TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science
TUT Centre for Digital Forensics and Cyber Security

ITC70LT
Taavi Sonets 132702IVCM

IMPROVING USER SIMULATION TEAM WORKFLOW IN THE CONTEXT OF CYBER DEFENSE EXERCISE

Master's thesis

Supervisors
Elar Lang, MSc
Rain Ottis, PhD

Tallinn 2016

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.

Taavi Sonets

May 25, 2016

.....

(Signature)

Abstract

Cyber Defense Exercise is a good way to train for the cyber incident handling in a real life situation. Improving the efficiency of different teams in Cyber Defense Exercise is very important in order to keep the exercise scalable. The main purpose of this thesis is to improve the workflow of the User Simulation Team in the context of Cyber Defense Exercise by reducing manual labor. This is achieved by designing and implementing a proof-of-concept framework for enhancing the key aspects of the User Simulation Team's workflow with technical solutions in the exercise Locked Shields 2016.

The design behind this framework is devised by conducting semi-structured interviews with the key members of the different organizing teams for the exercise and extracting their suggestions of which parts of the User Simulation Team workflow should be improved.

Based on this design the first implementation was deployed for the *Test Run* event of the annual Locked Shields exercise. The finalized iteration of the solution was deployed for the *Execution* event of the same exercise.

The evaluation of the framework was conducted based on data gathered from the framework during the evaluation events and feedback gathered from semi-structured interviews with the User Simulation Team and Red Team Client Side attacks sub-team leaders. Additional features were suggested based on the evaluation for future development.

The evaluation shows, that it is effective to use technical solutions to improve the workflow of User Simulation Team in the context of Cyber Defense Exercise.

The thesis is written in English and contains 70 pages of text, 7 chapters, 19 figures.

Annotatsioon

KASUTAJA SIMULEERIMISE MEESKONNA TÖÖVOO PARANDAMINE KÜBERKAITSE ÕPPUSE KONTEKSTIS

Küberõppus on hea moodus harjutamaks küberintsidendile reageerimist. Erinevate küberõppuse meeskondade töövoo parandamine on väga oluline tagamaks õppuse skaleeruvust. Antud lõputöö põhieesmärgiks on parandada kasutaja simuleerimise meeskonna töövoogu vähendades käsitsi tehtava töö mahtu. See saavutatakse töötades välja ja rakendades konseptsiooni tõestav raamistik, mis parendab küberkaitseõppuse Locked Shields 2016 kasutaja simuleerimise meeskonna töövoo põhi aspektide täitmist läbi tehniliste lahenduste.

Raamistiku arhitektuur ehitatakse üles lähtudes soovitustest, mis on saadud läbi poolstruktureeritud intervjuude harjutust organiseerivate võtmeisikutega erinevatest meeskondadest. Intervjuude põhirõhk oli teemal, milline osa ja kuidas kasutaja simuleerimise meeskonna töövoost peab saama täiustatud.

Lähtudes välja töötatud arhitektuurist, loodi esimene raamistiku implementatsioon õppuse Locked Shields *Test Run* (prooviõppus) ürituse ajaks. Lõplik versioon raamistikust implementeeriti sama õppuse *Execution* (päris õppus) ürituse ajaks.

Raamistiku hindamiseks kasutatakse õppuse käigus kogutud andmeid. Lisahinnangu andmiseks kasutatakse punase meekskonna kliendipoolsete rünnete meeskonna ülema ja kasutaja simuleerimise meeskonna ülema poolstruktureeritud intervjuudest kogutud tagasisidet. Lisaks toodi välja erinevaid võimalusi praeguse lahenduse täiendamiseks.

Raamistiku hindamine näitab, et küberõppuse kasutaja simuleerimise meeskonna töövoo efektiivsus tõuseb läbi tehniliste lahenduste rakendamise vähendamaks käsitsi tehtava töö mahtu.

Lõputöö on Inglise keeles ja sisaldab teksti 70 leheküljel, 7 peatükki, 19 joonist.

List of Acronyms

LS Locked Shields

CS Client Side

NIC Network Interface Card

SINET Simulated Internet

VNC Virtual Network Computing

NATO North Atlantic Treaty Organization

CCD COE Cooperative Cyber Defence Centre of Excellence

RDP Remote Desktop Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

npm Node Package Manager

OS Operating System

UST User Simulation Team

CDX Cyber Defense Exercise

VIX Virtual Infrastructure eXtension

API Application Programming Interface

NET Network

SITREP Situation Report

VPN Virtual Private Network

IT Information Technology

SSH Secure Shell

HTTP Hypertext Transfer Protocol

TCP Transmission Control Protocol

Contents

List of Figures

Code examples

1. Introduction

The technology is transforming our world into something that we do not recognize yet and it is doing so quickly. Cyber and Internet have become part of people's everyday lives. The number of Internet users has gone from 502 million in 2001 to 3.34 billion by April, 2016 [1]. With such growth, the topic of security in cyberspace becomes important. Yet, it is not only the average citizen, who has become more integrated into technology. It has been found happening also on the country level. As the time goes by, more and more of countries' critical infrastructures become digitalized. The understanding of what cyber defense actually is, has never been so important as it is today.

The process of development and furthermore implementing cyber security in the national domain is slow [2]. The speed of cyber defense oriented improvements may vary in different countries. International cyber security is inherently even slower, since it usually involves figuring out cyber defense on national level first and then cooperating it between the nations. Not many ways exist of how a country could improve its own cyber defense and furthermore develop international cooperation that would be required in real life international cyber incident. One of those few ways is Cyber Defense Exercise (CDX).

> *Cyber exercises are an important tool to assess the preparedness of a community against cyber crises, technology failures and critical information infrastructure incidents. [3, p. 1]*

Cyber exercises can generally be divided into several categories: capture the flag, discussion based game, drill, red team versus blue team, seminar, simulation, table-top and workshop [4, p. 18-19]. Out of those types of exercises, this thesis mainly focuses on *simulation* type of exercise where the participants have to react to situations presented in the exercise in real time as they arrive [3, p. 9].

1.1 Locked Shields as an Example of Cyber Defense Exercise

North Atlantic Treaty Organization (NATO) Cooperative Cyber Defence Centre of Excellence (CCD COE) held Locked Shields (LS) CDX is one of the biggest and most advanced of its kind [5]. The first exercise of the series was held in the year 2010 and back then it was called Baltic Cyber Shield [6]. There was no exercise in 2011 and since the year 2012 the exercise is known as Locked Shields. The exercise has grown in size. It started out with 6 Blue Teams and 20 Red Team members attacking them in the year 2010. In the year 2016 this number was 60 Red Team members against 20 Blue Teams (see appendix A.2). The majority of this section of the thesis is based on the NATO CCD COE published Locked Shields After Action report 2013 [7] and the author's own personal experience with participating in the exercise in the years 2015 and 2016.

1.1.1 Teams in Locked Shields

The LS has four organizing teams responsible for its execution: red, white, green and yellow. Furthermore each year there is a growing number of target training audience, the Blue Teams.

1.1.1.1 Blue Team

Blue Teams are made up from the participating countries' specialists. They take part of the event from remote facilities (typically in their own country) and they connect to the exercise over a Virtual Private Network (VPN) connection. Their tasks are:

- Secure the provided virtual Information Technology (IT) infrastructure and defend it against the Red Team attacks;
- Maintain services described in the exercise documentation assuring the availability, confidentiality and integrity of the systems;
- Report detected incidents to the White Team through continuous lightweight reporting and management level Situation Reports (SITREPs);
- Complete business tasks injected by the White Team;
- Respond to information requests from the media sub-team.

The IT infrastructure that the Blue Team has to defend, is developed by the Green Team. It contains pre-planted security vulnerabilities. Blue Teams are allowed to use their own tools to secure the infrastructure [7, Blue Teams, p. 66].

1.1.1.2 Yellow Team

The Yellow Team is there for situational overview and awareness. Furthermore they are intermediate for the data flow between different teams. For example, they collect the lightweight reports provided by the Blue Teams to the White Team, statuses of attack campaigns from the Red Team and the results of manual and automatic scoring. Using this data they prepare different views and visualizations of the exercise current situation [7, Yellow Team, p. 9].

1.1.1.3 Green Team

The Green Team is responsible for the technical infrastructure of which the game is played on. They need to make sure, that from the technical perspective, game goes without any failures. They are also responsible for designing and setting up the core infrastructure such as physical devices, virtualization platform, storage, networking, remote access, traffic recording, VPN routers for the Blue Teams user accounts, etc. Further more they design and build the Gamenet [1] and Blue Team systems, program the automatic scoring bot and agents. They also develop solutions for traffic generation and set up monitoring the general exercise infrastructure [7, Green Team, p. 9].

1.1.1.4 Red Team

The Red Team is there to simulate the attacker. They are responsible for penetrating and degrading the Blue Team's systems in order to simulate a live opponent. They execute the predefined objectives in a specific time frame. Objective, in the context of LS Red Team is a attacking task for scenario play. Objectives are executed in accordance with the scenario dictated by the White Team. Red Team does not just attack everything they can, rather the attack plan is highly controlled in order to preserve fair game towards all Blue Teams.

[1] Gamenet is the whole simulated network in the context of CDX. It includes Simulated Internet (SINET) and all the other virtual networks in the game.

To achieve this controlled execution, the Red Team is made up of three sub-teams (see appendix A.2):

- WEB attacks team;
- Network (NET) attacks team;
- Client Side (CS) attacks team.

The WEB attacks team is responsible attacks against web applications that the Blue Teams have to defend. They include attacks such as defacements and bypassing authentication in the web application to access restricted data or functionality (see appendix A.2).

The NET attacks team is responsible for attacks that happen in the network layer (see appendix A.2). They can include attacks such as sniffing the network traffic to steal some secret data traveling through the network, denial-of-service attacks to render some part of the targeted Blue Team systems unavailable or man-in-the-middle attack, where an attacker is located between the targeted Blue Team client and a service. In case of a successful man-in-the-middle attack the parties targeted, might not realize with whom they are exchanging data with [8].

The final sub-team in the Red Team is CS attacks team. They are responsible for attacking the Blue Teams' workstations. They do this initially with the *white box* method, where all the vulnerabilities of the workstations are known to the CS attacks team. Those vulnerabilities are meant to be discovered by the Blue Teams and for the full course of the exercise, they are not a viable method (see appendix A.2). To simulate the real world scenario, the CS team uses user-interaction to gain initial foothold to the system. Similar to the real world scenario, they will prepare a piece of malware and smuggle it inside the Blue Teams defenses with the method that involves user-interaction such as enticing them to click a link, open a document or get them to open a malicious website [9].

1.1.1.5 White Team

The White Team responsibility is to manage the game play itself. To work out the exercises scenario and enforce that the game is played according to said scenario. They are the ones who define the training objectives, high-level objectives for the Red Team, write the rules, prepare media, scenario, legal injects and the communication plan. During the execution of the exercise, they are the exercise manager cell. They make scoring

decisions and they control the execution of the Red Team's campaign. They also include liaison officers, who are the intermediates between the organizers and the target training audience the Blue Teams [7, White Team, p. 8].

The structure of White Team, provided by Aare Reintam in appendix A.1, is built up as follows:

- Exercise control;
- Communications team;
- Injects and scoring;
- User Simulation Team (UST).

Exercise control is responsible for running the exercise and making decisions such as when to start certain phases. Phases are sections of the exercises time line that the game play is organized into. The communications team is there to prepare the communication plan and coordinate the work, they also communicate the feedback to and from the Blue Teams. The injects and scoring is the team that plan and manage scenario injects for the game. The two additional sub-teams in the injects team are the media and legal sub-teams. They are there to provide related injects to their field and also to evaluate the responses to said injects.

The main purpose for UST is to play the role of the users in the simulated organization. Depending on the format of CDX, this can involve a various of tasks. For example, in case of a exercise with very narrow scope, this could mean, that the whole team periodically executes a single type of task, such as entering the workstations in the morning, sending some emails during the day or perhaps leaving a workstation unattended for the duration of the lunch. On the other hand, in case of a full-scale live fire simulation where there is an active Red Team, they must work very closely with the Red Team to assist them on their campaign of penetrating the target training audience's systems. They are the ones who execute user-interaction tasks for the CS attacks inside the Blue Teams workstations.

1.1.2 Tools used in Locked Shields

When the exercise grows into some certain parameters, the need for efficiency in communication between teams also grows. In many cases the speed of growth is not the same in every aspect of the game [7]. To manage the communication between the

teams, different tools are used. There is a **Collaboration Environment**, that serves as a main channel for information exchange. It is a MoinMoin Wiki [2] type of platform, where different teams exchange long haul information, such as development instructions, ruleset for the Blue Teams, time schedule of the exercise, passwords for various services in the SINET and much more. For more rapid communication there also exists a **Jabber XMPP chat** [3]. This is for fast and critical information exchange mainly throughout the execution phase of the exercise. For example it is used between the Red Team and the UST to exchange malicious Uniform Resource Locators (URLs) that the UST members have to visit to serve the Red Team the user-interaction part of a CS attack.

1.2 Main Problems

Each iteration identifies new shortcomings and features for the organizers of the exercise that could be improved or implemented. In case of UST, there still are problems that require solving. UST members have various tasks to complete and not much time to accomplish them. The main tasks outlined by the exercise director, Aare Reintam, for the UST in the order of importance are [10]:

- Provide user-interaction inside Blue Team workstations for Red Team CS attacks sub-team;
- Conduct service checks, to verify that the Blue Teams are providing services and that the services are still usable and available for the UST;
- Demand defending teams to hold up the services and provide relevant scoring regarding that.

The tasks involve manual labor, communication between teams and are mostly prone to time consuming errors - time being one of the most expensive and scarce asset in the context of live fire exercise [6].

1.2.1 Providing User-Interaction Inside Blue Team Workstations for Red Team Client Side Attacks

The most important task of a UST member, is to execute the malicious code (also known as payload) that they receive from the Red Team inside the Blue Team systems. This

[2]Read more about the MoinMoin Wiki engine from the project homepage: `https://moinmo.in/`
[3]Read more about Jabber chat from `http://www.jabber.org/`

is important for the Red Team CS sub-team in order to simulate similar to the real world threat actor (see Appendix A.2). Since the communication of this process happens through Jabber chat client without any verification or validation for error in human input, then mistakes are prone to happen. In addition to code execution, the UST is required to also give feedback for the task given them by the Red Team. Initial feedback of acknowledging that the task has been received can take up to 15 minutes [11].

The main problem for UST, in previous iterations of the exercise, was that the copy-pasting between UST member workstation and virtual Blue Team workstation did not work reliably [10]. This resulted in errors and consumed additional time to fulfill the Red Team request by manually retyping the complicated URLs and commands. Furthermore, there were cases where Blue Team had set a complicated passwords to access the targeted workstation.

Other aspect that has to be considered is, that every year there are new UST and Red Team members who have no experience in working with each other. This is true even more for the UST, where in the case of LS 2016, there were 26 members in UST. All of them were newcomers to the team, meaning that they had no previous experience of participating in the UST. This has in previous exercises in LS series generated communication errors that led to inefficiency in UST and Red Team cooperation [12] [6].

1.2.2 Conducting Service Checks

UST member has to check the availability of different services from workstations. More specifically they have to check if the service is usable and the functionality has not been tampered with. In case of 2016 LS there were in total twelve services that had to be checked. In the year 2015 exercise, this number was eleven [10]. It is not enough to check availability and functionality of those services from only one workstation. In case of LS 2016, UST had 34 Windows, 10 Linux and 2 MacOS workstations for each Blue Team from where services had to be checked [13]. In an ideal case, UST member would check all the services from all of the workstations. In reality, as this is a secondary task for the UST, it was expected they could verify the services from each of the network segment in the simulated network. Furthermore, as the exercise grows, so does the complexity of it and with that the workload for every team participating in the successful execution of the event, including UST.

Most of the services that have to be checked for availability and functionality require a log in procedure with given credentials that Blue Team is allowed to change. This means

that every time the UST member wishes to authenticate their self to the service, they have to first log into the Collaboration Environment where the user credentials are stored in case of LS CDX and retrieve the correct credentials. This is prone to produce errors and also will make the whole process time consuming. Since the Blue Teams are allowed to change the credentials for those services as well, the verification of correct passwords becomes one of the subtasks for the UST [12].

1.2.3 Demanding the Usability of Services

Third task in the workflow of a UST member, is to demand the Blue Team to keep the services up and running. If it is discovered, that a Blue Team is not keeping the services usable (see section 1.2.2), then the responsible Blue Team will need to be notified. If Blue Team has not managed to make the service available after some time, then they will be scored accordingly. Scoring is another important part of the UST work. They have to be fair and all the Blue Teams must be scored equally in the sense that the scoring frequency has to remain the same. From the LS 2015 scoring, 11 teams got less than 60% of the points for usability of the services, with none of the teams receiving over 75% of possible 100% which would have meant that services were available and usable for the UST during the full length of the exercise [14]. It can be deduced, that the availability and functionality of a service is not a priority for the Blue Teams, and needs to be checked. The more efficiently this can be achieved the better.

