TALLINNA
TEHNIKAÜLIKOOL

Cyber Range Digital Library Study

Tallinn 2017

This research was commissioned by the Estonian Ministry of Defence as part of the “Valdkondliku teadus- ja arendustegevuse tugevdamine” (RITA) program. The project is funded 50% by the European Regional Development Fund and 50% by the Estonian Ministry of Defence.

Executive summary

A Cyber Range can be described as a virtual training ground for cyber security specialists. Estonia is in the process of developing the NATO Cyber Range, which is able to host a variety of cyber security exercises for the benefit of the Alliance. Among other benefits is the opportunity to share and reuse exercise content and thus reduce the overall cost of exercise development. This functionality will be provided by the Digital Library component of the NATO Cyber Range. Its main function is to serve as a shared development and storage environment for cyber security exercise related digital assets, such as texts, images, video, configuration files, scripts, executables, virtual machine images, etc.

The purpose of this study is to identify and analyze the technical and procedural requirements that are relevant in the Digital Library context. Based on expert interviews the study proposes a number of recommendations for the design and development of the Digital Library. The study includes an overview of the State-of-the-Art, the limitations that currently hinder reuse, use cases for assets in the Digital Library, various considerations for making the assets reusable, the Digital Library deployment options, as well as the recommendations for overcoming identified challenges.

This study was commissioned by the Estonian Ministry of Defence and financed by the European Regional Development Fund and the Estonian Ministry of Defence. The study was conducted by a joint team of experts from Tallinn University of Technology and GuardTime AS.

1. Introduction

A Cyber Range can be described as a virtual training ground for cyber security specialists. Estonia is in the process of developing the NATO Cyber Range (NCR), which is able to host a variety of cyber security exercises for the benefit of the Alliance. Among other benefits is the opportunity to share and reuse exercise content and thus reduce the overall cost of exercise development. This functionality will be provided by the Digital Library (DL) component of the Cyber Range (CR). Its main function is to serve as a shared development and storage environment for cyber security exercise related digital assets, such as texts, images, video, configuration files, scripts, executables, virtual machine images, etc. In DL each asset contains the digital representation of the entity, as well as the corresponding metadata and a unique identifier (Figure 1). The metadata is required to make the DL searchable. The unique identifier allows for quick access to the correct asset.

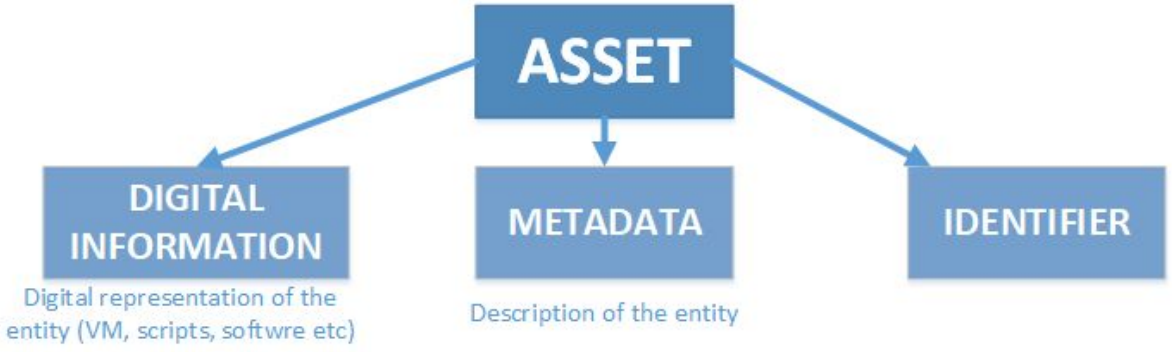

Figure 1 - Asset description diagram.

As exercises tend to grow in size and complexity, the need to reuse assets becomes more pronounced. It is especially true in case of large flagship exercises like “Locked Shields”. Developing a complex, non-trivial and realistic environment for an exercise requires significant investment in terms of time and money. The return on this investment would be increased remarkably, if (a part of) the developed assets can be reused in the follow-on exercise of the same series, or in different exercises.

Today, it is difficult to reuse assets as there are many different parties who are involved in creating content and there is no common environment or ecosystem that enables easy creation, reuse, collaboration and exchange of DL assets. Parties involved in asset development for DL and exercises or training events have different behavioral patterns, especially when it comes to documentation. One of the reasons for this is that asset documentation and exercise development processes are not standardised enough. The asset documentation and development process has to be standardized eventually, but in a manner that all the relevant users have the required level of freedom of choice and flexibility. Today, for example, “Locked Shields” uses a Wiki and “Cyber Coalition” uses SharePoint but the function of these solutions is similar. Hence the idea of DL, which would provide a more ordered approach for sharing content across different exercises and ranges.

DL is more than storage; it is what makes things reusable, enables development and collaboration. A DL that contains all necessary parts for exercise development would:

1. save time in exercise planning and development, if parts of the exercise can be reused across exercises;
2. improve quality of reused components, as the component developers benefit from lessons learned over several use cases, and this in turn will lead to higher-quality and more resilient exercises;
3. creates an asset development ecosystem that is not just exercise driven, meaning that DL participants can also submit assets that were not developed for a specific cyber security exercise, but were designed to be reusable (stock) content in the first place.

The DL should be separate but integrated solution where the library:

1. is simple enough to store DL assets that can include scripts, VM's, OS's, software, configuration files also texts, visual, audio and video materials;
2. has means for organizing, storing, and retrieving the assets and media contained in the library collection;
3. stores assets that meet the exercise's specific requirements;
4. provides effective means to find objects that meet the requirements;
5. informs a user how well an object has been documented and tested and whether claims have been verified;
6. informs users of software license terms.

The purpose of this study is to identify and analyze the technical, procedural and resource requirements that are relevant in the Digital Library context. Based on expert interviews the study proposes a number of recommendations for the design and development of the Digital Library. The study includes an overview of the State-of-the-Art, the limitations that currently hinder reuse, use cases for assets in the Digital Library, various considerations for making the assets reusable, the Digital Library deployment options, as well as the recommendations for overcoming identified challenges.

