to be using DNS tunnelling to perform attacks on blue systems[34].

The live exercise was interrupted by a network failure during the 1st day, in which the traffic generating botnet played a minor role. The underlying routing infrastructure could not handle all the traffic and failed. A quick fix was to lower the traffic thresholds and increase it in steps.

The Lessons learned from this CDX:

- Improving the communication to BT-s regarding understanding of rules is a must for future CDX-s. Rules must be crystal-clear to every participant.
- provide liaisons with more authority to give negative scores, if BT-s are not following rules,
- the underlying infrastructure, network in particular, is still in need of improvements,
- inter-BT traffic should be generated to have more realistic experience,
- traffic between BT systems and users should be in requirements set by the scenario,
- distribute target list early and as complete as possible.

5.5 Conclusion of Evaluation

The goals set for the system did not come out during the test run held several weeks before the live event. Instead, during the test run many flaws were discovered in the solution, and fixing them during the run was in focus. The main culprit was networking,

[34]Iodine - http://dev.kryo.se/iodine/

which was unreliable during the test run.

It can be said that the goals we set to deliver were met. Herein we provide the necessary overview of the solution functioning as promised.

- Willie traffic bot was running on the BT workstations, which again were running different operating systems (Windows XP, Windows 7, Windows 8 and Ubuntu Linux 12.04). This concludes that the **Goal 1** and also **Requirement 1** and **Requirement 2** are met. Our cross-platform developing proved to be also beneficial not only running on different operating systems, but also in maintaining a single set of code.
- Willie is able to run on several platform and utilizing several different tools while doing so. A solution that is based on open-source tools has not to be open source itself. But it helps along to get new developers to cooperate and introduce new ideas and functionality. Additionally giving some functionality back to the community is helping to conduct other CDX-s more reliably.
- **Requirement 3** was requesting for centrally managed traffic agents and in addition to be able to adjust the traffic profiles and rate during the execution. In conjunction with **Requirement 8**, asking for customizing support, it can be stated that both requirements were fulfilled. Throughout the exercise it was possible to adjust the traffic rate of the bots. During day one the high load supported the failure of the network. Therefore traffic load had do be decreased. This could be achieved by sending commands from the IRC C&C channel to all available bots. The increase and decrease of traffic can be witnessed from figure 6, which gives an overall view of traffic generated. Individual figures, showing the daily traffic pattern are shown respectively on figure 4 for day one and on figure 5 for day two. This also means that our **Goal 3** has been met.
- **Goal 4** was dependant on **Requirement 4**, **Requirement 5**, **Requirement 6**, **Requirement 8** and **Requirement 9**. The organizers gave the developers free hands and choice to implement different protocols. Initially the bots did not know of any other SAMBA servers and no cross-team traffic was generated during day one. After fixing minor issues with the traffic script we uploaded the script to all available bots. One more step prior to make the new code active. The reload command had to be given. After that indication of more generated traffic was noticeable. This is indicated on the figure 5 with the letter A. As for **Requirement 9** we did not implement the downloading of web-mail attachments, instead the downloading of files from network shares. From the given set of protocols, only DNS, SAMBA and telnet were generated. Despite the low variety, it was possible to create and upload new

modules and traffic profiles as requested and execute or load them at will. Meeting the expectations for **Goal 4** are satisfied.

- **Goal 5** was backed by **Requirement 8**, **Requirement 9** and **Requirement 10**. The developed solution provided the functionality to be customizable throughout the execution. This statement holds ground as the bot was multiple times re-deployed, the traffic profiles (ex. SAMBA) were updated during execution, DNS and other traffic rate was adjusted as requested by the organizers and networking team. The ability to be versatile and adjustable has been met.

6 Future Research

With this thesis the author has built a proof-of-concept framework for use in future CDX-s. Some requirements were not satisfied, but will be met with future development of the given solution. In order they appear, they are: first, **Requirement 7**, which was addressing the traffic that a system with internet connection would see. This requirement could be met in the future development of the tool, as the bots could be installed on servers that have direct SINET access. Second, **Requirement 12** suggesting traffic to be generated outside of BT networks. Moreover the various legitimate range of origin address could be developed by a additional module. The module should be able to assign an IP address to itself and after task completion the address is released. Third, **Requirement 13** needs some initial traffic capture or further investigation on simulating the attack vectors. In addition it has to be clarified which exact attacks are required to be simulated. And once more this request is for traffic originating from SINET. Fourth and last of the requirements is **Requirement 14**. Replaying attacks has been out of the scope of the simulator. To be able to replay the attacks additional functionality to listen in on a network and capturing the traffic unnoticed would be required to be developed. Other way around would be to capture and filter the traffic previously during a test run.

In addition to the previously mentioned enhancements, many more traffic profiles should be developed. A good example is SSH traffic. Ways to generate synthetic SSH traffic is also presented in [2]. Furthermore, there are additional capable tools available to create load on the network. Integration with such tools is one aspect that needs to be investigated. If possible creating modules that are running on multiple platforms. Many good tools are Linux oriented, but as shown in this thesis, achieving cross-platform ability should be pursued.

Next deployment of the completed solution will be set up in EITC-s virtual environment to support the security related laboratories such as ”IDS/IPS solutions” and ”WEB applications and security”.

7 Conclusion

This thesis was inspired by an existing problem within Cyber Defence Exercises (CDX). In closed environments there was not enough traffic to mask the actions of attacking team conducting their attacks. This was one of the main problems with current exercises.

The author explained the real hardship behind the main problem and why it had been an elusive achievement until this time. Furthermore, additional methods for generating network traffic were explained to present an overview of what tasks lie ahead.

During the analysis phase of the thesis, the author highlighted the requirements identified for network traffic generation. Derived from the requirements an architectural concept was proposed based on the requirements. As this solution contained many different components, a distinctive set was put together. Work towards developing a proof-of-concept solution was the next step.

Having finished the development, the completed solution was tested in a real CDX environment (LS14). An evaluative overview of the solution was provided based on the results derived from the execution and the participants feedback. Additionally, the lessons learned from the execution provided a way to enhance future events. Additional features were suggested for implementation in upcoming iterations to improve the solution.

The outcome of this thesis has a practical solution that can be utilized in closed environments to conduct competition alike events. The practicality was the ability to dynamically adjust the traffic profiles and rates provided. Moreover, the system was flexible enough that updating and re-deploying was possible. Updating the traffic modules and target-lists were handled by just downloading the new files to the traffic-agent. A cross-platform solution was developed that was running on several different operating systems available during the CDX. The Solution was released under Eifel Forum Licence (EFL) v2 license. The goals for this thesis were, first, assembling a list of requirements for network traffic generation. Second, developing the architecture of the solution. Third, develop a network traffic generator that allows to adjust the traffic rate and profiles during execution. Lastly, evaluate the solution in real life exercise.

The author's contribution was establishing requirements, system architecture, programming most parts of the solution and evaluating the developed solution in a live Cyber Defence Exercise, "Locked Shields 2014". Therefore all goals have been achieved.

The developed system was made available on Bitbucket repository. In case there was an interest to deploy the complete solution in another environment, the instructions were provided within the appendixes (Appendix C, Appendix D, Appendix E and Appendix F). For time being, the source code can be distributed only by request.

Bibliography

[1] A. Botta, A. Dainotti, and A. Pescapé. Do you trust your software-based traffic generator? *IEEE Communications Magazine*, 48:158–165, 2010. ISSN 01636804. doi: 10.1109/MCOM.2010.5560600.

[2] H. Djidjev. Network Traffic Generator for Cyber Security Testbeds. *Los Alamos National Laboratory Associate Directorate for Theory, Simulation, and Computation (ADTSC) LA-UR 13-20839*, 2:78–79, 2013.

[3] T. Esko. Keskse logihalduse loomine. https://wiki.itcollege.ee/index.php/Keskse_logihalduse_loomine, 2014. [WWW](2014-02-21).

[4] M. Feily, A. Shahrestani, and S. Ramadass. A survey of botnet and botnet detection. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on*, pages 268–273. IEEE, 2009.