Since the task of demanding Blue Teams to keep the services up and running builds upon the task of conducting service checks and since according to Aare Reintam, this is the lowest priority task, then it was decided that solving the problems related to this task are out of the scope of this thesis and are left for future development.

1.3 Problem Statement

The problem this thesis will tackle is that the workflow of the User Simulation Team involves manual and time consuming labor. Furthermore it is prone to produce errors and miscommunication and there is not enough time to fully accomplish all the tasks trusted to UST.

1.4 Main Objectives

In the section 1.2, it was described that the general issue with the UST is inefficiency and problems developed from the manual labor that the team has to perform. To reduce the time consumption for various tasks of UST and free up resources for something more useful, such as communicating with the Blue Team, demanding them to keep up their services (see section 1.2.3) or assigning score then the problem prone tasks of UST workflow should be solved on technical level.

This thesis is focusing on reducing the overhead in the UST workflow through improving the efficiency of UST tasks with technical solutions. The problems emerged from the White Team feedback stated that one of the main issues came from inefficiency in the communication between the Red Team and the UST (see 1.2.1).

Another problem that presented itself was the difficult interaction with the workstations. In order to address those problems the flowing objectives are set:

- Propose a solution that reduces the manual work of UST;
- Propose a solution that reduces the manual work in Red Team and UST cooperation;
- Evaluate solution in the Locked Shields 2016 Cyber Defense Exercise.

1.5 Outline of The Thesis

This thesis is organized into chapters. **Chapter 1** gives the general overview of CDXs and explains the essence of a CDX in the example of Locked Shields CDX. In **Chapter 2** it is being discussed how currently the UST operates and what are the main difficulties relating to that. Furthermore what research has been done and how it relates to this thesis. Also the current state of the underlying infrastructure, that supports the development of the framework is presented. In **Chapter 3** the author analyzes what are the issues of UST, how they can be solved through technical solutions. The design for the framework is proposed based on the suggestions gathered from the key members of the sample exercise, LS. **Chapter 4** focuses on the implementation description of the design composed in the previous chapter. **Chapter 5** focuses on evaluation of the solution by reflecting on how the framework performed during the *Test Run* and the *Execution* events of LS 2016. Did it improve the situation that was there before and if so, then how. The author furthermore

explores what lessons were learned. In **Chapter 6**, the author brings out aspects that should be implemented or improved in the future regarding the developed framework.

The conclusion is disclosed in **Chapter 7**.

1.6 Acknowledgments

The author would like to give thanks to Elar Lang and Rain Ottis for tireless supervision of this thesis. Author expresses special gratitude towards Aare Reintam, Ragnar Rattas, Jarkko Huttunen, Elvis Paat, Heliand Dema and Mehis Hakkaja for their contribution and input during the research and development phase of this thesis. Furthermore author wishes to thank the entire Clarified Security team for their support.

2. Current Situation and Related Research

The exercise where the proof-of-concept solution was deployed is NATO CCD COE organized LS 2016 event. LS is a complex exercise. As discussed in the main problems before, previous years experience has shown that the efficiency of the UST could be improved. To achieve the task of improving the user simulation team workflow, LS offers a good testing ground. With its numerous staff and organizing members, LS has experience and insight to CDXs in a huge scale, being one of the biggest technical live-fire CDX in the world [5].

2.1 Related Work

As it was discussed in the first chapter, the highest priority task for UST is *Providing user-interaction inside Blue Team Workstations for Red Team Client Side Attacks* (see section 1.2.1). In order to find a technical solution to this task, to make the UST work more efficient, it can be deduced that a way of remotely executing code inside the workstations is needed.

2.1.1 Centrally Managed Network Traffic Generation for Cyber Exercises

In 2014 Erki Naumanis described in his thesis Centrally Managed Network Traffic Generation for Cyber Exercises, how he built a traffic generation framework for LS 2014. He found that the best way of interacting with the Blue Team systems is to deploy a centrally managed botnet in the Gamenet [15].

In his thesis, Naumanis decided to build a centrally managed botnet of traffic agents. A part of his process was to install botnet clients into every workstation in every network segment for every team. This task alone is time consuming and needs automation. Furthermore he decided to use Python programming language as an universal language

that would work on every operating system used in the LS exercise from where the traffic generation was required.

The validation method for Naumanis was to evaluate the preliminary solution in the LS 2014 Test Run and the final evaluation was done after the LS 2014 execution event. This two step validation proved to be successful and in this thesis the same methodology for validation will be used.

2.2 Current situation

This section of the thesis will give a overview of the current situation in underlying infrastructure and processes related to the UST workflow.

2.2.1 Infrastructure

LS exercise and all the other exercises held currently on top of Estonian Defense Forces' Cyber Range[1], are simulated on top of VMware virtualization platform vSphere[2] [6].

VMware itself provides a way of interacting with the guest operating systems by executing commands inside the guest machines through its Virtual Infrastructure eXtension (VIX) Application Programming Interface (API) [16]. This means that in case of managing the workstations in the virtual environment, the need to build a botnet inside the Gamenet is redundant as vSphere VIX API can run commands in three major virtualized platforms: MacOS, Linux and Windows [17]. The main difference between managing computers through VMware VIX API and an simple botnet - there is no need for reliable simulated network connectivity, since commands will be executed through VMware itself. This would allow technical solution of user simulation for those network segments, that are completely isolated from the rest of the network.

The VMware VIX API is not the only way of executing code in the target workstation. So far the Green Team has used a combination of different methods to execute the code inside the targeted workstation. For windows they have previously used PowerShell Remoting [3],

[1]Read more about Estonian Defence Forces' Cyber Range `https://www.mkm.ee/sites/default/files/cyber_security_strategy_2014-2017_public_version.pdf`

[2]Read more about vSphere `https://www.vmware.com/products/vsphere`

[3]Read more about PowerShell Remoting `https://technet.microsoft.com/en-us/library/dd819505.aspx`

PsExec [4] and to achieve some goals Windows Group Policies[5] [13]. Then again, all of them require network access from source computer to targeted computer that is not always a requirement for the UST workstations in the context of a CDX. The solution to overcome this issue has been of having a separate Network Interface Card (NIC) solely for management networking, that stood outside the SINET.

In addition to executing the code inside the targeted workstations of the simulated infrastructure, the UST member also has to retain some form of control over the task that is being executed. The visual feedback for code execution is still required. This is needed in the context of LS 16, to retrain the possibility of full control to the UST, in case the technical solution fails [10]. Although the command execution could be done solely in the background. To have visual feedback of the target simulated workstation, the current situation in case of LS CDX has two existing possibilities [13]:

- The Remote Desktop Protocol (RDP) protocol for Windows and Virtual Network Computing (VNC) protocol for MacOS and Linux operating systems;
- The other possibility that exists and has been used since the LS exercise started running on top of VMware is the vSphere's client application feature *console* [6] for interacting with the Virtual Machines.

Out of those two, first one requires still management network to be there, as the RDP works only over networking. The VMware console does not require additional management networking, to the targeted machines, but it does complicate some of the aspects of the automation. The RDP seems to be better documented and understood. The RDP has existed longer and is a native part of Windows operating system.

The VMware Console, as the name suggests, is developed by VMware. This entitles VMware console to work with the other components from the same company. For example the Console does not require the targeted workstations to have additional management NIC to interact with them. All communication between the client application and targeted workstation is trafficked in the same network that the VMwares vSphere application lives. This offers a layer of abstraction in the network and therefore, when

[4]Read more about PsExec `https://technet.microsoft.com/en-us/sysinternals/psexec.aspx`

[5]Read more about Windows Group Policies `https://technet.microsoft.com/en-us/windowsserver/bb310732.aspx`

[6]Read more about VMware vSphere console feature from `https://pubs.vmware.com/vsphere-51/topic/com.vmware.vsphere.vm_admin.doc/GUID-C127F9F9-3E09-4A3E-B368-4C46B2A02F8D.html`

it comes down to visual interaction with the workstation, removes the requirement of management network for workstation interaction to the UST.

In previous years, this abstraction has been the main reason why the VMware console has been the main choice of tool when it comes to interaction with workstations. Yet, there have been problems. The security and guest isolation is very important in case of all virtualization [18]. Therefore by default VMware vSphere disables *shared clipboard* between the guest and host operation system in order to isolate the layers of virtualization. It is in the development instructions of Green Team to enable it where needed. Yet, the process from development to execution of the exercise is taking place over an extended period of time, therefore it has happened in several occasions, that the *shared clipboard* feature gets disabled in some part of the process [13].

2.2.2 User Simulation Team Interaction with the Workstations

The previous year's White Team feedback stated that the interaction with targeted workstation was too difficult and time consuming in general (see figure 2.1) [10]. The UST had to use a RDP connection to enter remote VMware managing server. From there they had to start VMware vSphere client, and manually navigate to the correct workstation from the total 46 virtualized workstations of targeted Blue Team systems (see example in the figure 2.2). After doing that they had to open the interaction appliance, and perform the activity that was required of them. Difficulties affected the completion UST tasks mentioned in sections 1.2.1 and 1.2.2 as for both of them interaction with the workstations is required.

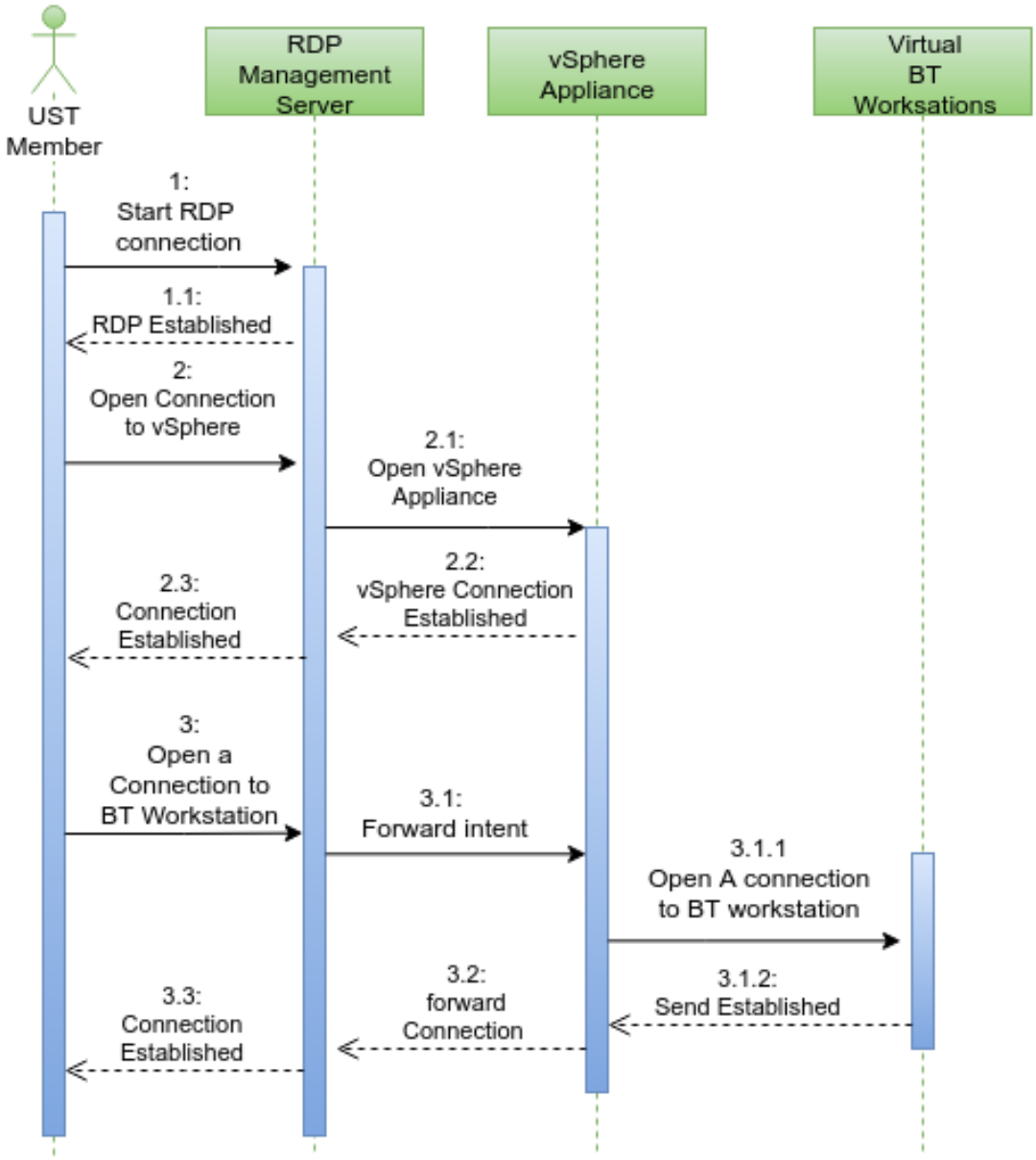

Figure 2.1: Interacting with Workstations in Previous Locked Shields Exercises

The virtualization platform that the LS is running on is VMware vSphere [13]. Considering that, one of the key requirements when building a target host, is to install a VMware Tools [7] software to enhance the performance and monitoring possibilities in the Virtual Machine [13]. As discussed in the beginning of last section (see 2.2.1), VMware virtualization gives the possibility to run guest Operating System (OS) commands in Windows, Linux and MacOS machines [17]. This process happens through VMware Tools service in the guest workstations and all of the workstations must run the VMware tools service [13].

[7]Read more about VMware tools from `https://kb.vmware.com/kb/340`

The benefit of the VMware virtualization is that it provides a possibility for developers to deploy and manage a hundreds or even thousands of Virtual Machines [13]. Yet, virtualized environment is not often built considering the needs of UST (see figure 2.2). For example in the context of LS 2016, the virtualized environment was built up of more than 1000 systems(see appendix A.3). There were Windows, Linux and MacOS workstations in addition to different servers and firewalls. To manage it properly various types of automation is built to handle the deployment and configuration process. This automation builds up the dependency tree structure in the environment considering its own needs, so that it might be difficult for a UST member, who has never seen the structure, to come and quickly find the machine that they need to interact with.

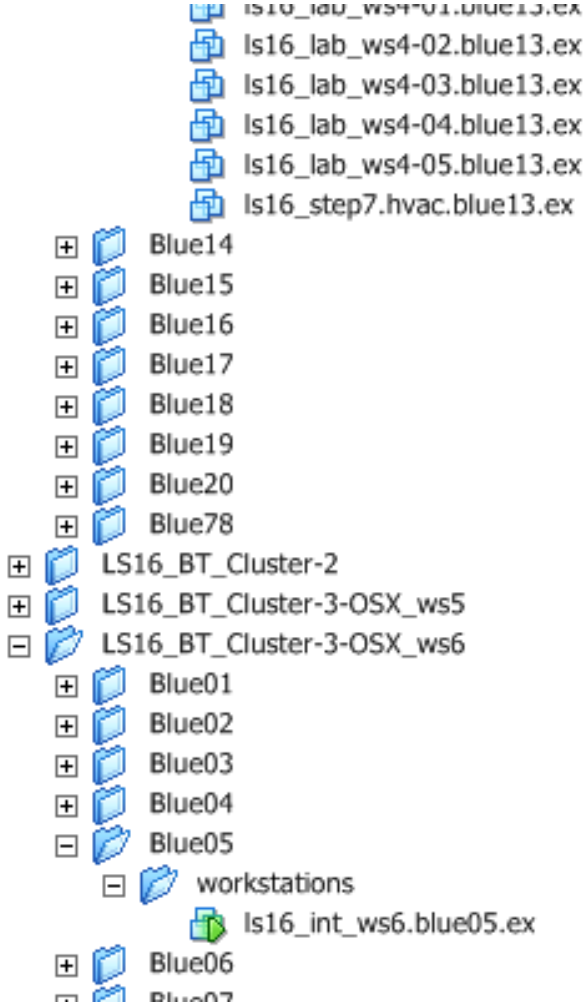

Figure 2.2: Example of VMware vSphere Virtual Machine Locations

In addition to the difficulty of finding the correct machine from the VMware structure tree, the inconsistencies in naming conventions also can contribute into the inefficiency of UST workflow. For example, when the Red Team

member asks the UST member to open a malicious web URL in the machine **xxx-machinename-networksegment-blueteamidentifier** while refereeing the format of the machine name defined in their own materials and the corresponding machine in the virtualization environment happens to be named anything else, such as in the format of **machinename-networksegment-xxx-virtualizationidentifier-blueteamidentifier**, then it might provoke certain level of confusion for the UST member and confirmation from the Red Team might be requested that leads to even more inefficiency.