1.1 Context

The Digital Library is a component of a larger system, namely the NCR. This larger system consists of other components whose functionality is described at a high level in the NATO Type B Cost Estimate (TBCE) (Estonian Ministry of Defence 2017a). The TBCE describes in broad terms the main role of each component. The main role of the Digital Library can be summarized as follows:

1. provide the functionality required to manage the development and storage of assets such as network configurations, virtual machines, operating system images, software applications, simulation specifications, scenarios, and training materials, as well as any other reusable asset;
2. store assets and expose these to cyber range users through the asset management system (AMS) in order to allow for their reuse within and across activities;
3. provide a development and testing environment to allow asset development and testing according to modern practices, including version control sub-system.

Based on this, the DL needs to be accessible directly as well as through the AMS, as illustrated in Figure 2.

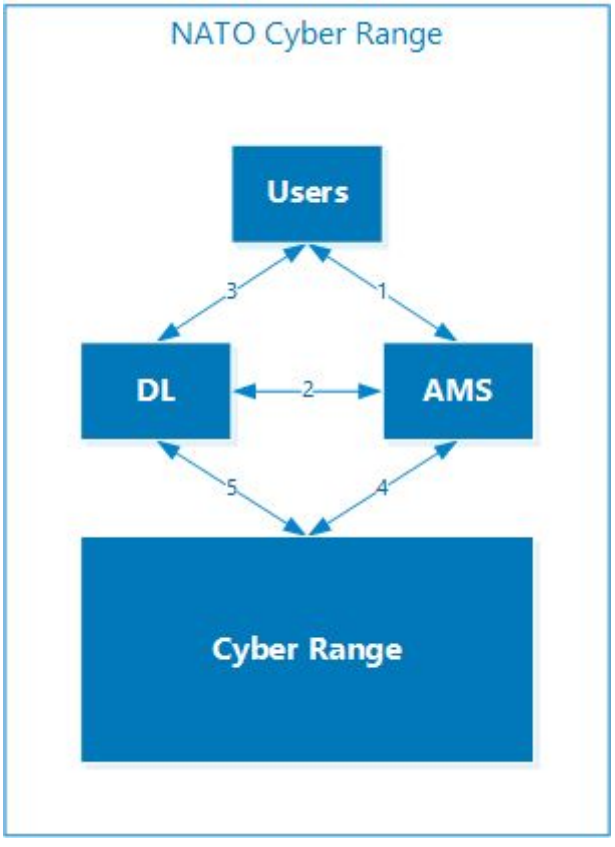

Figure 2 - NCR dataflow.

The dataflows of Figure 2 are explained as follows :

1. Dataflows 1 & 2: When planning and executing activities, users will access the DL via AMS.
2. Dataflow 3: Users may also access the DL directly in order to manage assets outside of the context of an activity. In some cases, it may also be that as part of an activity, users access assets directly from the DL.
3. Dataflow 4: Via the AMS, users may instruct that assets be deployed on the cyber range, and prepare the cyber range for receiving an asset from the library.
4. Dataflow 5: The DL will transfer assets to and from the cyber range.

It is also important to note that the CR is a very dynamic environment compared to a cloud environment in typical production use. CR supports relatively short-lived activities during which a large number of systems are activated and deactivated as well as regularly modified, whereas a cloud environment typically supports systems that are kept running for long periods of time. This subtle difference impacts the requirements and can make some solutions more suitable than others.

For other entities the Digital Library shall be accessible over VPN connection. After the VPN connection is established the above mentioned entities can access the DL directly or through the AMS, as illustrated in Figure 3.

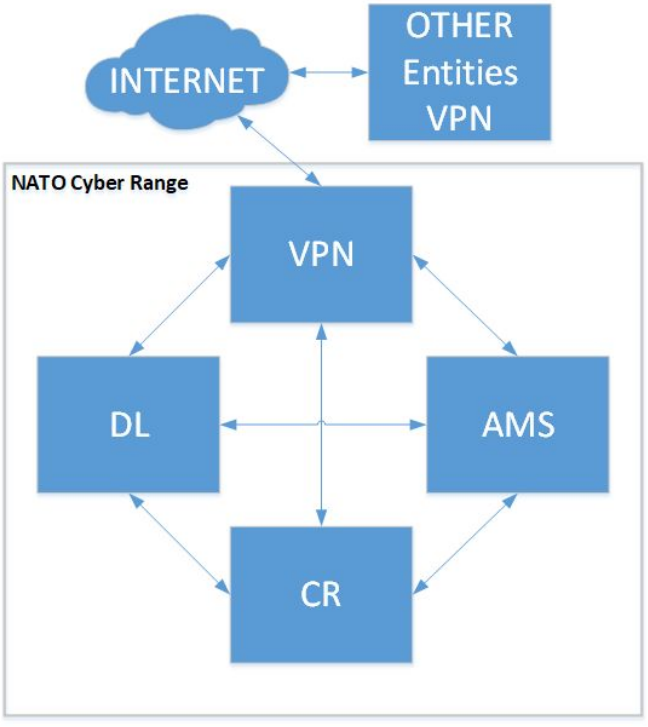

Figure 3 - Context diagram, including external entities and their relationships with the DL.

Finally, as outlined in the TBCE, it is important to note that the NCR capability is a long-term investment broken down into three phases. The first phase consists of building up the functionality and capability of the Estonian Defence Force Cyber Range (EDF-CR) at the unclassified level. The second phase consists of implementing a classified cyber range,

mostly through the re-use of the software deployed at the unclassified level. The third phase consists in the implementation of a federation of cyber ranges within NATO, integrating in some way NATO and National capabilities in a coherent whole.

1.2 Research Problems and Objective of this Study

The objective of the study is to identify the technical and procedural solutions for reusing NCR components that were developed for exercises. The results should include the technical and functional requirements for the NCR DL.

The research problems of this study are (Estonian Ministry of Defence 2017b):

1. Why has there been no component reuse so far?

   This question is discussed and answered in section 3.

2. Which use cases enable CR digital asset reuse?

   This question is discussed and answered in section 4.

3. What properties do the CR digital assets need in order to be reusable?

   This question is discussed and answered in section 5.