[5] S. Floyd and V. Paxson. Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking*, 9:392–403, 2001. ISSN 10636692. doi: 10.1109/90.944338.

[6] NATO CCD COE. Press announcement of the CCD COE - October 28, 2008. https://www.ccdcoe.org/21.html, 2008. [WWW](2014-05-02).

[7] NATO CCD COE. Press announcement of the CCD COE - December 9, 2008. https://www.ccdcoe.org/91.html, 2008. [WWW](2014-05-02).

[8] NATO CCD COE. Traffic generation for CDX requirements. draft, July 2013.

[9] NATO CCD COE. Cyber Defence Exercise Locked Shields 2013 - After Action Report. http://www.ccdcoe.org/publications/LockedShields13_AAR.pdf, 2013. [WWW](2014-05-02).

[10] NATO CCD COE. Cyber Defence Exercise Locked Shields 2014 - After Action Report. draft, 2014.

[11] RISO and MKM. Estonian information society yearbook 2011-2012. http://www.riso.ee/sites/default/files/info%C3%BChiskonna%20aastaraamat_2011_ENG_FINAL_0.pdf, 2011. [WWW](2014-04.01).

[12] A. Guerber, C. Fogle, C. Roberts, C. Evans, B. MacDougald, and M. Butkovic. Methods for enhanced cyber exercises. https://www.hsdl.org/?view&did=740209, 2010. [WWW](2014-04-29).

[13] S. A. Hildreth and T. Division. CRS Report for Congress - Cyberwarfare. http://www.fas.org/irp/crs/RL30735.pdf, 2001. [WWW](2014-05-01).

[14] D. Jacobson and J. A. Rursch. Workshop-using cyber defense competitions (CDCs) to engage and recruit students with IT: How to organize and run your own cyber defense competition. In *Proceedings - Frontiers in Education Conference, FIE*, 2009. ISBN 9781424447152. doi: 10.1109/FIE.2009.5350412.

[15] S. Molnar, P. Megyesi, and G. Szabo. How to validate traffic generators? *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 1340–1344, June 2013. doi: 10.1109/ICCW.2013.6649445. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6649445.

[16] V. Paxson and S. Floyd. Why we don't know how to simulate the internet. In *Proceedings of the 29th conference on Winter simulation*, pages 1037–1044. IEEE Computer Society, 1997.

[17] S. Sparks and C. C. Zou. An Advanced Hybrid Peer-to-Peer Botnet. *IEEE Transactions on Dependable and Secure Computing*, 7(2):113–127, Apr. 2010. ISSN 1545-5971. doi: 10.1109/TDSC.2008.35. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4569852.

[18] T. Tikan. My LS13 scripts. https://github.com/tarko/ls13blue8-scripts, 2013. [WWW](2014-05-20).

[19] P. Trimintzios and R. Gavrila. On national and international cyber security exercises. survey, analysis and recommendations. http://www.enisa.europa.eu/activities/Resilience-and-CIIP/cyber-crisis-cooperation/cce/

cyber-exercises/exercise-survey2012, 2012. [WWW](2014-05-22).

[20] K. V. Vishwanath and A. Vahdat. Realistic and responsive network traffic generation. *ACM SIGCOMM Computer Communication Review*, 36(4):111, Aug. 2006. ISSN 01464833. doi: 10.1145/1151659.1159928. URL http://portal.acm.org/citation.cfm?doid=1151659.1159928.

[21] Wikipedia. Comparison of internet relay chat bots. http://en.wikipedia.org/wiki/Comparison_of_Internet_Relay_Chat_bots, 2014. [WWW](2014-02-10).

[22] Wikipedia. Wireshark Tools — Wireshark Wiki. http://wiki.wireshark.org/Tools, 2014. [WWW](2014-05-10).

[23] H. R. Zeidanloo and A. A. Manaf. Botnet Command and Control Mechanisms. *2009 Second International Conference on Computer and Electrical Engineering*, pages 564–568, 2009. doi: 10.1109/ICCEE.2009.151. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5380180.

Appendix A — Validation Techniques for Traffic Generators

Table taken from [15, p.1361]

Generator Category	Traffic Generator	Description	Validation
Replay Engines	TCPreplay	User-level application for replaying *libpcap* file	No validation since TCPreplay is a user-level software
	TCPivo	High-speed kernel-level replay engine	Inter Packet Timing error
	Divide and Conquer	Replay technique for OC-48 traces using multiple Gigabit Ethernet PCs	Wavelet scaling analysis and IPT error
Maximum Throughput Generators	Iperf	User-level application for bandwidth, packet loss ratio and jitter testing	No validation since Iperf is a user-level software
	BRUTE	Kernel-level packet generator	Throughput deviation compared to tuned values
	BRUNO	Hardware implemented packet generator	Throughput deviation and IPT error
	KUTE	Kernel-level packet generator	Throughput and IPT accuracy
	Ostinato	User-level packet generator with friendly GUI	No validation since Ostinato is a user-level software
Model Based Generators	TG	Packet-level generator supporting various distributions for IPT and PS values	IPT and PS values compared to tuned parameters presented in [1]
	MGEN	Packet-level generator supporting various distributions for IPT and PS values	IPT and PS values compared to tuned parameters presented in [1]
High-Level and Auto-Configurable Generators	HARPOON	Flow-based traffic generator that can mimic net-flow based measurements	IComparison of original and synthetic traffic by throughput, byte, packet and flow volumes, PS distribution and wavelet scaling
	SWING	Closed-loop, network responsive traffic generator which is able to extracts distributions for user, application, and network behaviour of real measurements	Comparison of original and synthetic traffic by quantitative statistics values, wavelet scaling of different applications and distribution of various QoS metrics
	TMIX	Traffic emulator for ns-2 based on source-level characterization of TCP connections	IComparison of original and synthetic traffic by throughput, flow size, RTT and application data unit distributions and wavelet scaling.
	LiTGen	Open-loop, packet-level traffic generator based on realistic IP traffic modeling Extensive	Comparison of original and synthetic traffic by QoS parameters based on queuing models and wavelet scaling
	D-ITG	Extensive workload generation framework that can produce traffic for wide range of network scenarios	Comparison of original and synthetic traffic by throughput and distributions of IPT and PS values
Special Scenario Generators	EAR	Traffic replay technique for mimic IEEE 802.11 protocol behaviour	Unique metric for measuring wireless traffic replay called *Event Reproduction Ratio*
	ParaSynTG	Web traffic generator	Web specific metrics such as document size and popularity distributions

Table taken from [15, p.1361]

Generator Category	**Traffic Generator**	**Description**	**Validation**
	YouTube Workload generator	Workload generation methods for mimic YouTube video traffic	Online video specific metrics like video length, size and rating distributions or cache performance
	Graph-Based Traffic Generator	Flow trace generator based on Traffic Dispersion Graphs templates	Graph related metrics such as distribution of degrees or connected edges and verticals