2.2.3 User Simulation Team Communication with the Red Team

Current state of the communication is that when a Red Team CS attacks sub-team member has a task for the UST member, they use a common Jabber chat room to extend their requests in the format described in figure 2.3.

```
RT-Member: Hey, please download, unencrypt, unzip the file from http://
  malicious.url/malicious.file.zip with the password 'Password1234'in
  workstation XXX
UST-Member: ack.
... some time passes
UST-Member: Failed. Was unable to decrypt the file
RT-Member: The password does not include aphostropes
UST-Member: ack. Retrying
... some time passes
UST-Member: Success. Task executed
```

Figure 2.3: Example of Chat Between Red Team and User Simulation Team Member

Some of the UST members have not, in the previous exercises, been able to understand the Red Team requests correctly or the requests were not clear enough. This miscommunication between the teams consumed time [11]. In the LS14 and LS15 Red Team feedback it was described, that the main problem with UST in given years was miscommunication [6] [11]. UST was in cases unresponsive, uncertain of what to do or confused of what exactly is required of them. It took a considerable amount of time to get any kind of feedback from the UST. This resulted in many cases a Red Team CS attacks sub-team member physically tracking down the responsible UST member and identifying the state of a request [11]. It is very important for Red Team that the UST member is responsive throughout the entire communication process. The communication problems mainly affected the completion of task described in section 1.2.1, as this task requires communication between the teams.

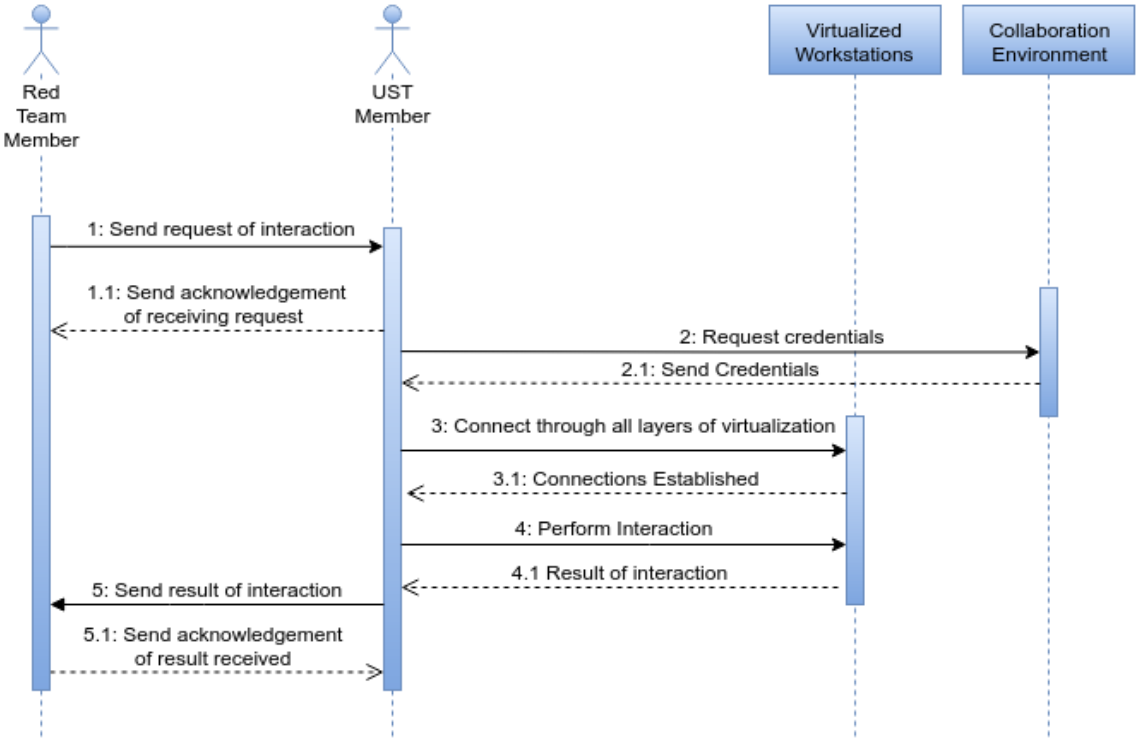

Figure 2.4: Handling the Red Team Request

In the figure 2.4 is described, that for UST to perform an interaction task requested by the Red Team, the UST member has to establish a connection to the workstations, through several layers of virtualization (see figure 2.1) . When the Red Team task arrives, the UST member first has to acknowledge that they have received the task, then they have to retrieve correct credentials, to authenticate themselves into the workstation, after that perform the task and send the results, if there are any, back to the Red Team member.

2.2.4 User Simulation Team Interaction with the Data

The UST has to handle several types of data throughout their workflow. Those types include but are not restricted to:

- Credentials for Blue Team systems;
- Tasks (also known as *request for interaction* in the context of this thesis) and their statuses from the Red Team;
- Statuses of the functionality and the availability for the virtual services of the Blue Team.

To handle all this data, currently the UST in the context of LS CDX uses set of tools mentioned in the paragraph 1.1.2. Those tools are Collaboration Environment and Jabber XMPP chat.

2.2.4.1 Credentials

Credentials for Blue Teams systems are stored in the Collaboration Environment in human readable plain-text format. They are not optimized for machine use or automation. The rules of engagement, in the context of LS CDX, allow the modification of credentials during the execution if there is a suspicion that the credentials might have been compromised by the Red Team during the game play [6]. Also mentioned in the 1.1.2, that the Collaboration Environment is built on top of MoinMoin Wiki engine, this means that the trace of all the changes made to the credentials are also stored in the MoinMoin Wiki *history* feature.

Keeping track of the change in the credentials is valuable feature in case of an unintentional human mistake when changing the credentials stored in the Collaboration Environment. When the UST fails to authenticate with the Blue Team systems, then they can try the previous credentials that might work. This process is time consuming but it provides a layer of redundancy.

2.2.4.2 Tasks and Their Statuses from the Red Team

During the course of the exercise, UST will be asked to perform user-interaction tasks in the Blue Team systems by the Red Team CS attacks sub-team. Those tasks and their statuses are relayed through Jabber XMPP chat client. The info that travels between the two teams, is by default not prepared in advance and is typed up dynamically by the respective team members (see figure 2.3). This method is prone to cause errors.

The data exchange is initiated by the Red Team (see figure 2.4). They will define what task, where and how will be executed. When the UST member receives the task, they must acknowledge that they have received the task. Hence exchanging the first piece of *task status* data. After the UST member has executed or failed to execute the task, they must provide additional *task status* data accordingly. This additional data is important factor for the Red Team workflow, since this entitles that the task at hand is not lost or disregarded.

2.2.4.3 Statuses of Functionality and Availability of the Virtual Services

One of the tasks of the UST was to conduct service checks to verify that the Blue Teams are providing services and that the services are still usable and available for the UST (see 1.2). The UST member does this by authenticating themselves into the Blue Team workstation and then performs the service checks. Most of the services are virtual web applications. If the service is usable and available for the UST, then in case of LS CDX the responsible Blue Team receives positive score. If the service is not usable or available for the UST then the Blue Team receives no score, but a notification and a limited amount of time to resolve the issue with named service.

The UST keeps track of the periodical service checks in an excel table [12]. Periodically the data is gathered by the sub-team leaders of the UST and after that the relevant scoring is assigned to the Blue Teams.

3. Analysis

In this chapter the author will identify the requirements for enhancing the UST workflow. After iterating over the requirements, the author will set goals to be achieved within the scope of this thesis. After identifying the goals, the design of a proof-of-concept framework will be proposed. The main research methodology used in this chapter is conducting semi-structured interviews with the leaders of related LS CDX organizing teams and from them extracting the best methodology to achieve the objectives stated in section 1.4.

3.1 Requirements

There are suggestions provided in the UST, Green Team and Red Team leaders interviews based on the previous year feedbacks of the exercise. The feedback from previous years is important to identify requirements for proposing the solution.

3.1.1 Requirements for Infrastructure

The underlying infrastructure of the exercise consists of many parts (see 2.2.1). The key element is to reuse existing infrastructure in order to avoid adding another layer of complexity to the existing and the future CDXs as noted in the semi-structured interview with Ragnar Rattas, the leader of Green Team (see appendix A.3). In the chapter 2.2.1 it was discussed, that current exercises are built up on top of VMware vSphere virtualization platform, that provides VIX API and a management network that can be used in order to develop the solution for problems stated in paragraph 1.2.

3.1.2 Requirements for the User Simulation Team

UST has in previous iterations had one day training right before the execution of the exercise to get acquainted with the infrastructure of the exercise [12]. They have been provided with the workstations. Using those workstations they have used RDP connection to enter the common management server, that in turn interacts with the virtualization environment. After successfully entering the targeted workstation, they need to furthermore interact with the services that are hosted inside the SINET. The path that the UST member has to pass in order to interact with a virtual service was previously described in the figure 2.1.

From the process it is deduced that currently there are three layers of virtualization between the UST member and the targeted workstation that they need to interact with. Getting accustomed to this kind of workflow does require some previous training of which volume, the author proposes, could be reduced with the removal of virtualization layers. This will directly address the situation (discussed in section 2.2.2) where the virtual workstations were too difficult to interact with.

After the UST team has managed to go through three layers of virtualization, they need to authenticate themselves into the workstation using credentials specified for that workstation. The credentials might be either domain credentials [1] or local user credentials. In case of domain credentials same credentials work for multiple workstations in the domain - this depends on the rules that the responsible Blue Team for said domain controller has enforced. In case of local credentials, they only by default work for the workstation that they were set. Since there are 46 workstations per Blue Team (see section 1.2.2) and each of them has a number of accounts that can be used to authenticate into the workstation, there are numerous credentials moving around during the execution. All the Blue Teams are allowed and encouraged to change the default credentials during the execution of the exercise when they suspect that the credentials are compromised by the Red Team [6]. This means that the **communication of credentials** between the UST and Blue Teams must be as efficient as possible (see 2.2.4.1 for current implementation).

The most important task of the UST is to play the role of user-interaction for the Red Team inside the Blue Teams' systems (see section 1.2.1). This means that they have to communicate very closely with the Red Team CS attacks sub-team. As soon as they receive an input from the Red Team they must respond to it.

[1] Read more about domain credentials from `https://msdn.microsoft.com/en-us/library/windows/desktop/aa380517%28v=vs.85%29.aspx#domain_credentials`

As can be observed from the figure 2.3 presented in section 2.2.3, the format of the communication is not immune to misinterpretation during the execution of the exercise. The request from the Red Team CS attacks sub-team member can be and has been in previous exercises misunderstood (see appendix A.2). Further more, the task can be complicated, time consuming and in the end the UST member might forget to give the Red Team member the much needed status report about the task at hand.

In order to address this problem, the framework should implement a *chat* functionality inside itself that exchanges data (requests for interaction, their statuses and additional messages) between Red Team and UST.

3.1.3 Requirements for User Simulation Team Workflow Improvement Framework

From previous CDXs, it has been said that there is a need to improve the efficiency and workflow of the UST, since there are several of problems with the team's workflow [10]. One of them being, that every year the UST is almost completely replaced by new members, and the resources put into training the previous year UST are lost.

From the event organizers semi-structured interviews (see appendixes A.1, A.2, A.3), the author has gathered suggestions and features that could increase the efficiency of the team through addressing the problems stated in the section 1.2:

1. The basic suggestion of being able to interact with the target workstations (helps to improve situation 2.2.2, that stated the interaction with targeted workstations was too difficult in previous years);
2. Improve Communication between Red Team and UST (helps to improve situation 2.2.3, that stated there were problems in communication between the teams):
   (a) Reduce UST member generated errors;
   (b) Reduce Red Team member generated errors;
   (c) Red Team member gets automatic feedback of the status of the task that is being handled.
3. Automation should be easy to use for the UST member (addresses the situation 2.2.2 furthermore, that stated the interaction with workstations was too difficult):

(a) Reduce the complexity of UST workflow;

(b) Clear overview of manageable systems;

(c) UST member gets visual feedback of what is happening in the system;

(d) Automatic retrieval of credentials from the Collaboration Environment for the UST member.

There also were some technical suggestions made, that are important for the process of developing the framework:

1. Automation framework should be impervious to Blue Team defense mechanisms. The defense of Blue Team should not intervene with the functioning of the framework.

2. Framework should be modular enough, so adding and removing features would not break the functionality of other parts.

3. There have to be redundancy fall-backs to other methods. If something fails, the framework should be able to fall back to a little bit more manual layer of automation (Partially addresses the problem with copy-pasting described in section 1.2.1);

4. Log every action that is being executed, so better real-time and after action feedback can be provided for Blue Teams.

5. The UST member must have full control over the automation. This is the final technical requirement, stated by Aare Reintam, the exercise director.

3.2 Development decisions

In order for framework to stay modular enough for future developments, from suggestion 2, the programming language with multi OS support should be used. At the time of writing this thesis, JavaScript has become one of the most popular programming languages [19]. In addition of being popular, JavaScript is also modular, in the sense that its OS counterpart NodeJS has a Node Package Manager (npm). This allows the developer reuse already written code and modify it to suit his needs. NodeJS also runs on every major platform and the packages coming with npm outnumber any other language package count by at least twofold [20].

A choice was made to use NodeJS [2] on the server side, MongoDB [3] in the database and native JavaScript with various libraries such as jQuery[4], Bootstrap [5] and socket.io [6] in the front-end.

Socket.io is a technology that utilities Web Sockets [7] for real time communication between the browser of the UST member and the central server. Web Sockets is a technology that is built on top of the classical Hypertext Transfer Protocol (HTTP) protocol. Biggest improvement with Web Sockets over HTTP is, that after the initial connection is established by the UST member's browser, the central server can also initiate data exchange events between itself and the browser opposed to the classical request-response type of communication where server was never able to initiate data exchange.

With this set of tools chosen for development, the whole stack can be developed in one programming language, JavaScript. This is the language that the author has had the most experience with. It is likely that this kind of choice will save time in the development process.

Taking in consideration that author also has some experience with PowerShell scripting language and the VMware has provided capabilities accessing its VIX API with PowerShell PowerCLI [8] add-on (see section 2.2.1), then PowerShell is chosen as the main scripting language to interact with Windows environment located inside VMware virtualization through before mentioned VIX API. This also inherently leads into the requirement of having Windows Server as a central management platform.

From the technical requirements briefly listed in section 3.1.3, the main requirement was that the framework should be impervious to Blue Team defense mechanisms. This means that when the Blue Teams deploy their defense mechanisms they can not interfere unintentionally with the framework. This works in favor for choosing VIX API as it works through VMware Tools service that is forbidden to damper with for the Blue Teams.

The modularity suggestion 2, emits the technical requirement, that the addition or removal

[2]Read more about NodeJS from `https://nodejs.org/en/`

[3]Read more about MongoDB from `https://www.mongodb.org/`

[4]Read more about jQuery `https://jquery.com/`

[5]Read more about Bootstrap `https://getbootstrap.com/`

[6]Read more about socket.io from `http://socket.io/`

[7]Read more about WebSockets from `https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API`

[8]Read more about PowerCLI add-on from `https://pubs.vmware.com/vsphere-60/topic/com.vmware.ICbase/PDF/vsp_powercli_63_usg.pdf`

of features should not break the existing features.

The technical requirement of logging all pieces of data that go through the framework is there to provide better feedback after the exercise to the Blue Teams.

The final technical requirement was that the UST member must have full control over the automation. This is especially important to note, as in if something fails, the team member must be able to step in so that the success of the exercise for the UST is not dependent only on the framework.

3.2.1 Methodology

The chosen methodology for development was chosen to be iterative development model[9]. Since it was known before hand, that there will be two evaluation events, where an iteration of the framework needed to be deployed, then this methodology seemed to be the best choice.

3.3 Design

The framework will have 3 main parts:

- Central server, that controls the data flow between the different parts of the framework, and it will act as the controller for the framework;
- Red Team client application to interact with the framework;
- UST client application to interact with the framework.

The design of the framework is shown in the appendix A.7.

3.3.1 Central server

The main foundation for the framework should be the central server. The server should be able to interact with the Blue Team systems and same as the whole framework, it should be modular enough to support future development (see suggestion 2). As the

[9]Read more about iterative development model `http://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm`

proof-of-concept solution will be developed for the LS CDX then the developed solution should be based on said exercise but should not be limited to it.

To interact with the Blue Team systems, the method of managing them remotely has to be chosen. Possible solutions that provide this kind of interactions in Windows workstations are VMware VIX API, PowerShell Remoting, and PsExec (see 2.2.1). When it comes down to user-interaction in the context of LS CDX, then the majority of systems and focus has historically been on the Windows workstations. The proof-of-concept framework will not disregard any other operating system, furthermore the framework will be developed in the aim of adding support additional operating systems in the future. The best way to do this is to cross-reference the gathered suggestions from the semi-structured interviews (see 3.1.3) with the capabilities of before mentioned ways of interacting with the targeted windows workstations.