4. What part(s) of the CR digital asset reuse can be automated?

   This question is discussed and answered in section 5.7.

5. What other prerequisites are needed for CR digital asset reuse?

   This question is discussed and answered in sections 6 and 7.

The study will also provide recommendations for work-processes and the technical and functional description of the CR DL in sections 8 and 9.

1.3 Method

This study analyzes data from three different sources: literature, expert interviews and the authors' own experience with cyber ranges. Since the topic is relatively new, there are few relevant literature sources available. Therefore, the focus is on the expert interviews, which are complemented by the authors' experience, where appropriate.

The expert sample was chosen using a combination of convenience and snowball sampling methods. Convenience refers to ease of access, which was a limitation imposed by the relatively short period and low budget of the study. As a result, the interviewees were selected from Estonia and mostly included members of the Estonian Defence Forces, the NATO CCDCOE and Tallinn University of Technology who are among the cyber security exercise development community. Snowball refers to a sampling technique where

interviewees can propose other experts to be included. The list of interviewees was approved by the Ministry of Defence.

The interviews were conducted using a questionnaire (see Annex A) with open-ended questions and a semi-structured interview format. Open-ended means that the questions did not have pre-determined answers. The Interviewees were free to use any terminology they wished and answer in as much depth as they could. Semi-structured interview format allowed the interviews to deviate from the questions, if the interviewer saw it as relevant to the study.

It quickly became apparent that after the first few rounds of interviews there were no significant additions or deviations from the interviewees. This is likely due to the fact that the selected interviewees form a tight knit community where these problems and possible solutions have been discussed for years. Therefore, after conferring with the Ministry of Defence, the interview round was cut short and the authors shifted their focus on the analysis instead.

Information from the interviews was analyzed and synthesized into the findings presented in this study. The authors decided not to identify the holders of specific views in this study, in order to encourage a more free expression of ideas by the interviewees. In cases where there was significant disagreement between interviewees, the wording is chosen to reflect the underlying issue and to describe the key views.

2. State-of-the-Art

What is needed is a state of the art CR Institutional repository software which is designed for archiving, organizing and searching all DL assets. Taking into account that the DL has to store all kinds of file formats, the closest corresponding solution to build a state of the art DL solution for the NCR would be a Digital Library, Definitive Media Library (DML) or combination of both of them. From the NCR concept diagram (Figure 4) it is visible that the Library or DL in case of this study is a centric service that supports development and other NCR operations. In many aspects it should be a DML but it should incorporate also elements from regular digital and digital media library.

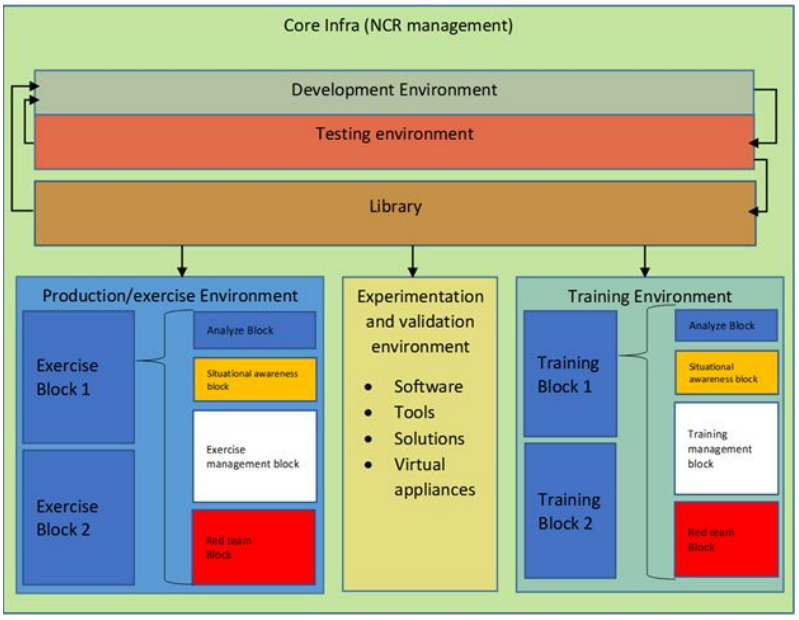

Figure 4. NCR concept diagram (Estonian Ministry of Defence 2015)

A formal DML is the authoritative logical library of configuration item (CI) media, including applications, documentation, licenses, system images, packages and multimedia files of a defined scope (Spafford, 2010).

In the NCR DL case we should have a combination of regular Digital or Digital Media Library and Definitive Media library as we want it to be available and open for CD related development, research and collaboration.

The State-of-the-Art DL would have the following properties or functionalities:

1. Workflow:
    a. Asset documentation
    b. Asset submission/loading
    c. Asset organization
        i. Metadata, tagging
    d. Asset publishing
        i. QuOS, Asset loading
    e. Asset indexing and storage

    f. Asset delivery (services)
2. Key components:
    a. Asset submission
    b. Extraction or creation of metadata or indexing information that describes the asset in order to facilitate searching and discovery, as well as administrative and structural metadata to assist in asset viewing, management and preservation.
    c. Storage of asset and its metadata in storage/repository
        i. Storage service includes rights management
        ii. Intellectual property management
        iii. E-commerce functionality to handle accounting and billing
    d. Client services for the browser
        i. Repository query
        ii. Workflow
    e. Asset delivery
    f. Access through a browser and API
    g. Network (private and public)
3. Software features:
    a. Support for different content formats
    b. Associate metadata with different assets
    c. Support for different file types
    d. Asset loading/publishing
    e. Indexing and storage
    f. Access and delivery
    g. Accessrights management
    h. Usage monitoring and reporting
    i. Preservation
        i. Link checks
        ii. Content refreshing and update
        iii. persistent asset identification
4. Interoperability (Open Archives Initiative)
5. Standards compliance (XML, Unicode etc)
6. Scalability

Other library management solutions that currently exist

There are several library management systems available but they are implemented either as Digital Media Libraries, Digital Libraries or Software Repositories for developers. Below are few examples of library management solutions that are close to the DL that is discussed in this document. However, there are no exact matches on the market today.