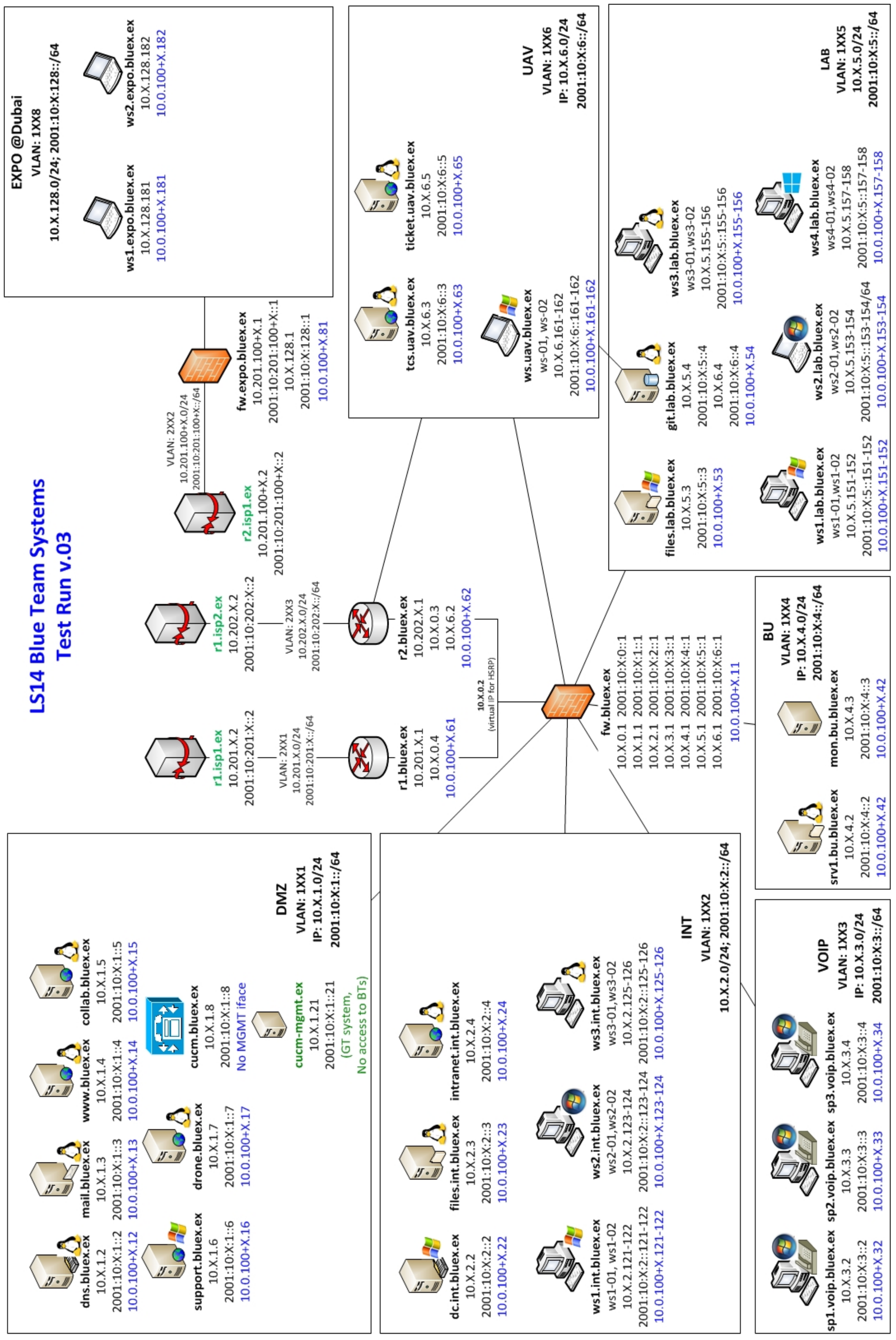
Appendix B — LS14 Topologies Overview

All the provided Figures in Appendix B are taken from [10]. Permit to use these figures was granted. The figures are hereby the property of NATO CCD COE until ”Locked Shield 2014 After Action Report” will be published.

Figure 7 - Blue Systems Overview for Blue Teams

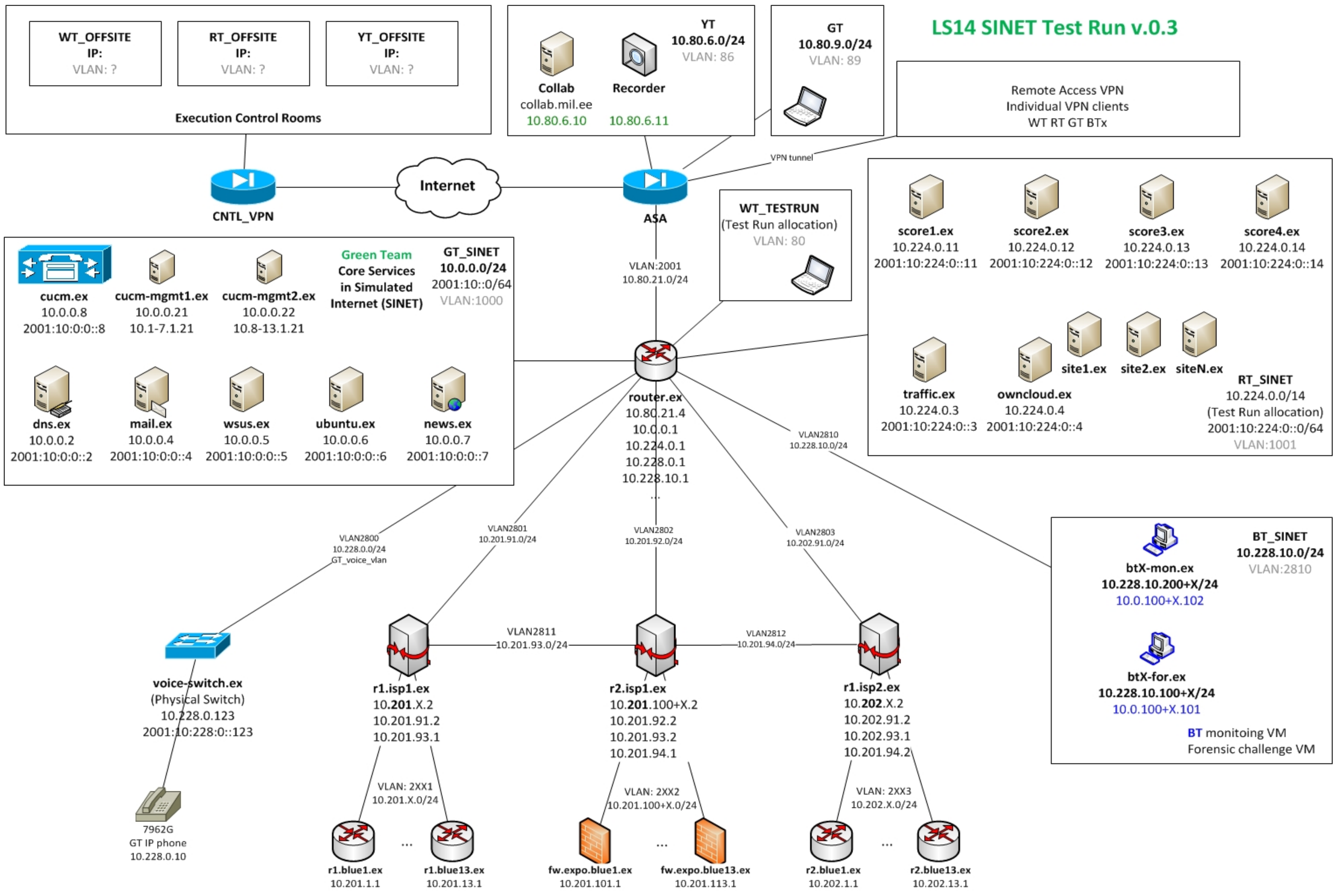
Figure 8 - SINET overview for Blue Teams