Improving the speed of UST workflow does not depend on the methodology of chosen interaction method. All said types of automation are able to improve the speed of the UST workflow, since they would be replacing manual labor. Again since all of them are just methods of executing commands inside targeted workstations, then there will be need for an UST front-end client interface to communicate with this part of the framework, and the need for ease of use is transfered to the UST client application. What is relevant is the **resilience to possible Blue Team defense methods** and **support for known future developments**.

The previous years have yielded, that PowerShell and PsExec, when not configured properly by the Blue Teams, are very powerful tools in the arsenal of Red Team CS attacks sub-team. They have used it for both: gaining initial access and spreading laterally through the Blue Teams networks (see appendix A.2). This has made Blue Teams cautious about these tools and it is not uncommon anymore, to see that the PowerShell and PsExec are disabled during the execution of LS exercise (see appendix A.2). This is not considered to be invalid or against the rules, on the contrary - it is encouraged behavior. On the other hand, VMware VIX API uses VMware Tools, installed in the workstations, in order to interact with the Virtual Machines. Neither Blue Teams or the Red Team are not allowed to limit or use the VMware Tools service in any way during the game. Meaning, that the VMware VIX API interaction with the targeted workstations is indeed impervious to Blue Teams defense mechanisms. Furthermore, the PowerShell and PsExec have network requirements attached to them. They can not work remotely without networking. As previously discussed, this means that additional management NIC has to be attached to targeted workstation, in order to ensure the communications between the

central server and itself. VMware VIX API has no such requirement. All communication is done through VMware own ecosystem and there is a layer of abstraction provided, so that even the systems without any NICs attached, can be interacted with.

The other key aspect of the comparison was the **support for known future developments**. As the focus of Red Team CS attacks sub-team in the context of LS 2016 was on the Windows workstations, it was decided, that the proof-of-concept framework will initially support the windows workstations. In the future, support for Linux and MacOS must be added.

Since the PowerShell and PsExec are Windows environment tools, then to add the support for interaction with the other operating systems would mean to implement another layer of control in the framework. This layer would decide, to whom the intended interaction is addressed to, and behave accordingly. Additional tools, such as Secure Shell (SSH) [10] should be implemented to support the interaction with the Linux and MacOS systems. VMware VIX API has this layer of control already built in. Since the VMware VIX API is part of the VMware ecosystem, the VMware is aware of which type operating system it is communicating with.

In case of the interaction with the Blue Team systems, the VIX API should be used.

The workflow of central server must be kept as modular as possible (see suggestion 2), since it is the main part of the entire framework. The modularity in this case means that the central server will need to handle different types of data independently. In order to do this, first it needs to be identified what kind of data is being transfered throughout the framework:

- **Suggestion 1** demands, that the central server is capable exchanging interaction between the UST client and the targeted workstation;
- From **suggestion 2**, it can be deduced, that there is a need for communication exchange between the UST member and Red Team member;
- From **suggestion 3d**, the central server is not only an exchange between the Red Team member and UST member, but also between the Collaboration Environment and the UST member;
- **Suggestion 4** implies that the server must log all the data exchange passed through it, so a database solution is also required.

[10] Read more about SSH `http://searchsecurity.techtarget.com/definition/Secure-Shell`

Out of given list, a depiction was formed, of how the data should flow through the central server.

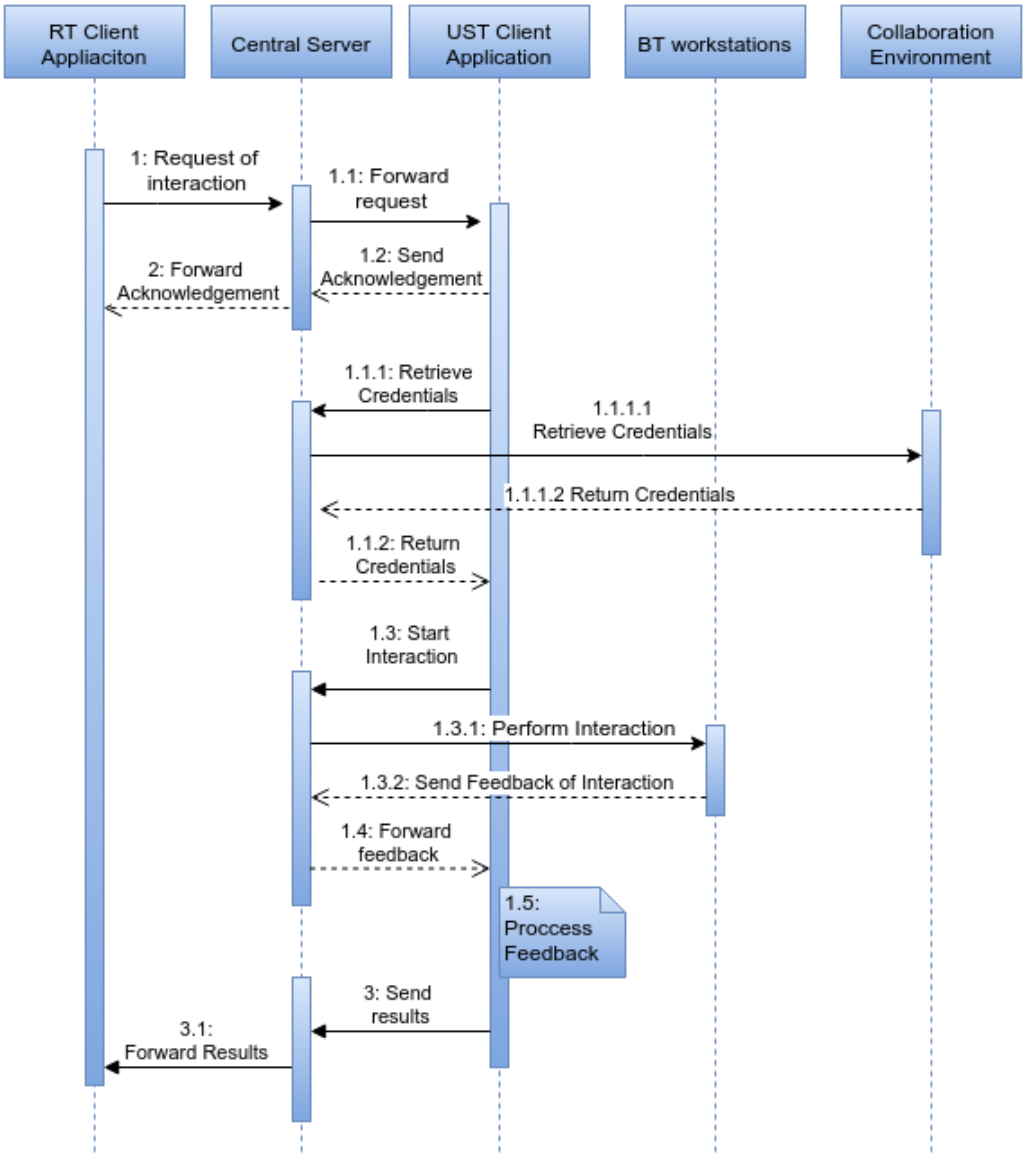

Figure 3.1: Data Flow Through Central Server

In figure 3.1, it is described, that three different types of data flow through the central server. These types are:

- Request for interaction;
- Feedback of interaction (including acknowledgment of request received);
- Credentials.

Request for interaction is a request for code being executed inside the Blue Team system. This can be triggered only from UST client (in order to satisfy suggestion 5). Then again, the Red Team needs to be able to prepare that kind of request for the UST team. This is why *1: Request of Interaction* on the figure 3.1 starts from the Red Team client application.

Feedback for interaction is the feedback transmitted back to the interaction initiator. This means that as if the Red Team member sends a request for interaction to UST member, they will receive automatic feedback, as soon as their request has been acknowledged. If the UST member requests interaction inside the Blue Team systems without the request from Red Team member (for example, in case of service checks), then no feedback is sent to the Red Team member.

Credentials are requested for all the Blue Teams by the central server periodically. They are stored inside the central server, and served out to the UST client when they require it.

3.3.2 User Simulation Team Client Application

All the request made through that client interface, that require to go through central server, are automatically forwarded for the correct target Blue Team workstation and logged in the central server.

The RDP connections for visual overview are established directly to the target workstation over management NIC of the target machine. The credentials, that are synchronized periodically with the central server, will be chosen by the UST member before the execution of the request for interaction, to ensure the request for interaction gets executed as is expected. The credentials are automatically polled from the central management server, that itself polls the credentials from collaboration environment (as is shown in the figure 3.1) , where the responsible Blue Team can change the credentials as it chooses. The author proposes that this kind of workflow should satisfy the suggestion 3.

In addition to credential synchronization and initiating the RDP connections to target workstations, the UST client provides a direct instant messaging system to the Red Team. Not only does it provide a instant messaging functionality to any Red Team member, but all instant messaging is filtered by target Blue Team number. This means, that if the UST member who is simulating for the Blue Team 78 for example, says something into the chat, only they and the members assigned for the same Blue Team on the Red Team side will be able to see the message. This eliminates information noise and leaves only relevant

information for current framework user.

Furthermore, based on the chat functionality, there was also implemented a module for the framework, that allows UST members to receive and execute tasks sent to them by Red Team member. When a Red Team member prepares a task, they will choose the attack options. The attack code is then sent through the central management server to the UST member who will then decide if and when they will execute the attack code. The UST member can change whatever they find necessary in the request for interaction, they have full control (see suggestion 5). If they find, that everything is fine, they will first initialize a RDP connection to the target workstation, to receive visual overview of the system. If the RDP connection has been made, they will send a request for interaction forward to central server and meanwhile the UST member client will send automatic feedback to the Red Team member, that the request is being handled.

3.3.3 Red Team Client Application

Since the Red Team client needs to be accessible to all the Red Team CS attacks sub-team members and opposing to UST client, where interaction with the local machine was necessary, the Red Team client application will be an web service hosted in the Central Server.

Red Team client application is a web interface. It exist so that the framework has input from the Red Team member. Red Team member will be the one who gives input to the framework. The Red Team member decides what has to be executed, where and how (see figure 4.1). UST, the main focus group of this thesis, is the team that processes and controls the request. To do this efficiently, they will need to have parameters, such as what, how and where that were mentioned before. In case of CDX "what" deducts to basic types of attacks that the Red Team members use. For proof-of-concept framework, two attack options were developed:

- PowerShell cmdlet [11];
- Malicious webpage/hosted file.

The Red Team member can choose a attack type out of those two, next they will need to choose a target and some other options such as targets of where to execute the request for

[11]Read more about PowerShell cmdlet from `https://msdn.microsoft.com/en-us/library/ms714395%28v=vs.85%29.aspx`

interaction or which browser to use. Keeping in mind the suggestion 2, the possibility for additional attack types was left open.

When the request for interaction has prepared, the data will be sent through the central server to the UST client. As soon as the UST member starts interacting with the request, the Red Team receives automatic feedback through the framework, that their request is being handled.

In addition to preparing a interaction requests and automatic feedback for them, there was additional “chat” functionality added, to give additional feedback when needed and to provide additional fall-backs, in case the automatic interaction with the Blue Team fails, and a task has to be executed manually.

4. Implementation of the Framework

In this chapter the author presents the implementation of the framework and its nodes: central server, red team client application and user simulation team client application.

4.1 Central Server

As discussed in section 3.3.1, the method for interaction with the workstations was chosen to be VMware VIX API. From the section 3.2, it was decided that the main scripting language to interact with the VIX API PowerShell through PowerCLI add-on.

The code example 4.1 demonstrates how the PoweCLI Invoke-VMScript [17] cmdlet is prepared before the execution:

```
1 Invoke-VMScript
      -vm '${req.params.target}'
      -GuestUser '${req.params.username}'
      -GuestPassword '${req.params.password}'
      -ScriptText '${schtasks_template}'
      -ScriptType '${req.params.scripttype}';
```

Code example 4.1: Invoke-VMScript Cmdlet Executing Code Inside the Workstation as an User Using VIX API

The framework uses five parameters out of possible fourteen parameters defined in the documentation. These parameters are:

- **vm**, defines the Virtual Machine, where the interaction request will be executed;
- **GuestUser**, defines the username, as whom the command will be executed inside before mentioned Virtual Machine;
- **GuestPassword**, defines the password for the username;

- **ScriptText**, defines the code that will be executed inside the targeted Virtual Machine;
- **ScriptType**, defines the type of the script that will be executed inside the targeted Virtual Machine. Possible supported types are BAT, BASH, and PowerShell. This supports the suggestion 2, as in for future developments there is a support for Linux and MacOS OSs.

There were two main difficulties when implementing the VIX API based solution. Firstly, the commands are executed inside the targeted workstations under the VMware Tools service. This meant that any code executed inside that service runs in the background and is not interacting with the User Interface of the Windows directly. To bypass this problem, one option would be to run the command with local administrator credentials. This level of integrity is high enough to bypass this restriction. Using administrator credentials has to be avoided, since in the context of CDX exercise, the simulated users have rarely administrative rights.

Solution was to introduce a wrapper for the commands in the form of **schtasks** [1].

```
1 var schtasks_template = `
2   schtasks /create /sc once /tn ${randomHash} /tr "${req.params.cmd}"
      /st 13:00 /F
3   & schtasks /run /tn ${randomHash}
4   & ping 127.0.0.1 -n 6 > nul
5   & schtasks /create /sc once /tn ${randomHash} /tr " - FailSafe - "
      /st 13:00 /F
6   & schtasks /delete /tn ${randomHash} /F
7 `;
```

Code example 4.2: Using Schtasks to Bypass Windows Limitations

Schtasks is a windows built in command line feature of managing a Scheduled Tasks [2]. The command example 4.2, demonstrates how the framework uses scheduled tasks in its advantage:

1. A JavaScript variable called schtasks_template is created that will be injected into the ScriptText variable described in code example 4.1;

[1] Read more about Schtasks feature in Windows from `https://msdn.microsoft.com/en-us/library/windows/desktop/bb736357%28v=vs.85%29.aspx`

[2] Read in more detail about Scheduling a Task from `http://windows.microsoft.com/en-au/windows/schedule-task`

2. Line 2 in the code example 4.2 is the actual Scheduled Task wrapper task that is generated around the code that came from the UST client application - $req.params.cmd;

3. Line 3 in the code example 4.2 executes the task;

4. Line 4 in the code example 4.2 is artificial delay that lasts around 5 seconds;

5. Line 5 in the code example 4.2 is overwriting the Scheduled Task that was created before;

6. Line 6 in the code example 4.2 is deleting the Scheduled Task that was created before.

It was required to introduce the artificial delay, since the target group of windows workstations contained different versions of windows from Windows XP to Windows 10, with the exception of Windows Vista. Although of Scheduled Tasks and its command line interface schtasks, are native part of Windows, there were some differences in the way they handle scheduled tasks. For example, some of scheduled tasks executed very quickly (less than one second), however in some cases, the execution took over three seconds and it was possible that the scheduled task was attempted to be removed before it was actually executed (in case of Windows XP).

The overwriting of previously defined Scheduled Task was needed, because in case of Windows 7 and **only in windows 7**, the schtasks had an interesting way of action: a regular user is allowed to create, run and modify tasks but not delete them if the name of the task was shorter than nine characters. The correct fix for this problem was deployed - all the names of the tasks are random strings with the length longer than ten characters. Yet, this overwriting of the Scheduled Task was left inside, to counter this kind of problems that might happen in the future additions of the development.

4.2 Red Team Client Application

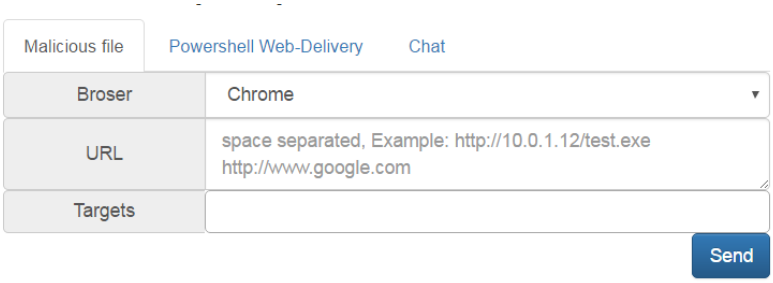

Figure 4.1: Red Team Client Application Malicious File Tab

The interface for Red Team Client Application for sending over malicious file to the UST member is shown in the figure 4.1. There are two tabs for preparing the user-interaction for Red Team attacks, as described in 3.3.3 and additional tab called "chat" for feedback and additional communication with the UST member to satisfy the suggestion 2.

Figure 4.2: Red Team Client Application PowerShell Web-Delivery Tab

In the figure 4.2 it is shown the second tab of the Red Team Client Application. There are only two inputs in that tab, one for the PowerShell malicious code, that the Red Team member wishes to execute and another one is a input for the targets that the Red Team can attack. The targets are synchronized from the Central Server with each page refresh.