1. Fedora Commons - Fedora (or Flexible Extensible Digital Object Repository Architecture) is a digital asset management (DAM) architecture upon which institutional repositories, digital archives, and digital library systems can be built. It can store, preserve, and provide access to any type of file - no restrictions. Has a variety of storage options for files and metadata, including file systems, databases etc. Fedora provides a set of core repository services via RESTful APIs using modern web standards. By the description it has most of the necessary parts for DL and it is an open source platform (http://fedorarepository.org/features)

2. Microsoft System Center: VM Manager. The VMM fabric retains a library of file-based and non file-based resources that are used to create and deploy VMs and services on virtualization hosts. File-based resources include virtual hard disks, ISO images, and scripts. Non file-based resources include templates and profiles that are used to standardize the creation of VMs. Library resources are accessed through library shares. (https://docs.microsoft.com/en-us/system-center/vmm)
3. DSpace open source software is a turnkey repository application used by more than 1000+ organizations and institutions worldwide to provide durable access to digital resources. DSpace preserves and enables easy and open access to all types of digital content including text, images, moving images, mpegs and data sets. (http://www.dspace.org/)

3. Factors Currently Limiting Asset Reuse

The practice of reusing assets across activities is currently relatively limited according to cyber range users. Based on the interview results there are a number of reasons why this is the case.

First, in the absence of a digital library, current users have followed ad-hoc procedures to reuse assets that work within their community of interest. Generally speaking, these communities are the various groups of people that are involved in designing, preparing, and conducting cyber exercises. Reuse practices and associated repositories are community-specific and the degree of documentation varies. Reusable assets are therefore currently spread across several organizations and knowledge of these assets is not necessarily accessible by those not involved in their development, and certainly even less so across organizational boundaries. Without functionality to query the various repositories, users are often unaware of even the existence of reusable assets.

A second limitation on sharing stems from digital rights and intellectual property aspects. In some cases, assets may be subject to various copyrights. For example, a virtual machine may contain software whose license prohibits usage by anyone but the licensee. As well, developing assets often requires significant effort and results in the generation of intellectual property, even if not explicitly registered or formally recognized. Some organizations are not willing to widely share their work, particularly if there is no compensation or control of further sharing.

A third limitation comes from the difficulty of properly documenting assets to make them easy to find in the DL. Proper documentation takes time and effort. Sometimes assets may be completed at the last minute (or even during the exercise) and the documentation will be completely or partially omitted due to time restrictions. Lastly, there is always the chance of a developer forgetting to document changes.

A fourth limitation comes from the difficulty of changing established practices. Current users of cyber ranges have grown accustomed to creating assets and developed their own processes, and do not wish to invest the time required to make their assets reusable by others. The effort this would take is not worth what they expect back from the community. A rigid sharing-for-reuse system would further exacerbate this problem. Therefore, it is important to create a flexible system that suits the different approaches in use by the existing developer community. The functionality through which assets are documented and queried is especially important, as it must be easy, intuitive, and convenient to use. In particular, the effort required to document and store an asset for reuse needs to be balanced with the benefits that will be gained through reuse. Certainly the implementation of a digital library will need to include awareness-raising, dedicated training, and support and encouragement for reuse in order to be successful. The transition from the current practice to the use of a DL will require current range users to make an effort. In practical terms, this implies that developing an exercise will require more time and effort if assets are to be

stored in a DL for reuse, at least initially. The time saving effect will appear once the DL has enough reusable content.

Lastly, another limitation is the intrinsic problem that some assets need to have specific features that are often more easily created than searched.

4. Digital Library Use Cases

There are several use cases in which a DL can provide added-value, the main ones being:

1. Reuse of virtual machines
2. Reuse of automation tools
3. Reuse of configurations
4. Reuse of scenario components
5. Reuse of patterns
6. Storage of activity records

The following subsections describe the identified use cases and illustrate the reusability value of tangible activity components.

4.1 Re-Use of Virtual Machines

Certainly one of the most time and effort-consuming task in preparing an exercise, training or experimentation environment is the creation of the virtual machines that deliver the simulated reality supporting the activity. If virtual machines can be stored in a DL for later reuse, the required effort to create an environment could be greatly reduced.

Similarly to other reuse use cases, virtual machines stored in a DL will need to be properly documented for future users to be able to determine their suitability for a specific need. In the case of virtual machines however, the amount of such metadata can be considerable due to their complex nature. It is thus not practical to fully document a virtual machine through associated metadata. Adding further difficulty, the metadata captured by one user might not fulfill all the needs of another user. Thus while it may be easy to find “possibly suitable” virtual machines using their metadata, a user may still need to activate them to confirm their suitability. Consequently, the reuse of virtual machines through a DL transforms the time and effort required to build a virtual machine with a specific configuration into an effort to find a suitable reusable virtual machine. And if no suitable virtual machine is found despite the time and effort spent searching, one still has to be built. The value of this use-case therefore largely depends on the associated metadata (coverage and quality), and diminishes quickly for use cases with highly specific requirements not captured by that metadata.

In some cases, reusability is also limited by the inability to align time information contained in the virtual machine with the time requirements of an exercise scenario. Indeed, for some exercises, time-related filesystem data as well as time-related information stored in files need to be carefully considered in order to ensure realism and create a coherent environment for the users. In situations where time-related filesystem metadata and other records of time need to be aligned to specific scenario requirements, reuse will not be possible except in the simplest of cases.

Another challenge with the reuse of virtual machines concerns licensing. Many operating systems and applications are only available under licenses that must be purchased and that

may not be easily transferable. While it is technically possible to put any virtual machine in a DL, in a number of cases reusing such virtual machines may breach contractual obligations and infringe copyright laws.

Yet another challenge concerns the dynamic nature of modern operating systems and applications. In particular, their auto-update functionality can cause a virtual machine to update itself automatically in a manner that makes it non-conformant to its own metadata and unsuitable for its intended use after activation.

Finally, the possible presence of malware is also a concern. Despite best efforts, a virtual machine stored in a DL could contain malware that has been introduced at some point of its development and prior usage. While this risk exists for a virtual machine built from scratch as well, the longer a virtual machine has run with connectivity to other systems (in particular the Internet) and the greater its software load, the higher the risk of undetected infection. In the absence of integrity verification, there is also the risk that a malicious user with access to a DL introduces malware into stored virtual machines.