Figure 9 - LS14 Overall SINET Overview v2.2



LS14 Blue Team Systems
Test Run v.03
EXPO @Dubai
VLAN: 1XX8
10.X.128.0/24; 2001:10:X:128::/64
ws1.expo.bluex.ex 10.X.128.181 10.0.100+X.181
ws2.expo.bluex.ex 10.X.128.182 10.0.100+X.182
VLAN: 2XX2 10.201.100+X.0/24 2001:10:201:100+X::/64
fw.expo.bluex.ex 10.201.100+X.1 2001:10:201:100+X::1 10.X.128.1 2001:10:X:128::1 10.0.100+X.81
r2.isp1.ex 10.201.100+X.2 2001:10:201:100+X::2
r1.isp1.ex 10.201.X.2 2001:10:201:X::2
r1.isp2.ex 10.202.X.2 2001:10:202:X::2
VLAN: 2XX1 10.201.X.0/24 2001:10:201:X::/64
VLAN: 2XX3 10.202.X.0/24 2001:10:202:X::/64
r1.bluex.ex 10.201.X.1 10.X.0.4 10.0.100+X.61
r2.bluex.ex 10.202.X.1 10.X.0.3 10.X.6.2 10.0.100+X.62
10.X.0.2 (virtual IP for HSRP)
fw.bluex.ex
10.X.0.1 2001:10:X:0::1
10.X.1.1 2001:10:X:1::1
10.X.2.1 2001:10:X:2::1
10.X.3.1 2001:10:X:3::1
10.X.4.1 2001:10:X:4::1
10.X.5.1 2001:10:X:5::1
10.X.6.1 2001:10:X:6::1
10.0.100+X.11
UAV
VLAN: 1XX6
IP: 10.X.6.0/24
2001:10:X:6::/64
tcs.uav.bluex.ex 10.X.6.3 2001:10:X:6::3 10.0.100+X.63
ticket.uav.bluex.ex 10.X.6.5 2001:10:X:6::5 10.0.100+X.65
ws.uav.bluex.ex ws-01, ws-02 10.X.6.161-162 2001:10:X:6::161-162 10.0.100+X.161-162
LAB
VLAN: 1XX5
10.X.5.0/24
2001:10:X:5::/64
files.lab.bluex.ex 10.X.5.3 2001:10:X:5::3 10.0.100+X.53
git.lab.bluex.ex 10.X.5.4 2001:10:X:5::4 10.X.6.4 2001:10:X:6::4 10.0.100+X.54
ws3.lab.bluex.ex ws3-01,ws3-02 10.X.5.155-156 2001:10:X:5::155-156 10.0.100+X.155-156
ws1.lab.bluex.ex ws1-01,ws1-02 10.X.5.151-152 2001:10:X:5::151-152 10.0.100+X.151-152
ws2.lab.bluex.ex ws2-01,ws2-02 10.X.5.153-154 2001:10:X:5::153-154/64 10.0.100+X.153-154
ws4.lab.bluex.ex ws4-01,ws4-02 10.X.5.157-158 2001:10:X:5::157-158 10.0.100+X.157-158
BU
VLAN: 1XX4
IP: 10.X.4.0/24
2001:10:X:4::/64
srv1.bu.bluex.ex 10.X.4.2 2001:10:X:4::2 10.0.100+X.42
mon.bu.bluex.ex 10.X.4.3 2001:10:X:4::3 10.0.100+X.42
DMZ
VLAN: 1XX1
IP: 10.X.1.0/24
2001:10:X:1::/64
dns.bluex.ex 10.X.1.2 2001:10:X:1::2 10.0.100+X.12
mail.bluex.ex 10.X.1.3 2001:10:X:1::3 10.0.100+X.13
www.bluex.ex 10.X.1.4 2001:10:X:1::4 10.0.100+X.14
collab.bluex.ex 10.X.1.5 2001:10:X:1::5 10.0.100+X.15
support.bluex.ex 10.X.1.6 2001:10:X:1::6 10.0.100+X.16
drone.bluex.ex 10.X.1.7 2001:10:X:1::7 10.0.100+X.17
cucm.bluex.ex 10.X.1.8 2001:10:X:1::8 No MGMT iface
cucm-mgmt.ex 10.X.1.21 2001:10:X:1::21 (GT system, No access to BTs)
INT
VLAN: 1XX2
10.X.2.0/24; 2001:10:X:2::/64
dc.int.bluex.ex 10.X.2.2 2001:10:X:2::2 10.0.100+X.22
files.int.bluex.ex 10.X.2.3 2001:10:X:2::3 10.0.100+X.23
intranet.int.bluex.ex 10.X.2.4 2001:10:X:2::4 10.0.100+X.24
ws1.int.bluex.ex ws1-01, ws1-02 10.X.2.121-122 2001:10:X:2::121-122 10.0.100+X.121-122
ws2.int.bluex.ex ws2-01,ws2-02 10.X.2.123-124 2001:10:X:2::123-124 10.0.100+X.123-124
ws3.int.bluex.ex ws3-01,ws3-02 10.X.2.125-126 2001:10:X:2::125-126 10.0.100+X.125-126
VOIP
VLAN: 1XX3
IP: 10.X.3.0/24
2001:10:X:3::/64
sp1.voip.bluex.ex 10.X.3.2 2001:10:X:3::2 10.0.100+X.32
sp2.voip.bluex.ex 10.X.3.3 2001:10:X:3::3 10.0.100+X.33
sp3.voip.bluex.ex 10.X.3.4 2001:10:X:3::4 10.0.100+X.34

Figure 7: Blue Systems Overview for Blue Teams [10]



LS14 SINET Test Run v.0.3
WT_OFFSITE IP: VLAN: ?
RT_OFFSITE IP: VLAN: ?
YT_OFFSITE IP: VLAN: ?
Execution Control Rooms
Collab collab.mil.ee 10.80.6.10
Recorder 10.80.6.11
YT 10.80.6.0/24 VLAN: 86
GT 10.80.9.0/24 VLAN: 89
Remote Access VPN Individual VPN clients WT RT GT BTx
VPN tunnel
CNTL_VPN
Internet
ASA
VLAN:2001 10.80.21.0/24
WT_TESTRUN (Test Run allocation) VLAN: 80
cucm.ex 10.0.0.8 2001:10:0:0::8
cucm-mgmt1.ex 10.0.0.21 10.1-7.1.21
cucm-mgmt2.ex 10.0.0.22 10.8-13.1.21
Green Team Core Services in Simulated Internet (SINET)
GT_SINET 10.0.0.0/24 2001:10::0/64 VLAN:1000
dns.ex 10.0.0.2 2001:10:0:0::2
mail.ex 10.0.0.4 2001:10:0:0::4
wsus.ex 10.0.0.5 2001:10:0:0::5
ubuntu.ex 10.0.0.6 2001:10:0:0::6
news.ex 10.0.0.7 2001:10:0:0::7
score1.ex 10.224.0.11 2001:10:224:0::11
score2.ex 10.224.0.12 2001:10:224:0::12
score3.ex 10.224.0.13 2001:10:224:0::13
score4.ex 10.224.0.14 2001:10:224:0::14
traffic.ex 10.224.0.3 2001:10:224:0::3
owncloud.ex 10.224.0.4 2001:10:224:0::4
site1.ex
site2.ex
siteN.ex
RT_SINET 10.224.0.0/14 (Test Run allocation) 2001:10:224:0::0/64 VLAN:1001
router.ex 10.80.21.4 10.0.0.1 10.224.0.1 10.228.0.1 10.228.10.1 ...
VLAN2810 10.228.10.0/24
BT_SINET 10.228.10.0/24 VLAN:2810
btX-mon.ex 10.228.10.200+X/24 10.0.100+X.102
btX-for.ex 10.228.10.100+X/24 10.0.100+X.101
BT monitoing VM
Forensic challenge VM
VLAN2800 10.228.0.0/24 GT_voice_vlan
VLAN2801 10.201.91.0/24
VLAN2802 10.201.92.0/24
VLAN2803 10.202.91.0/24
VLAN2811 10.201.93.0/24
VLAN2812 10.201.94.0/24
voice-switch.ex (Physical Switch) 10.228.0.123 2001:10:228:0::123
7962G GT IP phone 10.228.0.10
r1.isp1.ex 10.201.X.2 10.201.91.2 10.201.93.1
r2.isp1.ex 10.201.100+X.2 10.201.92.2 10.201.93.2 10.201.94.1
r1.isp2.ex 10.202.X.2 10.202.91.2 10.202.93.1 10.201.94.2
VLAN: 2XX1 10.201.X.0/24
VLAN: 2XX2 10.201.100+X.0/24
VLAN: 2XX3 10.202.X.0/24
r1.blue1.ex 10.201.1.1
r1.blue13.ex 10.201.13.1
fw.expo.blue1.ex 10.201.101.1
fw.expo.blue13.ex 10.201.113.1
r2.blue1.ex 10.202.1.1
r2.blue13.ex 10.202.13.1

Figure 8: SINET overview for Blue Teams [10]