The third and final tab, Chat, in Red Team Client Application showed in figure 4.3, serves two purposes. The main purpose being to improve the communication between the teams.

From there, a Red Team member can communicate with the responsible UST member directly. Furthermore, all the tasks, that are sent through the previous two tabs, are converted into human readable format and are given background color representing their statuses. “Red” representing failure, “Yellow” is in progress and “Green” is completed successfully. This way of presenting the data should, in authors opinion, give a good situational overview of the current situation to the Red Team member.

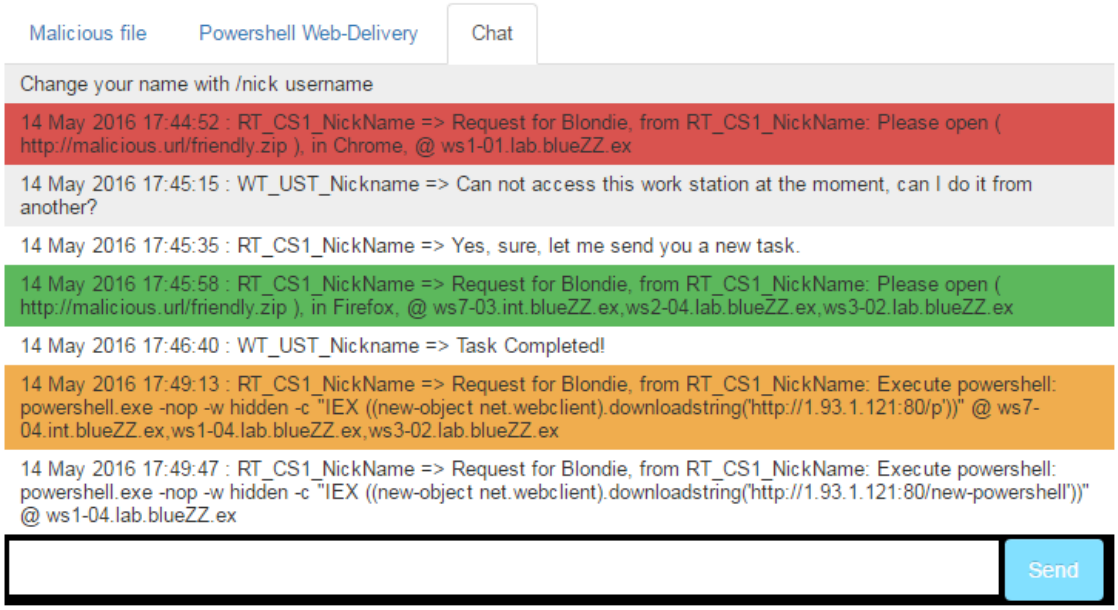

Figure 4.3: Red Team Client Application Chat Tab

4.3 User Simulation Team Client Application

The UST client is an application that serves an web-interface for localhost to seamlessly communicate with the central server. The application requires to run locally on UST member workstation in order to communicate directly with the local RDP client installed there. To make interaction between different Blue Teams clear, the application serves different interface depending on the exact Uniform Resource Identifier (URI).

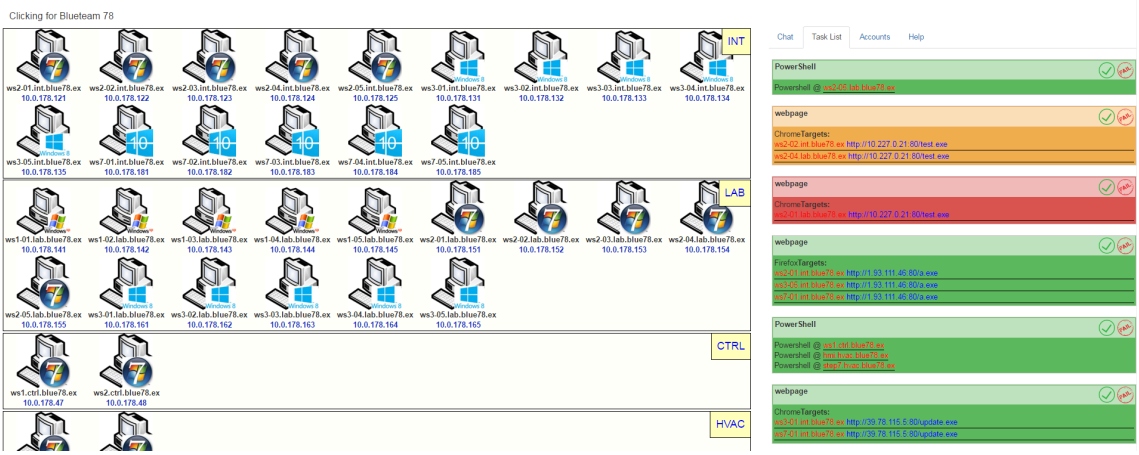

Figure 4.4: User Simulation Team Client Application Dashboard

The figure 4.4 depicts the dashboard of the User Simulation Team Client Application. On the left hand side, there are all the windows workstations that the particular UST member has to interact with, satisfying the suggestion 3b. On the right hand side there were four different tabs developed:

- Chat;
- Task List;
- Accounts;
- Help.

Chat feature is to improve the communication between the UST member and the Red Team member, as suggested in suggestion 2.

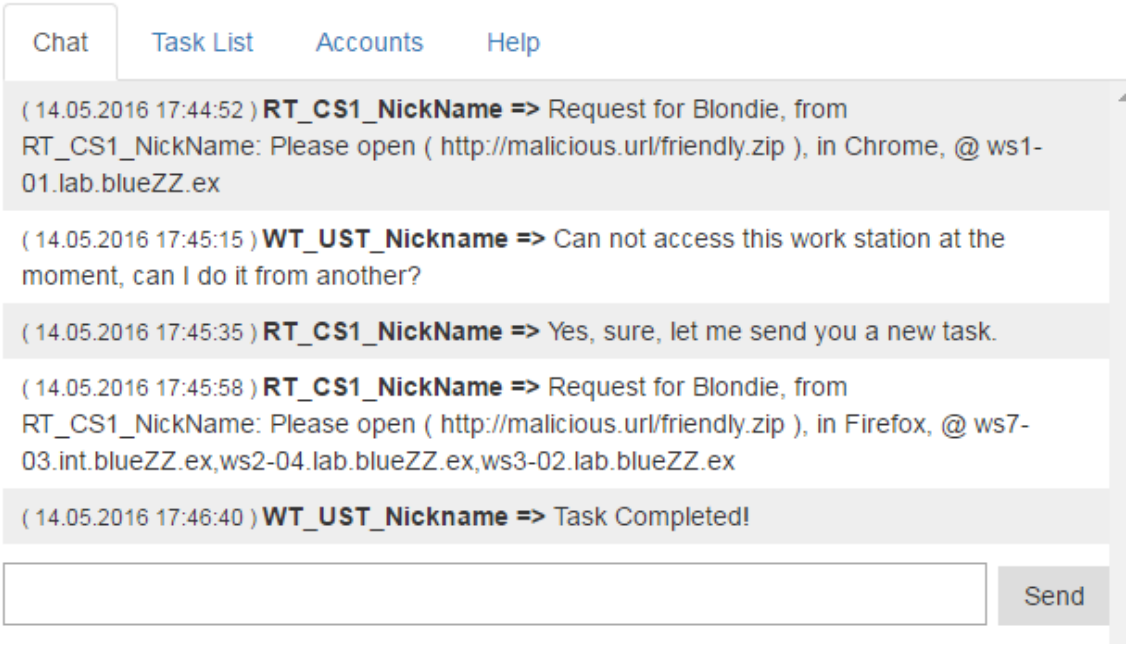

Figure 4.5: User Simulation Team Client Application Chat Tab

The **Task List** is there to handle and control the requests for interactions from the Red Team member. Every request for interaction from the Red Team comes through the central server and is displayed for the UST member here. They are color coded into four different variations for ease of use: Yellow, Green, Red and White. As the task arrives, it has "white" color attached to it, marking it as a new. As soon as the UST member starts to interact with it, it turns into "yellow". After the task has been executed, the UST member has to evaluate if the task was a success or failure by using the buttons on the right hand side of the said task. Success will be marked with the "green" and failure with the "red" color. Each change in the state of the task will be instantly sent back to the Red Team member to satisfy suggestion 2c.

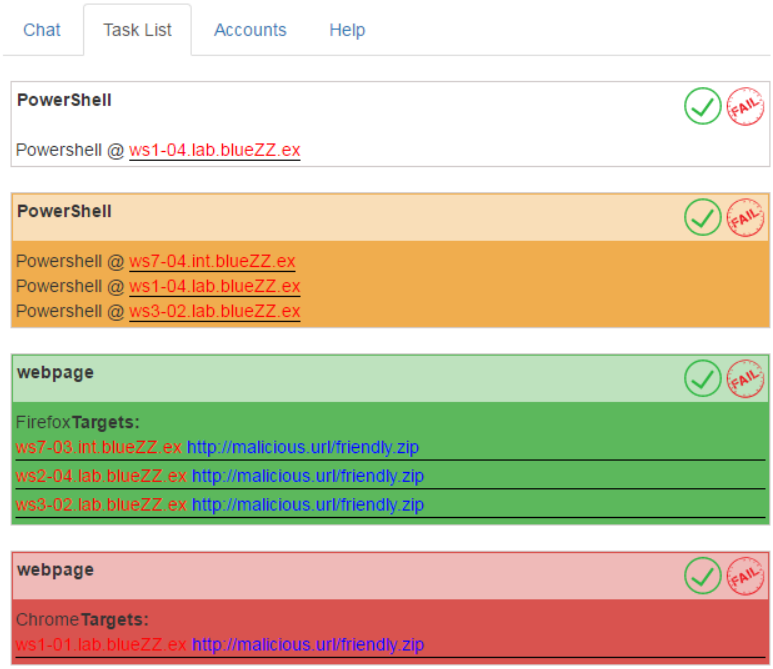

Figure 4.6: User Simulation Team Client Application Tasks Tab

Both, the chat and the task exchange functionality was built on top of the socket.io engine as was described in development decisions (see section 3.2).

The **Accounts** tab is there to present a clear overview for the synchronized passwords. It is a list of all the passwords retrieved through the central server from the Collaboration Environment. Since the User Simulation Team client Application uses the passwords semi-automatically already, then the reason for this tab is just to furthermore satisfy the suggestion 3. If something fails, such as using the credentials automatically to perform tasks, the UST member has a possibility to conveniently copy-paste the credentials for manually performing the task retaining some improvement in efficiency of the UST member workflow.

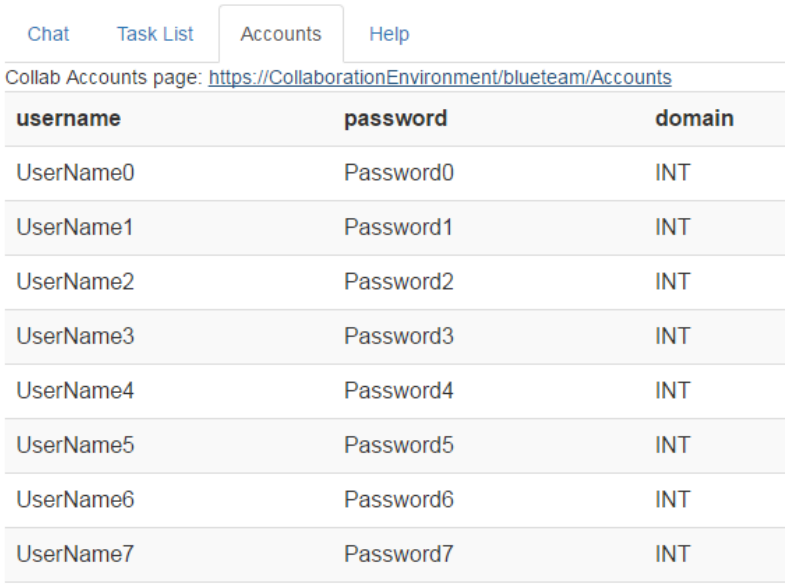

Figure 4.7: User Simulation Team Client Application Accounts Tab

The **Help** tab is there to provide additional helpful features that are not directly part of the framework, such as hyper-links to credentials page in the Collaboration Environment of the selected Blue Team if the work has to fall back to manual.

To satisfy suggestions 2a, 3c and 3, there was another feature added to the dashboard. It was decided to use RDP as a service for visual feedback to the UST member. To trigger a RDP connection from inside the UST member workstation to targeted Blue Team workstation, two requirements had to be met. Firstly, the client application had to be run locally inside the UST workstation, so that it is possible to interact with the system. This is a requirement that does not exist for a Red Team client application.

The final implementation of the User Simulation Team client application triggering RDP connection is depicted in the figure 4.8. In more detail, the MSTSC [3] command line interface was used to trigger RDP connection to the targeted workstation over MGMT network IPv4 address. To achieve this, credentials required for the RDP connection were stored inside the Windows Credential Manager [4]. The credential storage process was synchronous, as in, when the RDP connection was initiated from the Web Interface, the local server first stored the credentials in the local machines Windows Credential Manager. Next, the RDP command was triggered in the format shown in Code example 4.3, where **$ip** is the MGMT network IPv4 Address for targeted workstation.

[3]Read more about MSTSC from `http://ss64.com/nt/mstsc.html`

[4]Read more about RDP and CredMan from `https://blogs.msdn.microsoft.com/winsdk/2015/01/02/rdp-and-credman/`

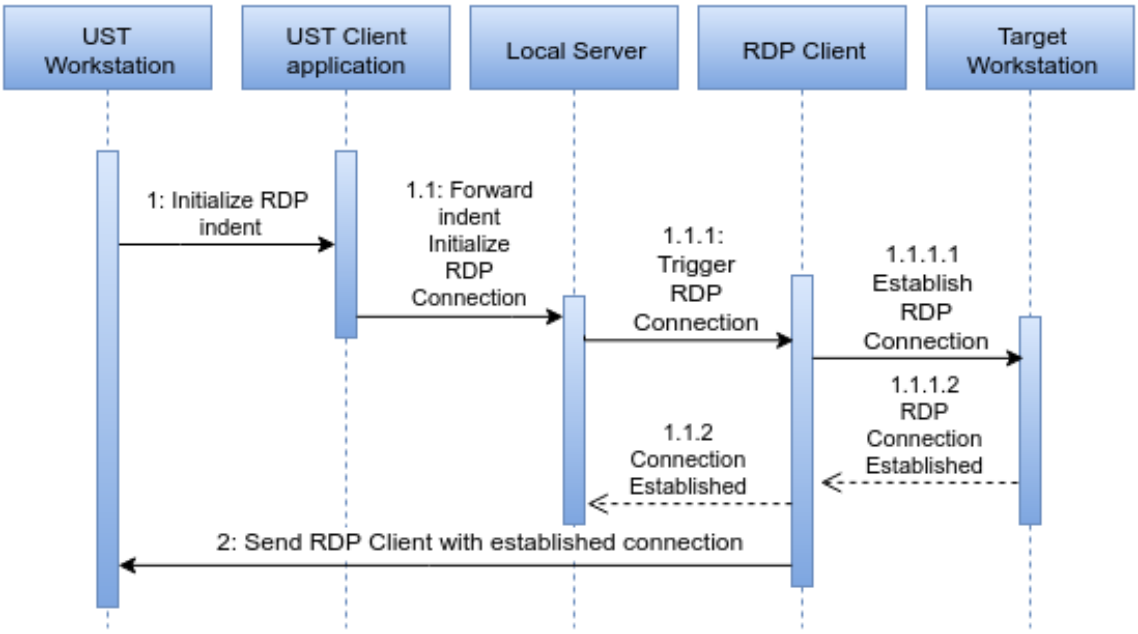

Figure 4.8: RDP Connection Solution Inside User Simulation Team Client Application

When the MSTSC process responsible for the RDP connection was closed, then the stored credentials were deleted from the Windows Credential Manager.

```
1  mstsc /v:$ip
```

Code example 4.3: Triggering RDP Using MSTSC Command Line Interface

5. Evaluation of the solution

5.1 The Test Run

The LS 2016 Test Run was held on 10th March, 2016. LS Test Run 2016 was a limited game-play between fully manned Red Team and five Blue Teams. At the Test Run there were 5 members of UST present of total 23. The first iteration of the framework was deployed for the LS 2016 Test Run event.

5.1.1 Preparation Phase of the Exercise

The preparation phase is considered in this context to be the since the beginning of the development of this framework until execution of the Test Run. All the critical features were developed and implemented before the Test Run event, and were ready to be fully tested. There were left room for improvements, as the color code enabled feedback system between the UST members and the Red Team members was deployed later.