Despite these challenges, the reuse of virtual machines remains a valid and high-value use case for a good portion of the full spectrum of cyber range activities because of the significant time and effort required to build them from scratch. Its value will increase if the DL facilitates and automates metadata management (including incremental evolution as recurring usage patterns emerge), ensures the integrity of assets, and addresses issues related to licensing.

4.2 Re-Use of Automation Tools

While modern virtualization environments provide the ability to replicate or clone virtual machines without much effort, most CR activities require that these be implemented with specific configurations, applications, users, and incorporated into a larger system (e.g. a Microsoft Active Directory Domain). As well, many CR activities require diversification of virtual machines in order to function properly and conform to the reality being simulated. The current best practice combines cloning with automated tools (e.g. scripts) for the tuning, diversification, and individualization of cloned virtual machines to reduce the time and effort required to generate an environment.

Since the development of the automated tools that deliver the required diversification and tuning of complex virtual machines itself requires significant time and effort, such tools are prime candidates for reuse. Simpler than virtual machines, automation tools are more easily satisfactorily characterized by metadata. A key aspect for their reusability value is their flexibility to function on a diverse range of systems. A tool that works on a family of operating systems will be more reusable than one that only works on a specific version. However, most tool developers do not have the time to test their tools on all possible versions. A quality assurance system that tracks the targets against which a tool has been successfully proven to work is thus necessary.

Two important aspects to further improve automation tool reusability is the development of a framework for the management of execution parameters and the development of a best

practice for modularity. Many tools will take user inputs that specify the particular details of the desired end state or the aspect that needs to be diversified for a set of targets. Having a framework for how these parameters are provided to a tool can greatly increase reusability by facilitating usage and allowing tools to be chained together.

Similarly, a modular approach to automating a set of changes increases reusability of tools as some of the tools in a chained set could be individually reused, whereas a monolithic tool would not have been suitable. In fact, the reusability of a virtual machine might be increased by making a very generic basic installation and developing a set of automation tools that further deploy software, adjust its configuration, and introduce user activity, for example. Guiding users on how to make more reusable virtual machines and automation tools through modularity will improve the value of a DL because its content will be more flexible and thus reusable for a larger set of activities.

Example:

1. gamenet infra elements like in-game ticketing system
2. Scripts, that is, scripts and vulnerabilities that were used ether in development or in exercise etc.

4.3 Re-Use of Configurations

The reuse of assets is not enough if we talk about fully automated CR. The assets need content and context. For content and context the assets are configured according to given parameters related to the CR exercise or other event. These configurations can be described, scripted and saved as configuration items. Items like:

1. vulnerability related assets (malware)
2. scripts and vulnerabilities that were used either in development or in exercise, etc.
3. dummy user names, list of known "bad" passwords, rainbow tables, etc.

4.4 Re-Use of Scenario Components

For every exercise new scenarios and related material is produced. Materials like:

1. maps (imaginary states)
2. images (of fictional characters, facilities and events, etc.)
3. audio-Video (press releases, interviews, etc.)
4. manuals and documents (fake documents, etc.)
5. background stories for countries, governments, organizations and people (language, culture, politics, economics, military power, etc.)
6. key players and their background stories (who, when, what, where, etc.)
7. companies (public, private)
8. websites (e-shops and social media)
9. user profiles

All these scenario components need to come with metadata that can be used to search them and to help to combine different elements.

4.5 Re-Use of Variables

For many exercises teh developers produce 20+ environments worth of information that has certain variables. Variables as called in case of vLM are to ease the customisation of VM's. In vLM there are three types of variables. (ByteLife OÜ 2014)

1. VM definition field patterns - patterns which values will be taken from already existing field in VM definitions (ByteLife OÜ 2014)
2. VM definition custom parameters - patterns that are added to VM definition by developers (ByteLife OÜ 2014)
3. VM count variables - variables that represent the count numbers of the VM in VM definition (ByteLife OÜ 2014)

For a better standardisation the DLI should also have a collection of ready made variables like names, usernames, passwords etc. These variables should not be limited only for VM definition but should be available for scenario development.

4.6 Storage of Activity Records

The conduct of CR activities also generates a lot of data as a by-product. Some of this data has potential for reuse. For example, traffic generated during red team vs blue team engagements in most cases reflects actual exploitation of vulnerabilities and transmission of malicious code. Captured files containing this traffic could be useful for testing malicious activity detection systems. As well, Tools, Techniques and Procedures (TTPs) for both blue teams and red teams can be analyzed across several exercises in order to generate statistical studies and help identify the best practices. In these use cases, the types of assets include packet capture files, system event logs, logs of actions taken, and patterns of usage/behaviors, amongst others.

Activity records can also cover metrics about range usage both in terms of scope and types of assets and performance metrics (processor, memory, storage, and networking usage). Studies spanning multiple activities can help better understand range requirements and planning for expansions and new ranges.

5. Digital Library Assets

5.1 Scope of Content

Based on the use cases described above a preliminary classification of reusable assets that could be stored in a DL is as follows:

1. **Operating Systems**: the first level of software installed on a host that provides the functionality required to run applications. In many cases, an operating system can also include some applications.
2. **Applications**: software that runs on an operating system to provide specific functionality.
3. **Services**: Composition of hosts, operating systems, and applications that deliver specific functionality, generally in a networked environment.
4. **Architectures**: Specifications of configuration and connectivity of services.
5. **Automation Tools**: Software that can increase efficiency by performing one or more action.
    a. **Deployment tools**: tools that help take a resource from a DL and instantiate it on a cyber range.
    b. **Configuration tool**s: tools that help set up an instantiated resource and adjust it to meet specific requirements.
    c. **Simulation tools**: tools that help generate artefacts that make a virtual environment appear more realistic (e.g. simulate user activities in the absence of actual users).
6. **Scenarios**: Elements that are used to build a story that supports an activity and engage the participants into it.
    a. Scenario components
    b. Storylines
    c. Media
    d. Behavioral patterns (e.g. threat actors)
    e. Simulated activities: data that can be used to simulate user activities in the absence of actual users, generally delivered through a simulation tool.