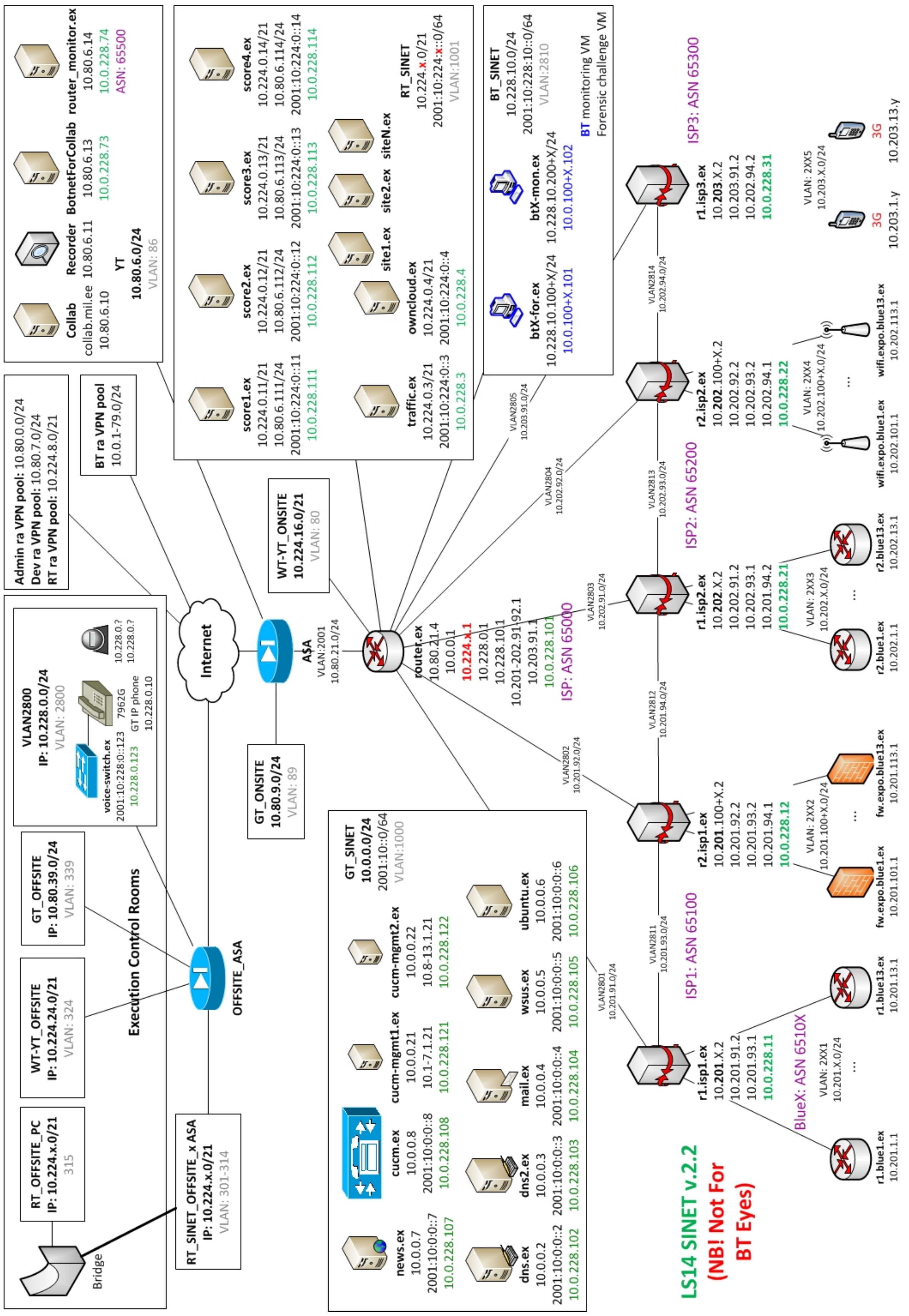

LS14 SINET v.2.2
(NB! Not For BT Eyes)
Bridge
RT_OFFSITE_PC IP: 10.224.x.0/21 315
WT-YT_OFFSITE IP: 10.224.24.0/21 VLAN: 324
GT_OFFSITE IP: 10.80.39.0/24 VLAN: 339
Execution Control Rooms
VLAN2800 IP: 10.228.0.0/24 VLAN: 2800
voice-switch.ex 2001:10:228:0::123 10.228.0.123
7962G GT IP phone 10.228.0.10
10.228.0.? 10.228.0.?
RT_SINET_OFFSITE_x ASA IP: 10.224.x.0/21 VLAN: 301-314
OFFSITE_ASA
Internet
Admin ra VPN pool: 10.80.0.0/24
Dev ra VPN pool: 10.80.7.0/24
RT ra VPN pool: 10.224.8.0/21
BT ra VPN pool 10.0.1-79.0/24
GT_ONSITE 10.80.9.0/24 VLAN: 89
ASA VLAN:2001 10.80.21.0/24
WT-YT_ONSITE 10.224.16.0/21 VLAN: 80
Collab collab.mil.ee 10.80.6.10
Recorder 10.80.6.11
BotnetForCollab 10.80.6.13 10.0.228.73
router_monitor.ex 10.80.6.14 10.0.228.74 ASN: 65500
YT 10.80.6.0/24 VLAN: 86
score1.ex 10.224.0.11/21 10.80.6.111/24 2001:10:224:0::11 10.0.228.111
score2.ex 10.224.0.12/21 10.80.6.112/24 2001:10:224:0::12 10.0.228.112
score3.ex 10.224.0.13/21 10.80.6.113/24 2001:10:224:0::13 10.0.228.113
score4.ex 10.224.0.14/21 10.80.6.114/24 2001:10:224:0::14 10.0.228.114
traffic.ex 10.224.0.3/21 2001:10:224:0::3 10.0.228.3
owncloud.ex 10.224.0.4/21 2001:10:224:0::4 10.0.228.4
site1.ex site2.ex siteN.ex
RT_SINET 10.224.x.0/21 2001:10:224:x::0/64 VLAN:1001
btX-for.ex 10.228.10.100+X/24 10.0.100+X.101
btX-mon.ex 10.228.10.200+X/24 10.0.100+X.102
BT_SINET 10.228.10.0/24 2001:10:228:10::0/64 VLAN:2810
BT monitoring VM Forensic challenge VM
router.ex 10.80.21.4 10.0.0.1 10.224.x.1 10.228.0.1 10.228.10.1 10.201-202.91-92.1 10.203.91.1 10.0.228.101
ISP: ASN 65000
GT_SINET 10.0.0.0/24 2001:10::0/64 VLAN:1000
news.ex 10.0.0.7 2001:10:0:0::7 10.0.228.107
cucm.ex 10.0.0.8 2001:10:0:0::8 10.0.228.108
cucm-mgmt1.ex 10.0.0.21 10.1-7.1.21 10.0.228.121
cucm-mgmt2.ex 10.0.0.22 10.8-13.1.21 10.0.228.122
dns.ex 10.0.0.2 2001:10:0:0::2 10.0.228.102
dns2.ex 10.0.0.3 2001:10:0:0::3 10.0.228.103
mail.ex 10.0.0.4 2001:10:0:0::4 10.0.228.104
wsus.ex 10.0.0.5 2001:10:0:0::5 10.0.228.105
ubuntu.ex 10.0.0.6 2001:10:0:0::6 10.0.228.106
VLAN2801 10.201.91.0/24
VLAN2802 10.201.92.0/24
VLAN2803 10.202.91.0/24
VLAN2804 10.202.92.0/24
VLAN2805 10.203.91.0/24
VLAN2811 10.201.93.0/24
VLAN2812 10.201.94.0/24
VLAN2813 10.202.93.0/24
VLAN2814 10.202.94.0/24
ISP1: ASN 65100
ISP2: ASN 65200
ISP3: ASN 65300
r1.isp1.ex 10.201.X.2 10.201.91.2 10.201.93.1 10.0.228.11
r2.isp1.ex 10.201.100+X.2 10.201.92.2 10.201.93.2 10.201.94.1 10.0.228.12
r1.isp2.ex 10.202.X.2 10.202.91.2 10.202.93.1 10.201.94.2 10.0.228.21
r2.isp2.ex 10.202.100+X.2 10.202.92.2 10.202.93.2 10.202.94.1 10.0.228.22
r1.isp3.ex 10.203.X.2 10.203.91.2 10.202.94.2 10.0.228.31
BlueX: ASN 6510X
VLAN: 2XX1 10.201.X.0/24
VLAN: 2XX2 10.201.100+X.0/24
VLAN: 2XX3 10.202.X.0/24
VLAN: 2XX4 10.202.100+X.0/24
VLAN: 2XX5 10.203.X.0/24
r1.blue1.ex 10.201.1.1
r1.blue13.ex 10.201.13.1
fw.expo.blue1.ex 10.201.101.1
fw.expo.blue13.ex 10.201.113.1
r2.blue1.ex 10.202.1.1
r2.blue13.ex 10.202.13.1
wifi.expo.blue1.ex 10.202.101.1
wifi.expo.blue13.ex 10.202.113.1
3G 10.203.1.y
3G 10.203.13.y