5.1.2 Execution Phase of the Exercise

The first day began with the training of the framework for UST members. The training session took 15 minutes before the event. Training for the Red Team members took approximately five minutes during the event. Both teams were warned and instructed that if the framework should fail, the fall-back is to do everything manually, as were done in the previous years (see paragraph 2.2). For Red Team this meant that instead of the framework they would use Jabber XMMP chat client to communicate with the UST. For UST failure of the framework would have meant to manually open RDP connection to one central windows server, to connect to VMware vSphere, to find the correct target machine (see figure 2.2) and open the said machine console, so they could execute the task given to them by Red Team member. Failure of the framework did not happen.

Surprisingly the underlying infrastructure - VMware vSphere - failed. At the moment of writing this thesis, the reasons for this are still unknown [13]. What has been made certain of, is that the failure was not triggered from the developed framework. After the failure, it was impossible for the UST member to connect to the vSphere, therefore the UST lost capability of using the method *doing it the old way* to interact with the systems.

The failure event showed that the modular fall-back design described in the paragraph 3 was a good choice. The VMware was unaccessible, so the framework could not provide code execution through VMware VIX API. This event did not in any way interfere with the framework's capabilities to establish a RDP connection to target workstation nor synchronizing the passwords with Collaboration Environment, so the only inconvenience that the UST experienced was the absence of automation in fulfilling of the *request for interaction* that they had to perform manually. Since the framework provided them with all the necessary information needed to perform the task semi-manually, the inconvenience was rated minimal in the UST leader, Elvis Paat', feedback to the author.

Both teams, the Red Team and UST, identified some features that could be included and improved in the framework.

5.1.3 User Simulation Team Feedback

The UST members gave their feedback from the exercise that was as follows [12]:

1. Improve a service functionality checking with implementation of automatic authentication to services;
2. Implement a availability check for workstations;
3. Improve a system for automatic feedback to Red Team;
4. Improve the icons of the workstations. It was difficult to identify between Windows 10 and Windows 8.1.

New icons for user interface, automatic feedback improvements for Red Team members about the actions of a UST members and availability checking for RDP port in the target machine was implemented in the time of the live exercise. The service functionality checking improvement with automatic authentication to different services was not, since this would have meant major additional development to support it on the workstations or

infrastructure side. This was left to future development, furthermore the modularity of this framework (see suggestion 2) supports the extension in that direction.

The UST members also reported various minor bugs that were fixed on the go. Unfortunately, the author had failed to account for an updating system for the User Simulation Team client application and each iteration had to be deployed manually. It was decided to include automatic updating by the time of live event.

5.1.4 Red Team Feedback

The Red Team members were very generous of giving positive feedback. The main tone of the feedback was, that the interaction with the framework was easy and clear to understand. They did identify some minor bugs that were present during the Test Run, such as "Request for interaction" failed to be generated, because it started with unaccounted whitespace.

Such minor bugs inside the framework were fixed on the go, and improvements were deployed instantly during the Test Run.

5.2 The Live Event

The live event was held in the week from 18th of April 2016 to 21st of April in 2016. The week started off with preparations. The execution started in the morning of 20th. In the morning of 18th, the author decided to implement additional functionality for UST, regarding automatically opening a set of web pages to improve the task of manual functionality and availability testing for the web services.

5.2.1 Preparation Phase of the Exercise

The preparation phase in the context of this thesis' scope started imitatively after the Test Run event. All the feature requests from the UST members were addressed.

From the UST feedback (see 5.1.3), it was clear, that some of the features of the framework needed to be improved. As discussed before, it was decided to leave service functionality checking improvement with implementation of automatic authentication to services to the future, as this would require an extensive wrapper development around the request of interaction. It is possible, and author proposes that one of the ways to achieve

it would be using the Selenium [1], as it provides for browser automation. It was decided to postpone this additional feature as a future development, since the Green Team did the first functionality tests with Selenium for the year 2016 exercise [13] and it might be possible to integrate their framework with the framework proposed in this thesis.

Secondly, there was a request for initial indicator to check if the targeted workstation is serving out a RDP service over the MGMT network. This feature was implemented by the time of live event as an extra module to the framework using the flowing method:

1. Web interface sends a request to the local web server to check RDP connectivity to targeted workstation;
2. Web server tries to open a Transmission Control Protocol (TCP) connection on port 3389 (Default port for RDP) [21] ;
3. The response from the network test, is sent back to the Web interface and the central server for logging to furthermore satisfy suggestion 4;
4. The response is displayed in the right-upper corner of the related machine icon, as green or red for available unavailable respectively;
5. Process is repeated for every 40 seconds for all the workstations of the Blue Team the UST member is viewing.

Thirdly, improvement of the feedback system was made to include the color-coded status reporting between the UST member and the Red Team member discussed previously in section 4.2.

Fourthly, quality of life improvement was made in the format of making it furthermore clear to differentiate between the Windows 10 and 8.1 icons (See suggestion 3).

The final major improvement was to implement an automatic update system supported by the central server, where the UST member would only need to run one BAT file as administrator to update the User Simulation Team client application. The updating service was set up as follows:

- UST member runs updater.bat file as an administrator;
- Inside the updater file, there is a short PowerShell script(see code example 5.1), that downloads and executes a PowerShell code from central management server;

[1] Read more about Selenium from `http://www.seleniumhq.org/`

- The PowerShell code, downloaded from central server (see an example in appendix A.4), that is executed in the UST member workstation, first stops all processes of node and mstsc to avoid conflict during the installation. Next it downloads a new version of client application, extracts it, removes the zip container and launches the application again.

```
1 powershell.exe -nop -c "IEX ((new-object net.webclient)
    .downloadstring('http://redrobin.ex/updater'))"
```

Code example 5.1: Updater.bat Script for Updating the User Simulation Team Client Application

This update process proved to be very useful. The updating process was used to deploy some last minute fixes and modifications to the User Simulation Team client application.

5.2.2 Execution Phase of the Exercise

In the morning of 19th of April, when the training for UST members was scheduled, the framework performed poor. Debugging showed that the added functionality of the framework, which tested the availability of RDP connection in the target machines, was causing a heavy load on the central management server logging. The main factor that revealed the issue was that it was not allowed to practice on any other team than the designated *testing team*. Since the entire UST was practicing the framework on the same team, a use case was identified, that was not considered in the design process. All of the UST Client Applications were modifying the same data model in the central management server and doing it quickly. This caused a considerable amount of networking delay inside the framework. Each request, that should have been completed instantly, took in reality up to 2 minutes to process by the central server. This rendered the whole framework unusable. Thanks to the modular nature of the framework, removing this non-critical, quality of life feature and redeploying the UST Client Application through the added updater feature, took all-together less than half an hour. Most of the time was spent on identifying the problem. After the removal of the non-critical feature, the framework started performing as expected.

The following days framework did not fail at any point and it was able to continuously run throughout the execution of the exercise. No additional bugs were reported. Some features were requested for improvement. Some of them were left for future development, some of them were implemented immediately. For example, the Red Team member client

application, as mentioned in the 3.3.3 paragraph, uses web interface for sending requests for different UST members identified by the corresponding Blue Team number that they are responsible for. One Red Team member was attacking more than one Blue Team, usually four, this meant that the framework's web interface was open in multiple browser tabs. The title of the web page and therefore the tab title did not reflect the corresponding Blue Team number, so finding the correct tab took some time. This was a easy thing to upgrade and it was implemented immediately after the discovery.

Other, more difficult requests, such as Linux and OSX support was left for future development and research.

5.2.3 Aftermath and Feedback of the Exercise

All together there were 1813 messages exchanged between the UST and Red Team members. Out of those 499 were requests for user interaction. 157 were requests for interaction using PowerShell Web-Delivery, and 342 were requests to open a malicious web-page in the browser (see figure 5.1).

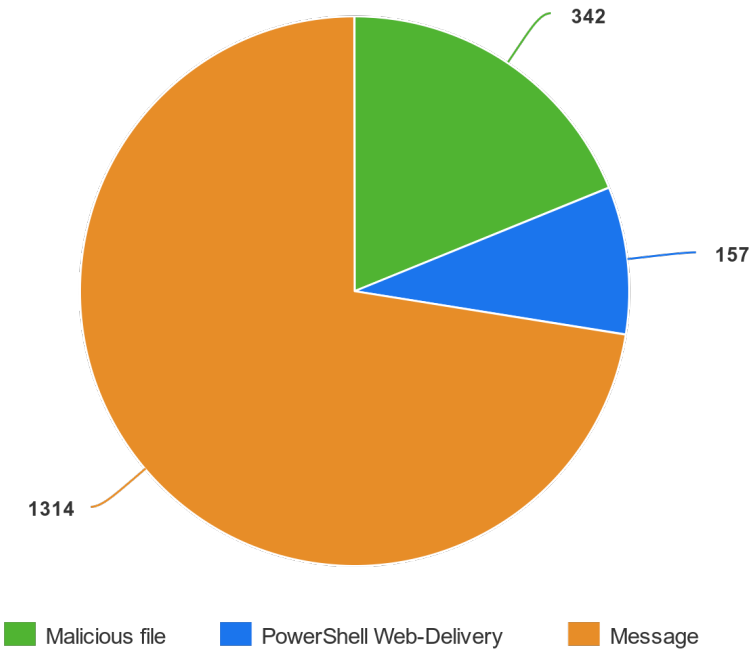

Figure 5.1: Communication Between Red Team and User Simulation Team during Locked Shields 2016

From all of the web page requests, 73 requests were made to open file in chrome web

browser, IE was requested 83 times, Firefox 64 times and in 122 cases “default browser” was left as an preference from the Red Team member side (see figure 5.2).

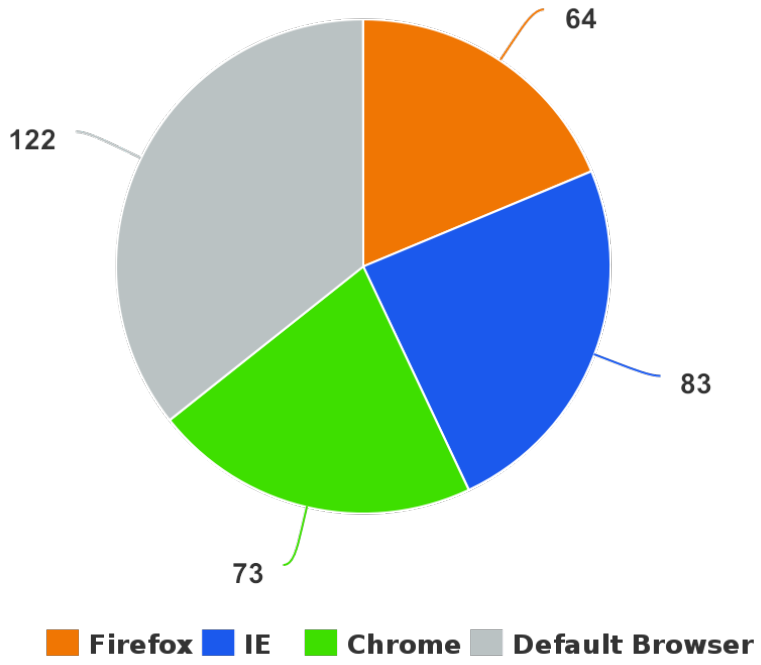

Figure 5.2: Red Team Browser Preference for Requests of Interactions

Automatic feedback was exchanged 3166 times. Out of those 296 were marked as a “failure”, 2458 as “in progress” and 412 as “done”.

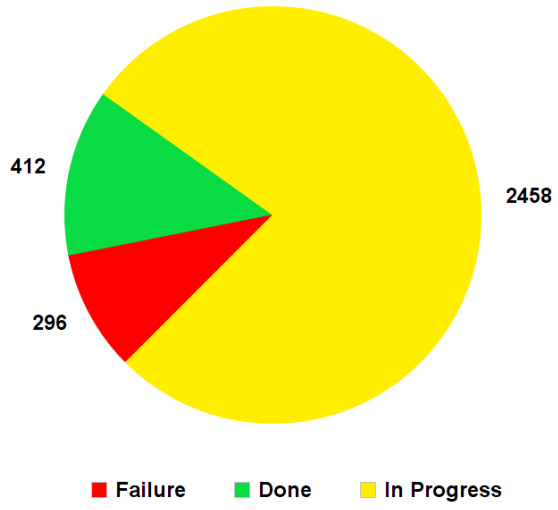

Figure 5.3: Automatic Feedback Reports Sent to Red Team

The automatic feedback was sent each time, a UST member was interacting with the task.

All together the UST tried to execute 973 commands inside Blue Teams systems. The service functionality testing was performed 423 times.

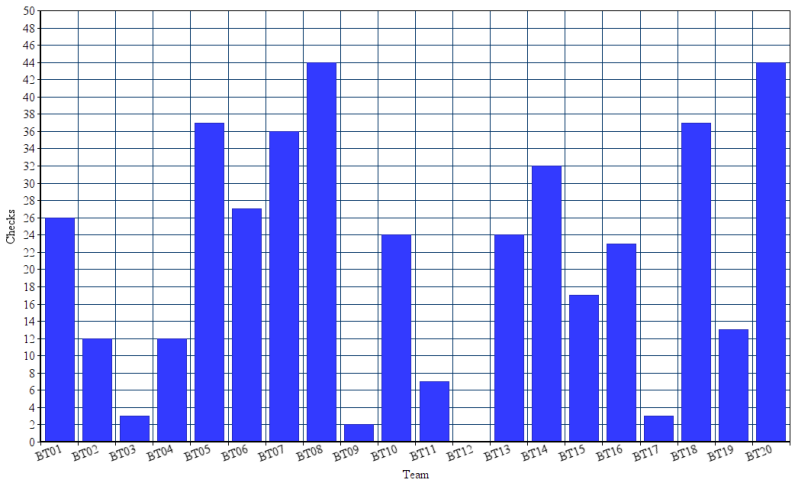

Figure 5.4: Service Checks Performed Through Framework

In the figure 5.4, it is shown how the UST member service checks divided over the Blue Teams in LS 2016. This data was recorded in order to satisfy suggestion 4. From this dataset it is clear, that the framework was popular as on average there was a service check execution in approximately every 2 minutes in the course of two days.

From figure 5.5 it is demonstrated how the density of interactions was divided over the course of two days. As foreseen by the Mehis Hakkaja (see appendix A.2), the heavier work-load for the UST was in the first half of the first day of the exercise. By the end of the first day, the workload did somewhat stabilize. In the beginning of the second day, when probably the Blue Teams had managed to rise the defenses, the user-interaction was needed again to re-compromise the systems. Both of figures showed, that the problem of difficulties with interaction of the workstations (see section 2.2.2) was resolved.

The total amount of messages exchanged between the teams is showed in the figure 5.6.