This taxonomy is an initial illustrative approach that expresses the scope of possible DL content. Given the complexity of the nature of reusable content, a simple taxonomy such as the one presented above will not be sufficient to provide the required search efficiency. The implementation of a DL must therefore consider developing a more advanced system that will help users efficiently search for assets based on their needs.

5.2 Asset Lifecycle Management

If a DL is taken as just a basic repository, the number of assets stored will eventually become so large that performance and usability will be reduced. While storage space can be increased over time, uncontrolled asset accumulation nevertheless impedes efficient searching. Furthermore, assets are not immutable. In many valid use cases, assets need to

be modified over time, either to fix errors or to increase interoperability and/or functionality. It is therefore important to consider the lifecycle of assets from cradle to grave.

As a DL mostly provides storage functionality, assets are primarily created outside of the DL as part of the development of an activity planned on the CR. From the DL's point of view, the asset's life cycle starts when it is first imported, regardless of where it was developed. While stored in the DL, the asset can be "deployed" on the CR, modified, exported to another DL, and eventually discarded. An asset should be discarded when it no longer provides any value (i.e. it will no longer be reused) and when its retention is no longer necessary for historical reference purposes.

It is expected that most assets will be modified outside of a DL, primarily while being used on the CR as the use cases for modification of an asset within a DL are quite limited. It is recommended that a basic edit functionality be provided from within the DL (regular text and binary editor functionality through the GUI) as a starting point and that more advanced asset editing functionality be implemented over time as valid use cases are clearly identified.

Assets need to be kept under version control. This is required because assets will likely be selected based on specific criteria during the planning of an activity and used at a later time when the various assets of the activity are actually deployed on the cyber range. If the asset has changed between the time it was selected and the moment it is deployed for use, its behavior may not be as expected by the user. By maintaining all assets under version control and using an asset identifier that references a specific version of an asset, users have a guarantee that what they will deploy is what they initially selected. Furthermore, by having version information, it is possible to build functionality that will verify if a newer version of an asset exists and allow the user to determine which version should be used. The DL should therefore provide functionality associated with version management including showing a history of an asset's evolution over time.

Assets can also be exported to another DL. Exporting an asset is different than deploying it on a CR as it implies that the asset will be stored in a different DL and be used from that DL, including possibly evolving over time. When a user exports an asset, its metadata including references to other adjacent versions should also be exported so that it can be also imported in the other DL (see Section 7.2 on challenges associated with asset identifiers and version control).

In addition, it will be necessary to assign and manage ownership of an asset over time, to identify the primary point of contact who can answer questions about the asset from other users, and to provide information about licensing the asset for use on a cyber range. When assets are no longer required, they will need to be deleted by an authorized user. Since this can be done in error, the assets should remain available for recovery (i.e. via a "recycle bin") for a configurable period of time. As well, it may be worthwhile to provide storage space at different performance levels. This would allow the DL to move assets not frequently used to slower, lower cost storage and provide better performance for frequently-used assets at the same price.

Another aspect concerns the evolution of assets over time. Given that their nature may change as they are modified, and that these changes could change their very nature, it may

be necessary to provide the means to “branch” out or “fork” an asset into two separate assets. For example, a post-deployment operating system configuration script could start with the basic functionality of setting IP addresses. Later, it could be modified to also add user accounts. It could then be discovered that the way it sets IP addresses doesn’t work on all versions of the operating system, and a user could take that functionality out. What started as a script to set IP addresses is now a script that adds user accounts. While this is a very basic example, the need to fork an asset should be more carefully studied. In any case, it is clear that the metadata associated with an asset needs to be also modified as the asset is modified to properly reflect its updated nature. As such, when a new version of an asset is stored, its metadata should also be amended as part of that submission of that new version through a single operation.

5.3 Metadata

Metadata in the case of this study is data that provides information about DL assets. In DL there are three types of metadata: descriptive metadata, structural metadata, and administrative metadata.

1. **Descriptive metadata** describes a resource for purposes such as discovery and identification. It can include elements such as name, abstract, developer, and other keywords.
2. **Structural metadata** is metadata about containers of data and indicates how compound assets are put together, for example, how different pieces are but together to form an asset. It describes the types, versions, relationships and other characteristics of assets.
3. **Administrative metadata** provides information to help manage an asset, such as when and how it was created, file type and other technical information, and who can access it.
    a. Means of creation of the data
    b. Purpose of the data
    c. Time and date of creation
    d. Creator or author of the data
    e. Location on a computer network where the data was created
    f. Standards used
    g. File size

5.4 Classifying Content

As the DL is expected to store a large number of assets, assistance in searching content will be required. A common approach is to use a “classification system” to classify assets in accordance with some instructions and provide functionality that utilizes this information to later help find the assets that meet a user’s need. In this context, the term “classification system” means all functionality related to assisting in the search for assets and their lifecycle management. Thus a user needs to provide information to the classification system at the moment of storage, and a user needs to use the search functionality based on this classification system at the moment of retrieval.

As is the case for all storage systems, there is a balance to be struck between the effort required at the moment of storage vs. the effort required at the moment of retrieval. While various schemes can be proposed, the right balance cannot be easily defined in advance and is therefore beyond the scope of this study. It is deemed unlikely that the right balance can be determined during the implementation from a simple analysis, so it is probably best to implement a DL with a configurable “classification system” consisting of various functional components that can be adjusted once a period of usage will have provided indications as what that balance is.

It is expected that the classification system will need to have a basic top-level taxonomy as a first mandatory component. It is expected that the use of additional classification components will likely depend on the type of asset selected from this top-level taxonomy. The DL should therefore have the means to configure the additional functional components that are presented to the user based on the top-level type of asset selected, as well as the means to set whether the use of these other components is mandatory or discretionary. Additionally, this should be adjustable during the operation of the DL, and such DL configuration changes should ideally trigger automated actions that help ensure existing content is adjusted to the new configuration. For example, if the use of a “tagging” functional component is made mandatory for assets of type “Operating Systems”, the system should provide functionality to help ensure existing “Operating System” assets in the DL can be tagged efficiently in order to conform to the newly applied configuration.