Figure 9: LS14 Overall SINET Overview v2.2 [10]

Appendix C — Willie Traffic Bot Installation Instructions

INSTALLATION GUIDE: Willie TrafficBot

Ubuntu GNULinux

Install pre-requirements

python-setuptools, python-yaml

- sudo apt-get install python-setuptools
- sudo apt-get install python-yaml

Installing willie IRC bot

Installation to be run with **root** privileges! From robot.itcollege.ee/ls14/ download (ask password from Margus||Erki)

- https://robot.itcollege.ee/ls14/install.sh

Run the shell script: **bash install.sh** It will ask for password to download further content from robot.itcollege.ee. After the script finishes, Willie should be installed and running (install dir. /root/scoring_traffic/willie).

Windows (XP/Vista/7/8)

Install pre-requirements

- Python2.7

Installation to be run under a user having **Administrative privileges** (installing a service)! From robot.itcollege.ee/ls14/ download (ask password from Margus||Erki)

- https://robot.itcollege.ee/ls14/PyYAML-3.11.win32-py2.7.exe

Installing willie IRC bot

Download:

- https://robot.itcollege.ee/ls14/willie.zip

Then unzip willie.zip to a temporary catalog. Browse into the catalog **willie\utils** and execute **install.bat** The script should accomplish the installation task on WinXP/7/8. Willie will be installed in **C:\windows\scoring_traffic\willie**

For further questions contact:

- Margus Ernits (margus.ernits@gmail.com)
- Erki Naumanis (erki.nuamanis@gmail.com; skype: erkinaumanis)

Appendix D — Installation and Re-Deploying Scripts

D.1 Linux Installation Script

```
#!/bin/bash
"""
Willie IRC Bot CDX Installation package creator
Copyright © 2014, Margus Ernits, <margus.ernits@gmail.com>
Copyright © 2014, Erki Naumanis, <erki.naumanis@gmail.com>
Licensed under the Eiffel Forum License 2.
"""
export LC_ALL=C

apt-cache policy  grep 'Installed: (none)' \ && apt-get install python-yaml -y
#apt-cache policy python-setuptools | \
        grep 'Installed: (none)' && apt-get install python-setuptools -y \
        && easy_install pyasn1

test -d /root/scoring_traffic/willie/ && rm -rf /root/scoring_traffic/willie/
stop willie
mkdir /root/scoring_traffic/willie/ -p
cd /root/scoring_traffic/
echo "Enter password for download willi bot"
read -s PASSWORD
wget --user=ls14 --password=$PASSWORD --no-check-certificate \
        https://robot.itcollege.ee/ls14/willie.zip -O willie.zip

unzip willie.zip
cp willie/utils/willie.conf /etc/init/
start willie
rm -f willie.zip
```

D.2 Windows Installation and Re-Depoyment Script

```
#!/bin/bash
"""
Willie IRC Bot CDX Installation package creator
Copyright © 2014, Margus Ernits, <margus.ernits@gmail.com>
Copyright © 2014, Erki Naumanis, <erki.naumanis@gmail.com>
Licensed under the Eiffel Forum License 2.
"""

echo "Creating installation zip file"

apt-cache policy p7zip-full|grep 'Installed: (none)' \
        && sudo apt-get install p7zip-full -y


test -r nssm.exe || {
    echo "Please put nssm.exe to utils directory"
    exit 1
    }


cat > install.bat <<EOF
@echo off
:begin
set OLDDIR=%CD%

if not exist "c:\python27" goto err
if not exist "c:\windows\scoring_traffic" goto mkscore
if not exist "c:\windows\scoring_traffic\willie" goto mkdir

:copy
cd %OLDDIR%
cd ..
cd ..
xcopy /s /e /q /y /h "willie" "c:\windows\scoring_traffic\willie\"

C:\windows\scoring_traffic\willie\utils\nssm.exe install willie \
        c:\python27\python.exe c:\windows\scoring_traffic\willie\willie.py

goto :eof
```

```
:mkscore
c:
cd \\windows
mkdir scoring_traffic
goto :mkdir

:mkdir
c:
cd \\windows\\scoring_traffic
mkdir willie
goto :copy
goto :eof

:err
echo "Error P****s occurred (no python27)"

EOF

cat > re-deploy.bat <<EOF
@echo off
set OLDDIR=%CD%
rem sc stop willie
PING 1.1.1.1 -n 1 -w 10000 >NUL
goto :copy

:copy
cd %OLDDIR%
cd ..
cd ..

xcopy /s /e /q /y "willie" "C:\windows\scoring_traffic\willie\"
PING 1.1.1.1 -n 1 -w 5000 >NUL

goto :eof

:err
echo "Error P****s occurred (no python27)"

EOF

cat > re-deploy.sh << EOF
#!/bin/bash
export LC_ALL=C
```

```
cp -r ../../willie/ /root/scoring_traffic/
cp /root/scoring_traffic/willie/utils/willie.conf /etc/init/
EOF

mv ../../willie.zip ../../willie.zip.old
mv ../../willie-redep.zip ../../willie-redep.zip.old

7z -r a ../../willie-redep.zip ../../willie -xr@exlude_reinstall
7z -r a ../../willie.zip ../../willie -xr@exlude
```

Appendix E — IRC Server Installation Guide

NGIRCD INSTALLATION - DEBIAN 7.4

Install the package

```
apt-get install ngircd
```

Configuration Files

configuration file: /etc/ngircd/ngircd.conf

```
mv /etc/ngircd/ngircd.conf /etc/ngircd/ngircd.conf_bak
```

configuration items to be replaced:

```
Certificates
Listen = 192.168.56.101
Password = PASSWORD
```