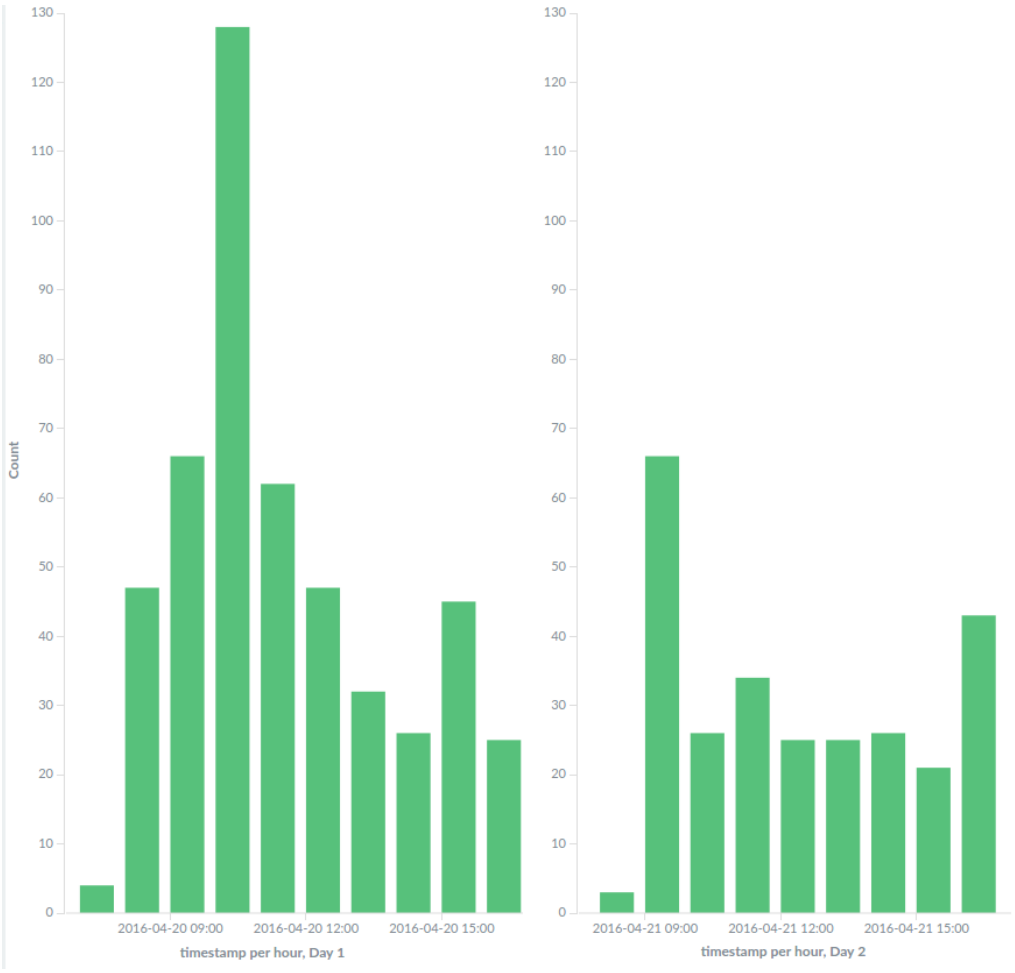

Figure 5.5: All User Interactions Performed over Two Days

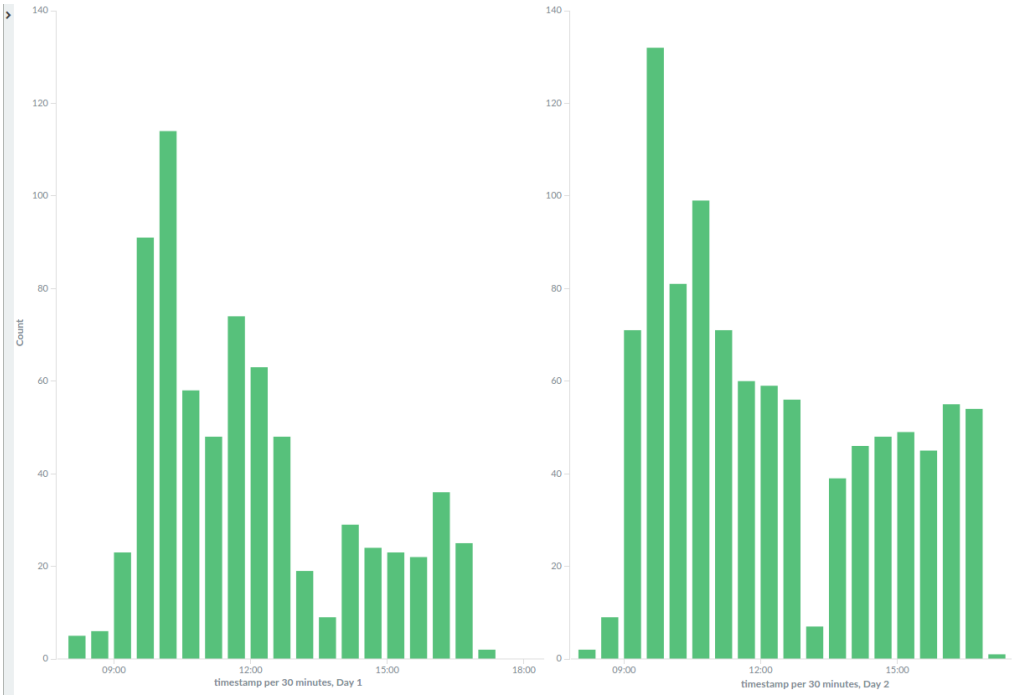

Figure 5.6: All Messages Exchanged Between Red Team and User Simulation Team

Correlating the activity from figure 5.6 and 5.5 shows that the peaks of both activities happened roughly close to each other. It can be deduced, that in addition to solving the interaction problem, the framework was also successful in solving the communication problem (see section 2.2.3) between the teams.

To evaluate the result of solving copy-pasting problem (see 1.2.1), a personal feedback was asked from UST leader, Elvis Paat. He stated that none of the team members reported any problems with the copy-pasting [12].

5.3 Lessons learned

Initial base-mark of the statistics was generated from the logs of the framework. The statistics proved to be a good evaluation for the framework. The framework did improve the UST member workflow, as it was used by almost all of the team members. Only in one case out of twenty (see figure 5.4), the framework was chosen to be not used.

The deployment process was the biggest improvement from the development point of view, as this simplified the redeployment process significantly.

From the Red Team CS and UST leader feedbacks, added in appendixes A.5 A.6, it was clear that the members of UST and Red Team CS sub-team were very satisfied with the framework. Both said that the communication was improved. The initial acknowledgment of received request took on average fifteen minutes from the second that the Red Team member sent it in previous exercises (see appendix A.5). Now it was almost instant and there was reduced amount of communication errors. Furthermore, the option in Red Team Client application, that provided the possibilities for choosing targets from, had reduced Red Team generated errors. For the UST the feature that synchronized passwords with the Collaboration Environment reduced the UST generated errors.

Both team leaders had some additional suggestions in mind for the future development. For the Red Team, support for Linux and MacOS operating systems would be beneficial, as it would increase the efficiency of Red Team CS attacks as it did for Windows this year.

For the UST a *UST Management module* was proposed. Since the framework is producing a valuable dataset about the state of the UST members, then analyzing this data and presenting it out to the team leaders in real time would furthermore increase the efficiency of the UST. Another state for the *request of interaction* was proposed - *delayed*. This state would mean that the UST member has received the task, but is unable to perform it at this

point of time. This would notify the Red Team member imitatively of the status and again furthermore improve the communication between those two teams.

5.4 Conclusion of evaluation

The framework managed to work successfully through both iterations of the deployment. The framework was used by the UST and the Red Team members as can be seen from the statistics provided in the section 5.2.3. Chosen evaluation method was successful for evaluating the performance of the framework to a degree, as conclusions can be drawn.

Framework improved the workflow of the UST and the Red Team members as it was used to exchange 1813 messages, 3166 feedback updates and 973 times to execute code inside Blue Teams systems. This statistics also demonstrates that in most cases the framework, managed to be impervious against Blue Teams defense methods (see suggestion 1).

The framework was modular enough in nature (see suggestion 2), so when the quality of life feature, RDP availability checking in targeted workstations, failed, the removal of the feature that caused the problem was quick and the framework continued to work afterwards (see section 5.2.2).

Due to suggestion 4, there now is measurable data about the interaction between Red Team, UST and Blue Team workstations, that can be used in the future research.

Feedback from the UST and Red Team CS leaders indicated several aspects of improvements, such as the reduction of human-generated-errors on both sides. Reducing the time between the clicks and therefore increasing the efficiency of UST and improving the communication between UST and Red Team.

6. Future Research

The evaluation of the framework showed, that the needed improvement in efficiency of the UST was achieved. Furthermore, since the framework is also gathering data of the actions from the UST, then it provides a possibilities for future improvements.

The data the framework gathers is valuable for future developments. One proposed development (see appendix A.6) is implementing an *UST management module*, where the team leaders get a current overview of the work the UST is doing. This can furthermore improve the efficiency of the team. The data for named module is present and the modularity of the framework fully supports implementing this kind of module for the future CDX.

In addition to that, some improvements can be made in the existing framework, as implementing the RDP availability checks inside the framework for the Blue Team workstation's management NIC and introducing the additional, *delayed*, state for the feedbacks that the framework exchanges between the UST and the Red Team to improve the communication between those teams even more.

The final suggestion that was also discussed in chapter 3.3, is the support for MacOS and Linux workstations. Due to development decisions made, to use VMware VIX API for interaction between the UST member and the Blue Team systems, this goal for future development is very realistic.

The possibility of adding those improvements will be analyzed in more depth and viable upgrades will be deployed for the next iteration of Locked Shields exercise in the year 2017.

7. Conclusions

The thesis focuses on user simulation improvements through technical solutions in the context of Cyber Defense Exercise. In more depth, it explores various methods of how the efficiency of User Simulation Team can be improved through automation.

The selected methodology combines analysis of the workflow trusted to the User Simulation Team with practical design and implementation of proof-of-concept framework for the Locked Shields 2016 Cyber Defense Exercise. The validation method is the analysis of after exercise observations from the before mentioned Cyber Defense Exercise.

The problems that this thesis tackles is the inefficiency in User Simulation Team workflow caused by the overhead of manual work that they have to do. To overcome this problem, the author first identified the problematic aspects of the User Simulation Team workflow. They turned out to be mis-communication between the User Simulation Team and the Red Team members and difficulties when interacting with the infrastructure for the User Simulation Team members.

The bottlenecks for User Simulation Team were identified and solutions to those problems were proposed. The main solution was to develop a technical solution in the form of a framework.

To make the best possible design decisions, a research was done in the form of conducting series of semi-structured interviews with the key Locked Shields exercise organizers and team leaders. Out of their suggestions for the framework a design was proposed.

The design was chosen to be modular in nature, so that features of the framework can easily be added and removed.

The framework was implemented in three major parts, interfaces for the Red and the User Simulation Team and central server to manage the data-flow between them.

The framework's main features were to enable enhanced communication between the User Simulation Team and the Red Team compared to already existing one. Second feature was to enable the User Simulation Team to send interaction commands directly to the Blue Teams workstations through the central server, for service checking. Third feature, that built upon the previous two, was to enable fluent flow for *request for interaction* from Red Team, through User Simulation Team, who acted as a controller for the request and through the central server to the targeted Blue Team's workstation. Automatic feedback for given process was sent back to User Simulation Team and then-after Red Team's client interface.

As was concluded from the feedbacks with the Red Team Client Side attacks sub-team leader Heliand Dema (see appendix A.5) and the User Simulation Team leader Elvis Paat (see appendix A.6), in the format of semi-structured interviews, the communication between the two teams was better than ever. The User Simulation Team was efficient in their work and managed to achieve all their goals.

The framework furthermore implemented logging that provided measurable data to set the baseline for next exercises about the workflow of User Simulation Team and opened up possibilities for future developments and improvements. In the end, framework was designed, implemented and evaluated.

This thesis showed that the it is possible to use technical solutions to improve the internal workflow of a User Simulation Team in the context of a Cyber Defense Exercise.

References

[1] Statistics and facts on internet usage worldwide, 2016. URL http://www.internetlivestats.com/internet-users/. Accessed 2016-04-03.

[2] APAC Cybersecurity Maturity Dashboard 2015, 2015. URL http://cybersecurity.bsa.org/2015/apac/. Accessed 2016-04-03.

[3] Panagiotis Trimintzios. National and International Cyber Security Exercises: Survey, Analysis & Recommendations , Oct 2012. URL https://www.enisa.europa.eu/publications/exercise-survey2012. Accessed 2016-04-27.

[4] Adrien Ogee, Razvan Gavrila, Panagiotis Trimintzios, Vangelis Stavropoulos, Alexandros Zacharis. Latest Report on National and International Cyber Security Exercises, Dec 2015. URL https://www.enisa.europa.eu/publications/latest-report-on-national-and-international-cyber-security-exercises. Accessed 2016-04-27.

[5] Locked Shields 2016, 2016. URL https://ccdcoe.org/locked-shields-2016.html. Accessed 2016-04-15.

[6] Mehis Hakkaja, CEO of Clarified Security and Red Team Leader in Locked Shields 2016, personal communication, April 2016.

[7] NATO CCD COE. Cyber Defence Exercise Locked Shields 2013, After Action Report, 2013. URL https://ccdcoe.org/publications/LockedShields13_AAR.pdf. Accessed 2016-04-02.

[8] Microsoft. Common Types of Network Attacks . URL https://technet.microsoft.com/en-us/library/cc959354.aspx. Accessed 2016-04-27.

[9] Offensive Security. Client Side Attacks . URL https://www.offensive-security.com/metasploit-unleashed/client-side-attacks/. Accessed 2016-04-27.

[10] Aare Reintam, Exercise Director in Locked Shields 2016, personal communication, 2016.

[11] Heliand Dema, Red Team Client Side Attacks Sub-Team Leader in Locked Shields 2016, personal communication, April 2016.

[12] Elvis Paat, User Simulation Team leader for Locked Shields 2016, personal communication, 2016.

[13] Ragnar Rattas, Green Team Leader for Locked Shields 2016, personal communication, 2016.

[14] Jarkko Huttunen, Yellow Team leader in Locked Shields 2016, personal communication, 2016.

[15] Naumanis Erki. Centrally Managed Network Traffic Generation for Cyber Exercises . Master's thesis, Tallinn University of Technology, 2014.

[16] VIX API Reference, 2016. URL https://www.vmware.com/support/developer/vix-api/. Accessed 2016-05-22.

[17] vSphere PowerCli Reference. Invoke-VMScript, 2013. URL https://www.vmware.com/support/developer/PowerCLI/PowerCLI55/html/Invoke-VMScript.html. Accessed 2016-04-02.

[18] Daniele Catteddu and Giles Hogben. Cloud Computing: Benefits, risks and recommendations for information security, Nov 2009. URL https://www.enisa.europa.eu/publications/cloud-computing-risk-assessment/at_download/fullReport. Accessed 2016-04-27.

[19] Andrie de Vries. The most popular programming languages on StackOverflow, 2015. URL http://www.r-bloggers.com/the-most-popular-programming-languages-on-stackoverflow/. Accessed 2016-04-03.

[20] Erik DeBill. Module Counts, 2016. URL http://www.modulecounts.com/. Accessed 2016-04-03.

[21] IANA. Service Name and Transport Protocol Port Number Registry, Ports 3389 TCP and 3389 UDP. URL https://www.iana.org/assignments/service-

names-port-numbers/service-names-port-numbers.xhtml?&page=64. Accessed 2016-04-30.

A. Appendixes

A.1 Semi-Structured Interview With Aare Reintam

Interview with Aare Reintam, the exercise director for LS 2016 CDX, was conducted in 31st of March during the Final Planning Conference event of LS 2016. Interview was held face to face, in the format of semi-structured interview, where the main topics were known beforehand to the author and deviation from those topics was allowed. Interview lasted approximately 30 minutes.

1. What are the main tasks of the UST that they have to tackle?

During the exercise, the UST has to tackle three main tasks. First and the most important task for them is to help the Red Team CS attacks sub-team to execute their malicious software inside the workstations that are defended by the Blue Teams. Secondly, they need to conduct service checks, to verify that the Blue Teams have not disabled some services or their functionality. Third, if shortcoming in the provided services has been identified, the UST has to demand from the Blue Teams that they restore and keep those services functional and available.

2. How many virtualized services do the UST has to check?

This number varies from year to year, as in two exercises are not the same. For example in the year 2015 the number of services inside a Blue Team network, that the user simulation team had to check, was eleven. In the year 2016 this number was twelve. Also, they have to check all these 12 services from at least 15 different VM-s.

3. What are the restrictions from the exercise point of view to the UST framework?

Since the exercise is complicated and it grows every year, then scalable automation is necessary and vital! However, the User Simulation team must be in control of the automation. If something fails, they must be able to take over and start the activities

manually.

3.1 So eliminating UST completely is out of the question?

Yes, at the moment. We still need the "human touch" in the exercises and behind the workstation that the Blue Team are defending.

4. What would you say, is the biggest shortcoming of UST now ?

The main disadvantage is that the User Simulation Team members are changing each year. We give the cyber security master students an opportunity to take part of that exercise and every year we give the possibility to new students. Our aim is to give them the experience to see a large scale international cyber defense exercise from inside.

5. What is the structure of the White Team?

White Team structure is as follows:

- Exercise control:
  - Leader, who is responsible for running the exercise and deciding when to start certain phases;
  - Deputy Leader;
  - Schedule Master whose responsibility is to keep the schedule;
- Communications team:
  - Leader, whose responsibility is to prepare the communications plan and coordinate the work;
  - Blue-White Liaison Officers, whose responsibilities are asking and providing feedback from and to the Blue Teams;
- User Simulation team:
  - Leader, who makes the plans and coordinates the work;
  - Team members, who simulate normal and dangerous user activities. The Objective is to simulate client-side attacks in cooperation with Red Team. This includes but is not limited to clicking on malicious links, opening malicious documents and mail attachments. They also are responsible for validating the functionality of Blue Team' provided services;

- Injects and scoring:
  - Inject Master and supporters, who plan the scenario injects and coordinate the overall plan for all injects. They also evaluate the scenario injects and assign scores;
  - Media Simulation Team, whose responsibility is to assign score for media responses;
  - Legal Team, who are preparing and leading the legal play;
  - Scoring Master, who is overall responsible for scoring;
  - SITREP Evaluator, who evaluates the periodic management reports (SITREPs) the Blue Teams are tasked to provide.

A.2 Semi-Structured Interview With Mehis Hakkaja

Interview with Mehis Hakkaja, the CEO of Clarified Security and the leader of Red Team in LS CDX, was conducted in 2nd of February. Interview was held face to face, in the format of semi-structured interview, where the main topics were known beforehand to the author and deviation from those topics was allowed. Interview lasted approximately one hour.

1. What needs to be considered when building up a CDX

For the CDX to have a real world impact outside of the simulation, then the exercise must be modeled after the real world. Since defenders are playing catch-up with attackers, emerging trends must be also introduced.