Table 1 provides initial insights into possible functionality of a classification system.

Functionality	Description
Top-Level Taxonomy	A top-level taxonomy would separate assets into distinct categories and provide the first, coarse grouping of assets that could help users search the DL and manage assets over time. A top-level taxonomy is recommended because the various assets in scope do have distinct characteristics that would make such a first-order classification easy to use and helpful. In defining the taxonomy and its taxa, the following characteristics of a good taxonomy need to be kept in mind: - Mutual exclusiveness: an asset fits in one taxon and no other ones. - Exhaustiveness: all assets will fit into a taxon, none are left unclassified. - Unambiguous: all users classify assets in the same way. - Repeatable: assets are classified in the same way every time. - Useful: the taxonomy helps the retrieval and lifecycle management of assets
Controlled tagging	A controlled tagging component allows users to select tags (a single word or a few words) from a list defined by the DL administrator. Tags differ from a traditional taxonomy in that

	any number of tags can be applied as opposed to having to assign the asset to a single taxon. By limiting the tags to come from a controlled list, spelling errors and variations are avoided, thus reducing the number of false negatives in search results. The drawback of controlled tagging is that users may not find the tags they are looking for, and any changes to the controlled list of tags will likely require a review of all assets to ensure they are appropriately tagged under the new approach.
Freeform tagging	A freeform tagging component allows the users to apply any tag they wish to an asset. Freeform tags are flexible when storing assets, but can result in false negatives in the presence of spelling errors and variations (e.g. “Windows” vs. “windows”). As well, the number of tags can become very large and make selecting tags in support of a search difficult. The meaning of some tags may also not be clear to all users, resulting in false negatives and false positives. Finally, without any control or guidance, a poor tagging practice can emerge and reduce overall usability and performance.
Ontology-based metadata	A functional component could provide various metadata based on a defined ontology. For example, it would be possible to extract basic parameters for virtual machines, such as number of CPUs, RAM, storage space, installed operating system and applications, etc.
Extracted metadata	A functional component could also expose some of the metadata automatically extracted from assets and allow it to be used in searches. The metadata can be specific to the taxon in the top-level taxonomy.
Pattern search	A functional component could also allow users to search for specific patterns (text or binary strings) in the actual asset data. This could also include regular expression matching.
Boolean logic search	All of the above functional components could be used with boolean logic (“and”, “or”, “not” and ordering parentheses) to allow users to formulate more complex queries.

Table 1. Initial requirements for the asset classification system

To strike the right balance, various configurations of the classification system could be trialed during the first period of usage of a DL.

5.5 Quality Assurance

Quality will play a large role in increasing asset reuse. If assets are poorly documented, they will not be found when users search for them. If assets are poorly implemented and fail to

function as advertised, users will see reduced benefits from using DL assets and will revert to implementing by themselves assets that otherwise already exist.

Quality Assurance (QA) consists of all processes and functions that help assure the quality of assets and their associated metadata. Use cases for the cyber range environment are generally quite similar in terms of quality requirements, and therefore it is expected that a single QA process will be needed. Such a QA process would need to cover:

1. In the case of an executable asset (e.g. an application or a script), verification that the asset functions properly on the specified execution targets (e.g. list of operating systems) and actually does what it claims to do.
2. Verification that the asset is reasonably free of malicious code.
3. Verification that the asset's metadata properly and fully describe it.

In the use cases envisaged for the cyber range, full QA automation will likely be impossible and that it will require human intervention. However the DL can provide partial automation that greatly reduces the associated effort. This can be through workflow automation, guiding users through the QA process and facilitating the capture of QA information.

QA information is another form of metadata that other users can rely upon while making decisions regarding the suitability of an asset for a particular use. It will be important that the various QA processes be properly documented so that users can make the necessary judgement of suitability. Basic information about the execution of the QA process should also be recorded as additional metadata for that asset. It will also be necessary to determine what kind of changes could be made to assets without causing the need to rerun the QA process. Finally, a determination will need to be made regarding who should execute the QA process. It may be that this can be left to any user if the process is easy to execute, or that it is best assigned to specific staff properly trained to execute the process.

All of these decisions will affect the level of effort associated with QA, and as is the case with many other activities, this likely follows the law of diminishing returns. The right balance will likely only be determined through usage, and might vary between different asset types. A flexible QA system that can support a range of different QA processes is therefore required.

5.6 Marketplace

During the course of this study, it was determined that there might be a valid use case for the "sale" of assets, as some entities may be more willing to make some useful assets available on a DL if they can be compensated. Generally speaking, the predominant incentive to share is the expectation of receiving in the future. However, in the context of cyber exercises, asymmetrical relationships can come to be rather easily as only a few organizations have the capacity to develop complex exercises while many more have the ability to run them if they can get assets without having to develop them. This may cause the benefits of a DL to diminish over time for the major producers as they no longer see value in widely sharing the assets they develop. However, if they can "lease" or "sell" their

assets, this compensation might be sufficient for them to continue sharing widely. Other users may be more able to pay than to contribute on a quid pro quo basis.

The implementation of a “marketplace” functionality would consist of giving the means to do the following:

1) Allow asset owners to set and advertise their assets, conditions of use, and price.
2) Facilitate buying and/or leasing of assets.
3) Restrict access to assets only to those who have legitimately procured the asset.
4) Account for the usage of assets so that cost can be appropriately determined.

The functionality for the above aspects does not necessarily need to reside within the DL. For example, it may be possible to integrate with an existing online payment system rather than implementing a financial transaction subsystem within the DL. In fact, the minimal functionality required is access control to assets, and everything else can be arranged externally to the DL.

In terms of type of transaction, it may be possible to rent or sell assets. For example, a company may agree to make its software available for a particular period of time on a cyber range in order to support an exercise. This is can be seen as equivalent to “renting” the software. As another example, an entity may wish to develop different background scenarios and sell them to exercise planners who would be free to reuse them in several exercises. More broadly speaking, even a full training program could be built through aggregated assets and sold in a DL. Financial compensation is an incentive to make reusable assets.