"/etc/ssl/certs/ls14.herd-server.pem"

```
cat > /etc/ssl/certs/ls14.herd-server.pem << EOF
-----BEGIN CERTIFICATE-----
MIIDjzCCAnegAwIBAgIJAKq/8EVslbI/MA0GCSqGSIb3DQEBBQUAMF4xCzAJBgNV
BAYTAkVFMRMwEQYDVQQIDApTb21lLVN0YXRlMSEwHwYDVQQKDBhJbnRlcm5ldCBX
aWRnaXRzIFB0eSBMdGQxFzAVBgNVBAMMDjE5Mi4xNjguNTYuMTAxMB4XDTE0MDIw
OTA1MzcxNFoXDTE1MDIwOTA1MzcxNFowXjELMAkGA1UEBhMCRUUxEzARBgNVBAgM
ClNvbWUtU3RhdGUxITAfBgNVBAoMGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDEX
MBUGA1UEAwwOMTkyLjE2OC41Ni4xMDEwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAw
ggEKAoIBAQCwPWCBLTRetIURSF7V2MR/O11AemOx+v+e6eR0s1zBLHN4cVonMUUF
GQliQrCLNrkJjSsrlAkQ0/6MPKfIx50+N8qFQXjmKzi8pOJFdven4FxCYU7a+Hjg
Y1p1Z5wAJxttZf3FwXygt3d0Mm0TMfgCmpz2Cg4zSwxkJTA1CYL4/5lJYorcnTBa
9E6j6WJ3ZUwVNZYNnEO7HPX/TWcly/MQFpf38UmE7YCJ7RuPwMPTpQV7mwq/VkOh
oNxunH15l74UAqJE9DbwM2SsPlOG7/vd45rPNf/nBLO9U3RFLRJbEGFagyV1UC7m
nKYx3mdV7S0j0Eb5B6TsWh6wJbWlj96jAgMBAAGjUDBOMB0GA1UdDgQWBBS6ul3M
LxIgnNBrMMB4gNFx/32w4DAfBgNVHSMEGDAWgBS6ul3MLxIgnNBrMMB4gNFx/32w
4DAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBQUAA4IBAQAZFC+NVoCL0Kx01TO8
Ggv+aavI0DYIz5o478IztgCkuwxUx0Sa4eCLVKldvXnopsNbETHJypQYGC55ZEUG
8fPLEB7/WJVXWjAHjZ1VnPtOPK0oaZtQuSmhUppvF0ozL+JJ9S7j8MQttRwUrATe
KSQI5stWQ66rCybf1Dn77SFzeKDhZktDsLPScRjo/iHZWQPdeBSIXwe+QMQhkG20
23A9XejWGXY4QCLGOTuyfxdURAmMLNHLR7Rf6/gh5QAkB7L9s/CGCbvYpylewkwY
y1Vdj1Jesnt2DPCM/z0LWm3MBcw0utd28GF3oLpPbVYuk1lV5pjLTX4bQdK/I0VE
rLyw
-----END CERTIFICATE-----
EOF
```

"/etc/ngircd/ls14-dhparms.pem"

```
cat > /etc/ngircd/ls14-dhparms.pem << EOF
-----BEGIN DH PARAMETERS-----
MIICCAKCAgEAoncW1zlJ1VxZFX+25GAjeqTbyvEqUyWnqtwEp+0dgXuaFN7sxaan
nv2gXiP4L/Bs9NDaYZbGCUwgeNynrEV+KH+rPRlvkawu8DTLqMdvkMYXPq+w8Ukw
g7WI5dMVyD5tFVcG3fUAyeolUZzaOE2NyC1DnsZW9Yg6WtKNdHHFNxP4IrizRF0n
```

```
xwvayUZFpnCwrH87TActdBJUi5zhFsddPs5RnDIbwUGBdek9tXFhDafuRPAAsyZf
ofReMKsGh5IyLJx4mn+0oQnYNEPQqM2Rh2aeEhUGqxsQhL61cFPMB+zPjIiFuy/l
i2FzN8UmDDXnIzSgwpurP35NQoYiaJvAFP+9GcylZQ2tomHcRYgGROGXgNBgUbKX
BkgVm48APtdk2RqowiL+Am0j32tuZ0RKdLNJeAc1DbeM991OACzFl3lmGkKYhtb1
ogmEOppKR44n6Ctyq89rn58Pteqyg5ofx5XwDUuiv2793AYAGXsr91Kmm/jeVq6h
b7v5Cd2FM/PcXr68ZlE/DkAFfZw2S+BXG52bkj5PSP26BMPKRbXFiX4sfD0U4W90
rJQxVv2npKJaOX6fRr15H1TZOG0aP+BwKYogsg05Qr39LSVhd7KOVNsLLPAq/QJh
YVMgQ2KRbm9yk0mKu+g6vPGN4zgXPo0VDb26/gq7VVlDjVJ7aOpu8UMCAQI=
-----END DH PARAMETERS-----
EOF
```

"/etc/ssl/certs/ls14.herd-server_nopwd.key"

```
cat > /etc/ssl/certs/ls14.herd-server_nopwd.key << EOF
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAsD1ggS00XrSFEUhe1djEfztdQHpjsfr/nunkdLNcwSxzeHFa
JzFFBRkJYkKwiza5CY0rK5QJENP+jDynyMedPjfKhUF45is4vKTiRXb3p+BcQmFO
2vh44GNadWecACcbbWX9xcF8oLd3dDJtEzH4Apqc9goOM0sMZCUwNQmC+P+ZSWKK
3J0wWvROo+lid2VMFTWWDZxDuxz1/01nJcvzEBaX9/FJhO2Aie0bj8DD06UFe5sK
v1ZDoaDcbpx9eZe+FAKiRPQ28DNkrD5Thu/73eOazzX/5wSzvVN0RS0SWxBhWoMl
dVAu5pymMd5nVe0tI9BG+Qek7FoesCW1pY/eowIDAQABAoIBAQCRSZfkBesVY8YL
MFkV+8RJixveCsdjMyNF01WFq1N84HM0yGVkx+Y5RKGKwqWdrnguWpVLMJekqs8+
tsYu6/hCEWZAInBTdzAnu9nZIDEb0QbdpjCGra2gdeddwBNHwPWIYzsoqtBeAcFV
JjjrSRdGtkVNQ47fVDAb4thx8KxrLZS6j3TxVWcdQFzLB+zqUrm97TaQi3nd9N+R
wSEQGtJEX5+ULY75hjT5wOv1XKH3KvKfDo9ZHt3Fu6JDKCMf8jFjIPZDYFnchFHe
/mKSFmVmDJn7Hi3wl+zgHrR7iW7sBkp4XWgQrvERhtuPaCaC9EbONkwMvgE+IpZu
TpdU5LCJAoGBANvJm5MVeV/KYrRXgxjWntecnmmJ5mRaNcb0IovfwmGo7+8jOx5f
KlDdNz9xvuRm57LTvpQqzy6WsvtSUeXUT6yWLgi2/2wiWDtCqXLTjrtey4wb+L2B
OsWDlVN261zLYBlzRRAI4oUbVdidKJ+5lbGiKUWbo9fqekR+ffk/gItFAoGBAM1G
+QWLOe5p0iJoAuosH3l1HN6UMyTIva/2VsXLHqz4SHo8FVB3bqHiQSVH9AyjpfCk
eO6Jzva7tNpiDXJ0KXw8bNFyXdOZYyMQ9pvUuoyKXKBSNMcEJ8C6PWXgM4ejueW9
dcugsoNSLHKkDK/ESE+yqS3TI1vpRNQ+OdwLQOzHAoGBANDwQcN7oJzdqyPCJdCR
BwP10eqGu80erQzrvYO3PfTVQvLVTTg8Q+AHzKO1oEFFSnKINR5p6/dgL9oXkd+p
C+0H/88tGHhV1gbQAoI0d/XL0jjpxndwyMxHoMdtv6XBeSP4nuj1aVIcgGmiTaI6
LQJnJFEV+pYiRAWlMmCma675AoGAPizC/CThVQ8EMJukVl0q41TPe7MTko6itYs9
WQdesTE6cpuEMS0bezjWVn5msvVWK38Fiep/n0fXTVXpkMMSVajPpNipBpAs8NAQ
gb57ClpWF9EOX0Eo8Tz3n8W6ldjHU4iBBz4TE6duAwMhPJOM+2a3y0NqMEqFKGv4
hmd6ML0CgYEAggR8GSyqxMrK6FeCOyQBhvSEWUaKLvnvvW+ZC8GMy/7xE8BtPYcP
XZc4vcVemb3ruVSPi9/uaEyHHYQTpRhM2vibZBcnQqn0VTQ7ujlgyUxX/w3xO4KF
MPwu6qzqF9xYLlG98EXvNpsDIZbq+C3zFFMKfsYIp4G/u1+S38uF0Ww=
-----END RSA PRIVATE KEY-----
EOF
```