2. What would you say has been the biggest shortcoming from the UST by your opinion?

In my opinion, as the overall leader for the Red Team, the issues with UST have been speed, scale-ability and clarity of communication. Since the communication between the Red Team and the UST so far have been mostly done by hand via Jabber chat medium, it is error prone and misunderstandings get worse as the exercise grows bigger. If the communication could be improved from both sides, Red Team and UST, then the overall workflow of both teams would improve a lot.

2.1 What are the issues with communication at the moment exactly?

The issue is cumulative from many small things when interaction and dependability of so many people is involved. One issue has been to ensure that tens of Red Team members stick to the UST emulation requests format and provide all the necessary details, such as what browser should be used in executing the request. Also reliable and reasonably fast feed-back system from UST players is needed. The tasks that the Red Team sends to the UST can take a lot of time and may require further interaction between UST and Blue Team or via White Team Liaisons to Blue Teams. Since most UST members are filling the role temporarily and for the first time, lots of confusion about workflows is inevitable without some (automated) enforcement or assistance. For example, some UST members provide quick feedback of success and any issues while others are very engaged in fulfilling the request, but forget to update Red Team of reasons holding them back. This inherently produces inconsistencies in the Red Team workflow and can slow things down. Also, since UST players have additional routine checklist of activities to perform

in certain time intervals, the less confusing and time consuming the Red Team request and interaction with Blue Team is, the better.

3. How has the exercise grown over the years?

CCDCoE large scale cyber exercises series stared for me already in 2010 with the exercise Baltic Cyber Shield where I composed a Red Team of about 20 members that had to entertain 6 Blue Teams in order to provide the game-play. LS naming convention goes back to 2012 when I had to compose a team of nearly 40 Red Team members to provide game-play for already 9 Blue Teams. In 2013 LS the Red Team was a little over 40 to evenly stress 10 Blue Teams. In 2014 it took almost 50 Red Team members to provide game-play for 12 Blue Teams. 2015 LS, with over 50 Red Team members having to scale for 15 Blue Teams (initially planned 16), was starting to really stretch workflow scale-ability. Prediction of around 20 Blue Teams for LS16 made it even more obvious that various workflow enabling tooling had to be improved or invented. When we look back to the UST role being played by 3-5 persons in the first years vs. already around 18 in LS14 and over 20 in LS15, the scale-ability and automation/tooling requirement becomes very obvious.

4. How is the Red Team built up?

In LS15 the Red Team of over 50 members was made up from three main sub-teams: network attacks team (or NET for short), client side attacks team (CS) and web application attacks team (WEB). All of those teams have their own separate objectives assigned to them. Network attack team is responsible for attacks on the network level, client side attacks team is responsible for compromising internal networks and lateral movement within internal networks of Blue Teams. The WEB attacks sub-team is responsible for attacks against web applications and other DMZ hosts. They provide more overt attacks during the initial phases by attacks like web site defacement or content destruction in order to draw attention away from more covert CS initial compromise attempts. However data theft is one of the less visible examples of WEB attacks.

4.1 What methods do the Red Team CS attacks sub-team use to penetrate the Blue Teams systems?

Client-side attacks sub-team emulates APT (Advance Persistent Threat) type threat actors and thus need an initial foothold in the internal networks which in turn typically in real world is achieved with attack vectors like “spear fishing” or “drive-by attacks” - thus, requiring user interaction. While there are some other initial compromise alternatives,

such as pre-planted known malware, such access is meant to be discovered and real initial foothold early in the game relies on UST players on all Blue Teams. Also, in later phases, if Red Team CS team gets fully kicked out of certain network segments, attempting to re-gain foothold yet again may require multiple retries with UST players. This becomes even harder and involved process once defensive measures and decent monitoring is finally put in place by the Blue Teams.

4.2 What is done after the initial compromise and which methods are used to achieve them?

After the initial compromise the Red Team CS attacks sub-team usually escalates privileges, attempts to achieve persistence withing gained privileges and usually will leverage the initial foothold for lateral movement. This may be required to achieve predefined objectives or to compromise other network segments when other easier options are exhausted. For escalation and lateral movement in Windows environment PowerShell and PsExec have proved to be quite useful tools in the previous years.

4.2.1 Has the success of PowerShell and PsExec reflected on the defense methods that the Blue Teams set up in any way?

Yes. Each year we encounter more teams that quickly disable or properly limit the use of those methods in the virtual environment, making it harder to perform initial compromise, escalation attacks and lateral movement through the Blue Team networks. This may yet again require going back to basics and to use more of the UST players for re-gaining lost access or gaining access in areas where escalation and lateral movement would have done the trick more easily, without user interaction ever being needed after the initial compromise.

A.3 Semi-Structured Interview with Ragnar Rattas

Interview with Ragnar Rattas, the leader of Green Team, was conducted in 28th of March. Interview was held over the Skype communication channel, in the format of semi-structured interview, where the main topics were known beforehand to the author and deviation from those topics was allowed. Interview lasted approximately one hour and fifteen minutes.

1. What is your opinion in for the UST automation framework? Is it needed?

LS is growing each year - both in the number of participating teams and also in the number of systems that UST must use. Manual work conducted by real persons does not scale. Automation is a must if similar quality of UST is expected with growing exercise complexity. The scalability of manual work is hard already by physically accommodating more and more people in the exercise. Currently there is one UST member per one Blue Team. This means that there are up to 75 systems that the UST member might have to interact with.

2. From the Green Team side, what are the most important things to consider?

1) Scalability - solution must support additional teams or workstations;

2) Usability from UST perspective - solution must be usable with limited or no prior training by people without extensive IT background

2.1 What about integration with existing tools?

System must integrate with currently available tools and solutions. Due to already existing technical complexities in Locked Shields exercise additional software or configuration changes must be kept minimal (integration with Collaboration Environment based password management system, use VMware Tools instead of custom agent running on host, use standard administrative solutions like RDP for connections instead of custom protocols etc)

2.1.1 Should we use or avoid using Management network for the communication inside the framework?

Management network is preferred for tool operation due to fact that many Blue Teams break their Gamenet connectivity This applies only to accessing the workstations - Red Team payloads must be delivered using as much as possible Gamenet itself Rationale:

MGMT network is out of the scope of Blue Team defensive operation.

3. Do you have any suggestions of how the framework should work on the technical level?

Yes, additional Red Team payload delivery vectors should be considered - via e-mail, usb sticks, etc

4. From the GT perspective, are tools like PowerShell Remoting and PsExec used to manage and support the Windows systems?

Yes, they are, extensively In theory they should not but in practice there is always need to apply some last minute fixes or changes and windows remote MGMT tools are only options for doing fixes to 1000 of VMs. Also these tools are considered as regular administrative tools also in real world then they are enabled by default on every target

4.1 As I understand, this is done before the execution? Are they still used to support the Blue Teams during the execution? Furthermore, are the Blue Teams allowed to remove or disable those tools?

GT uses them before execution, during execution we only interact with systems if we discover errors or mistakes made by GT. We do want to give as much freedom to Blue Teams as possible so the rules do not regulate if these tools can or cannot be disabled. It is up to each blue team. In practice they are commonly used by the blue teams, so they only apply proper security configurations and leave them enabled because they use them for their defensive actions.

A.4 Updater script for User Simulation Team Client Application

```
get-process node | Stop-Process
get-process mstsc | Stop-Process

# Download the file to a specific location
$clnt = new-object System.Net.WebClient
$url = "http://redrobin.ex/client.zip"
$file = "C:\Users\ls16\Desktop\client.zip"
$clnt.DownloadFile($url,$file)

# Unzip the file to specified location
$shell_app=new-object -com shell.application
$zip_file = $shell_app.namespace($file)
$destination = $shell_app.namespace("C:\Users\ls16\Desktop\")
$destination.Copyhere($zip_file.items(), 0x14)

Remove-Item $file;
cd C:\Users\ls16\Desktop\client\
Start-Process C:\Users\ls16\Desktop\client\run.bat

Start-Process chrome.exe http://localhost:3000
```

Code example A.1: Updater script served from the Central server for User Simulation Team client application

A.5 Semi-Structured Interview with Heliand Dema

Interview with Heliand Dema, the leader of Red Team CS attacks sub-team, was conducted in 28th of April. Interview was held face to face, in the format of semi-structured interview, where the main topic, Red Team feedback about the developed framework, was known beforehand to the author and deviation from that topic was allowed. Interview lasted approximately 40 minutes.

1. How long have you participated in the Locked Shields event?

I have been part of LS as a Red Team member since the first exercise in 2012. The first year I was a member of the CS team. During LS16 we had over 35 members in the CS team. This team is split in five sub-teams, each with a technical leader, reporter and five attacking members. At the moment I am the overall CS team leader as well as the technical leader for sub-team number one.

2. How was communication with the User Simulation Team carried on during previous Locked Shields

Our main communication channel with the UST was Jabber (XMPP), an IRC-like channel where we were submitting our requests and waiting for confirmation. Initially we were sending the requests in the main channel, however this created a huge confusion since sometimes requests went unnoticed due to the big amount of traffic-flow or UST member missing some of them. Also it was really difficult to keep an eye for confirmation on the chat while attacking at the same time. Later we moved to private messaging which was more one-on-one communication however since Jabber does not offer an easy logging system, it was difficult to read previous sent messages. Also another issue in the past has been continuous disconnection from Jabber (mainly due to network issues). Most of the times we had to physically go to the UST and ask for confirmation, which eventually means we were losing more time moving around and also generating more noise. On average it took about fifteen minutes to get any kind of response back from the UST member in previous years.

3. Was this framework useful for the Red Team Client Side attacks sub-team?

This framework prove to be very useful for the Client Side team this year. Not only we have the possibility to receive a confirmation from the UST, but by choosing the attacking machine from the given list we were also excluding the human error, making sure that the UST was receiving the correct target machine. The chat tab gave us the

possibility to communicate with the UST member without the need of moving from our seat. Best of all, the event feature, for the first time we can see in real life when the UST member starts processing our request and what was the status of it. A real life saver for anyone in the CS team. This was the first LS, where communication between Red Team and UST was mostly carried out on-line without any physical interaction, and all thanks to the framework, saving us, the Reds, not only valuable time, but also receiving very informative and useful feedback.

4. Did the UST members seem more responsive this year?

Thanks to the framework and also the training delivered to UST from the developer, the UST members were very fast at processing our requests and also very responsive in their feedbacks. The human error was limited to almost none thanks to the nature of the tool where everything is point-and-click. Very rarely we had to write additional requests/info in the chat.

5. Any requests for the future development?

This was the first time during the past five LS that we tried to enhance UST's work. The framework designed proved to be a real life-saver not only for UST but also for the Red Team. Improvements are always welcome and beneficial for both red and UST. Currently the tool is limited to only windows OS, supporting more OSs might be something to consider in the future.

A.6 Semi-Structured Interview with Elvis Paat

Interview with Elvis Paat, the leader of UST , was conducted in 29th of April. Interview was held over the phone, in the format of semi-structured interview, where the main topic, UST feedback about the developed framework, was known beforehand to the author and deviation from that topic was allowed. Interview lasted approximately 24 minutes.

1. How long have you participated in the Locked Shields event?

This is my first year participating in the LS event and my role is User Simulation Team leader. I have participated in the back end of the exercise before.

2. Was this framework useful for the User Simulation Team?

This framework was very useful in improving the workflow and reducing the overhead for the UST. There were no significant problems detected with the framework. The framework was especially useful for password synchronization, opening and managing connections to the Blue Team systems, performing service checks and communicating with the Red Team.

3. How was the communication with the Red Team?

Very good. There were no problems. The tasks came, they were completed and feedback was given. This year there were only one or two UST members, who struggled with the tasks, but overall performance was very good. There was also one case where a Red Team member was a little too active, constantly overwhelming single UST member with requests.

4. There was one case, where a User Simulation Team member did not use framework for code execution at all, any ideas why?

They probably opted out to using the framework just for management purposes to log into the Blue Team system automatically and then doing everything else manually.

5. What would you say was the greatest benefit from using this framework?

The sheer number of interactions that the UST members were able to make. Our main goal when managing the UST team, is to reduce the time between service checks, and this was certainly achieved. Also it was very good to see how quickly the features can be removed, if they start causing problems.

6. Any suggestions for future developments?

It would be nice, if there was a “UST Management tool” for the framework to see how active the UST members are. Who is doing their job more successfully and who is struggling. It would be nice, to get the Blue Team RDP availability checking feature back, to improve the usability of the framework. Furthermore, it would be beneficial if the framework chooses a random credentials every time the UST member tries to log in.

7. Any other remarks?

The tool was great. It did not crash. Debugging and fixing issues was fast. For future development, a Selenium scripting could also be considered, to implement automation even more deeply. Also, perhaps there could be a new state introduced for statuses of the feedback for the Red Team. Mainly the one that I think was missing is “delayed”. This would be triggered when a UST member receives task, starts processing it, but faces some difficulties with the Blue Team - such as workstations being unavailable, that have to be resolved before a task can be completed.

A.7 Architecture of the Framework

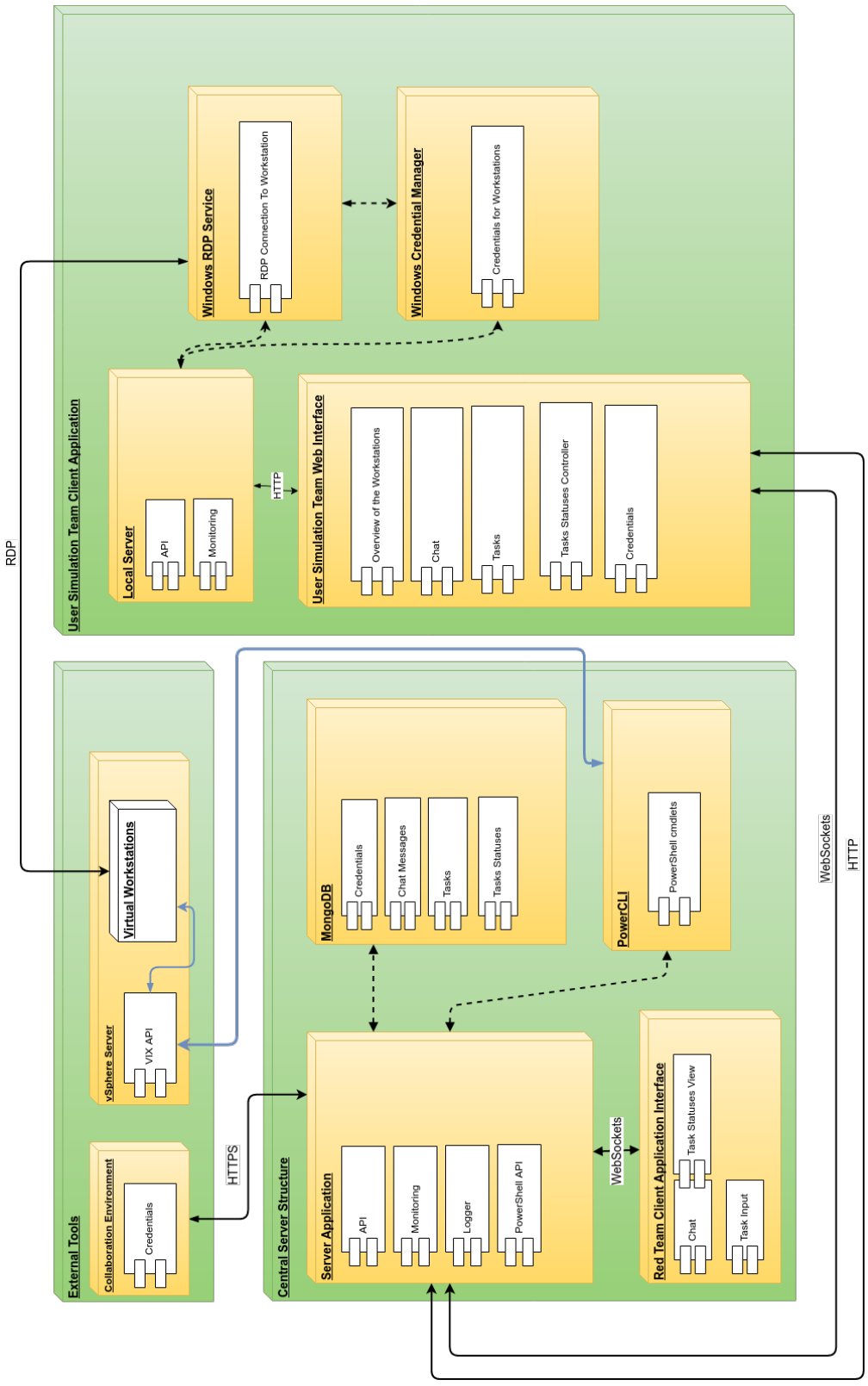

Figure A.1: Framework Architecture