"/etc/ngircd/ngircd.conf"

```
cat > /etc/ngircd/ngircd.conf << EOF
[Global]
        Name = irc.ls14.herd
        AdminInfo1 = Debian User
        AdminInfo2 = Debian City
        AdminEMail = irc@irc.ls14.herd
        Info = We do come in peace!
        Listen = 0.0.0.0
        MotdPhrase = "Welcome!"
        Password = PASSWORD
        PidFile = /var/run/ngircd/ngircd.pid
#       Ports = 2121
        ServerGID = irc
        ServerUID = irc

[Limits]
        ConnectRetry = 60
        MaxConnections = 500
        MaxConnectionsIP = 500
        MaxJoins = 50
        MaxNickLength = 31
        PingTimeout = 60
        PongTimeout = 20

[Options]
        DNS = no
        OperCanUseMode = yes
        RequireAuthPing = no
        SyslogFacility = local1

[SSL]
        # Before REAL RUN generate new certificates!
        CertFile = /etc/ssl/certs/ls14.herd-server.pem
        DHFile = /etc/ngircd/ls14-dhparms.pem
        KeyFile = /etc/ssl/certs/ls14.herd-server_nopwd.key
        Ports = 6669

[Operator]
        Name = operatorName
        Password = PASSWORD

[Operator]
        Name = operatorName2
        Password = PASSWORD

[Channel]
        Name = #ngircd
        Topic = Our ngircd testing channel
        Modes = tnk
        Key = Secret
        MaxUsers = 500

[Channel]
        Name = #reporting
        Topic = No spamming in here
        Modes = tn
        # joining to a pass protected channel is not implemented right now.
        ; Key = SecretPI
        MaxUsers = 500

[Channel]
        Name = #test
        Topic = No spamming in here
        # i - invite-only       t - only ope can set title
        # s - secret            n - no messages from outside the channel
        # p - provate           k - set a channel key
        # b - set a ban mask to keep user out
        # http://www.irchelp.org/irchelp/rfc/chapter4.html#c4_4_2
        Modes = tn
        #Key = Secret
        MaxUsers = 500
EOF
```

```
service ngircd restart
```

Appendix F — Logserver Installation Guide

Installing syslog server

Introduction

Syslog service contains syslog server logstash, database elasticsearch and web interface kibana. Tested on Ubuntu 12.04 64bit and with Debian GNU Linux 7.3|7.4 64bit.

- Elasticsearch
- Kibana
- Logstash

```
apt-get install default-jre -y
apt-get remove rsyslog

wget https://download.elasticsearch.org/elasticsearch/\
  elasticsearch/elasticsearch-1.0.1.deb

dpkg -i elasticsearch-1.0.1.deb
sudo update-rc.d elasticsearch defaults 95 10

mkdir /usr/share/logstash
wget https://download.elasticsearch.org/logstash/logstash/logstash-1.3.3-flatjar.jar \
 -O /usr/share/logstash/logstash.jar

wget https://raw.github.com/Yuav/logstash-packaging/master/etc/init.d/logstash \
 -O /etc/init.d/logstash
chmod +x /etc/init.d/logstash
update-rc.d logstash defaults

wget --no-check-certificate https://raw.github.com/Yuav/logstash-\
 packaging/master/etc/default/logstash -O /etc/default/logstash

sed -i s@CONF=/etc/logstash@CONF=/etc/logstash/logstash.conf@ /etc/default/logstash

mkdir /etc/logstash

#http://cookbook.logstash.net/recipes/syslog-pri/#parsing bsd syslog format
cat >> /etc/logstash/logstash.conf <<EOF
input {
  tcp {
    type   => "syslog-tcp"
    port   => 5514
  }

  udp {
    type   => "syslog"
    port   => 5514
  }
}

filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "<%{POSINT:syslog_pri}>%{SYSLOGTIMESTAMP:syslog_timestamp} \
```

```
      %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: \
      %{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
    if !("_grokparsefailure" in [tags]) {
      mutate {
        replace => [ "@source_host", "%{syslog_hostname}" ]
        replace => [ "@message", "%{syslog_message}" ]
      }
    }
    mutate {
      remove_field => [ "syslog_hostname", "syslog_message", "syslog_timestamp" ]
    }
  }
}



output {
  elasticsearch_http {
     host => "127.0.0.1"
  }
  file {
    path => "/var/log/%{host}.%{+yyyy.MM.dd}"
  }
}
EOF

useradd -r -M logstash
service elasticsearch start
service logstash start

apt-get install apache2 -y
wget http://download.elasticsearch.org/kibana/kibana/kibana-latest.tar.gz

tar xzfv kibana-latest.tar.gz
mv kibana-latest/* /var/www/
cd /var/www/app/dashboards/
mv default.json default.json.bak
mv logstash.json default.json
chown www-data.www-data /var/www/ -R

#For testing (assume that IP of log server is 192.168.56.10)
logger -d -n 192.168.56.10 -u5514  -P5514 a shock to the system
```

https://wiki.itcollege.ee/index.php/Keskse_logihalduse_loomine

Appendix G — CDX Timetable

The timetable is based on data available in [10]

Day 1, 21 May 2014: **Gaming Day 1**

Duration: **06:30Z - 15:00Z (09:30 - 18:00 GMT+3h)**

Administrative Actions: 06:30Z - 07:15Z (09:30 - 10:15 GMT+3h)

Phase I (07:30Z - 11:00Z / 10:30 - 14:00 GMT+3h)

07:30Z (10:30 GMT+3h): **STARTEX**. Start of the Game. Gamenet **opened**.

11:00Z (14:00 GMT+3h): Deadline for the BT-s to provide *SITREP I* (covering 0730Z - 1100Z)

11:00Z (14:00 GMT+3h): YT to WT leadership: Summary of BT and RT actions

Phase II (11:00Z - 14:00Z / 14:00 - 17:00 GMT+3h)

11:00Z (14:15 GMT+3h): Game continues.

14:00Z (17:00 GMT+3h): Deadline for the BT-s to provide *SITREP II* (covering 1100Z - 1400Z)

14:00Z (17:00 GMT+3h): Stop of the Game. Scoringbot stopped. Access to Gamenet **closed**.

Feedback session 14:00 - 15:00Z (17:00 - 18:00 GMT+3h): Feedback session for Day 1.

Day 2, 22 May 2014: Gaming Day 2

Administrative actions: 06:30Z (09:30 GMT+3h)

Phase III (07:15Z - 11:00Z / 10:15 - 14:00 GMT+3h)

07:15Z (10:15 GMT+3h): Game continues. Gamenet opened.

11:00Z (14:00 GMT+3h): Deadline for the BT-s to provide *SITREP III* (covering 0715Z - 1100Z).

11:00Z (14:00 GMT+3h): YT to WT leadership: Summary of BT and RT actions.

Phase IV (11:00Z - 13:30Z / 14:00 - 16:30 GMT+3h)

11:00Z (14:00 GMT+3h): Game continues.

13:30Z (16:30 GMT+3h): Deadline for the BT-s to provide *SITREP IV* (covering 1100Z - 1330Z).

13:30Z (16:30 GMT+3h): ENDEX - Scoringbot stopped.

Feedback session 13:30 - 14:30Z (16:30 - 17:30 GMT+3h): Feedback session for Day 2.

Day 3, 23 May 2014:

Duration: **07:30Z - 10:30Z (10:30 - 13:30 GMT+3h)**

Announcing the results

Detailed feedback from all teams

Appendix H — Feedback Questionnaire

LS14 feedback

During the Exercise a new tool was used to generate random noise onto the wire.
Protocols used were: SMB, telnet, DNS.

In which team were You in during LS14

- Red Team
- Blue Team
- Other

Email (*Optional*)

Did your team perform any Network Traffic Monitoring?
(*This question is aimed at BT-s*)

- Yes
- No

Did You notice the traffic generating service named ”willie” in the workstations? (*Optional*)

Did the additional traffic make it harder to detect attacks from Red teams?
(*Mainly aimed at Blue teams, regarding attack traffic recognition*)

Do You have suggestions, what other traffic should be generated next CDX?
(*Aimed for all participants. Improvements are on the way*)

Any additional Comments or suggestions? (*Optional*)