However, the establishment of a marketplace is not without its downsides. In addition to the added cost of implementing and operating this functionality, it can shift existing practices. Currently, the ecosystem around the cyber exercises executed on the EDF-CR are generally developed through contributions from different organizations willing to share the burden and the assets that can be later reused. If a commercial aspect is introduced, the current community dynamics will be affected. The relatively small number of asset developers might be spread over both approaches with neither reaching critical mass, thus limiting reuse.

Apart from this consideration, perhaps the most important benefit that marketplace functionality could bring would be to address the need for “exercise licenses” for software. Currently there is a large body of software that is needed in practically every exercise. Exercise organizers find their own means to license this software for the exercise, either through existing licenses or various interpretations of license terms. A better practice would be to provide the functionality to support licenses for cyber range activities that would cost much less than the typical full-usage license, with the cyber range providing assurance to software companies that reasonable mechanisms exist to ensure licensing terms are respected. These could be the automation of the deployment of the asset from the DL to the cyber range and the accounting of the the time it ran on the cyber range, without giving users access to the software itself. The major software vendors could agree to developing exercise licenses if the DL can provide the means for financial compensation for software usage.

5.7 Asset reuse automation

There are four considerations that are relevant from the perspective of automated asset reuse functionality:

1. Deployment: the user selects an asset from the DL or from AMS, and the DL and the AMS automates the deployment. User finds the asset on the AMS where he wants it without having to take the asset out himself.
2. Automated metadata extraction: scripts that are run on the asset when it is submitted and extract some key information that is used to automatically fill in the metadata.
3. Automated asset documentation: with standard documentation format it is possible to automate the documentation of assets. For that purpose a standard asset documentation formats should be developed.
4. Asset development: there should be DL asset automation service that is integrated to AMS in order to enable asset reuse so that the asset developer can start the assed automation process while developing the asset.

6. Establishing trust

One of the key issues for successful implementation of a DL is the establishment of trust. How will the various components of the system establish trust amongst themselves? Broadly speaking, there are two issues to consider here:

- Establishing trust between components (DL, AMS, cyber range, etc.)
- Establishing trust between the human users and each component.

The first aspect can be addressed through the use of a conventional machine-to-machine authentication method and encryption combined with minimum security requirements implemented in all components (technical implementations and procedures, possibly with audit reviews). With these in place, the various administrators can decide to trust other components that successfully authenticate. For the immediate needs of NCR (i.e. Phase 1, as described in TBCE), this is easily implemented using common authentication methods as all components are under the same authority. In future scenarios however, this may not be the case, and the decision of adopting a centralized approach based on a single root of trust such as a PKI or a decentralized approach will need to be taken.

In terms of encryption, the required protection could be provided at the application layer such as through the use of the Transport Layer Security (TLS) protocol, or at the operating system level such as through the use of IPSEC. While IPSEC may be a good option in the first phase of NCR, as the number of components deployed in different organizations and their functionality grow, this approach may be less desirable, as it would not provide the additional functionality and granularity that an application-level approach would provide. For this reason, the use of TLS is likely the best approach to providing confidentiality to inter-component communications. TLS can also provide authentication, thus addressing both issues at the same time.

The second aspect can be addressed through the use of a conventional user authentication method (e.g. username and password or the more complex approach of certificate-based authentication) combined with training. However, the issue is slightly more complex than for the inter-component aspect as even in its first Phase, NCR is expected to be used by a large community of users from different Nations and organizations. This is expected to quickly put a large burden on the operators of the range, as several hundred users can be expected within the first year of operation. The effort required to manage such a large number of user accounts is not to be underestimated.

To address this issue, it may be preferable to use a distributed identity management approach such as OpenID Connect. With this approach, the Estonian Ministry of Defence would trust some existing identity providers and grant access to specific users once they authenticate with the selected providers. Some user account management functions (e.g. password reset) would then be the responsibility of these providers. If desired, it could also be possible to delegate the authority to grant access to other organizations, thereby also delegating all aspects of user management to other organizations.

In terms of authorization, the use cases envisioned for the cyber range include a fair number of elements. For example, in the context of an activity on the cyber range, a particular user could be granted access to specific resources over a defined period of time. That user may wish to automate the deployment of a particular asset in a DL, which would be done through the AMS. This is a typical use case for an attribute-based access control (ABAC) mechanism, as such a system could take into consideration the assigned resources and timeframe in its authorization decision. As such, the use of OASIS' eXtensible Access Control Markup Language (XACML) as a policy language and OAuth to provide delegated access that enables transparent authorization across all components can be considered as a possible way of providing this functionality.

7. Challenges

7.1 Licensing

There is no good approach to manage licenses from big vendors like Microsoft and Siemens etc. If approached directly the vendors will ask you to buy the full licence for as many endpoints needed which would be very costly considering that the licence is used only limited amount of time and not with the intent to support the business directly. The way around that is to use evaluation licences but usually they are available only for limited time. Another way would be to use only freeware but that would not cover the whole spectrum of cyber security related software and thus would not add to realism of the CD related exercises. There are two ways to get this issue solved:

1. Pay for the licences
2. Get a special agreement with the vendors

The DL should have for the licenced assets a licence expiry indicator. The expiry date should be added manually but the license duration counter and expiry warning should be automatic.

ASSETs will be upgraded if it is necessary for the use case otherwise the assets will be kept in their original state.

For example the legacy systems should be stored in full (VM, not just description) in order to avoid looking for out of date versions of OS and applications. It is necessary for the sake of realism to keep all possible versions of OS and software available as in some exercise scenarios will have legacy systems and software involved. For security reasons it is reasonable to maintain accredited copies of all the software used in the exercises. The accreditation process needs to evaluate the origin and the availability of the software.

7.2 Asset Identifiers and Version Control

Assets in a DL will need to be assigned unique identifiers to facilitate their lifecycle management and referencing from external applications (e.g. AMS). This is a simple requirement to meet in the presence of immutable assets in a single DL. However, the problem becomes complex in the presence of several DLs which export and import assets and when the assets are modified over time by the users.

For example, the scenario in which an organization operates two DLs in two different, disconnected security domains introduces the issue of ID assignment. When an asset is exported from one domain and imported in the other domain, does the receiving domain assign a new asset ID or does it preserve the ID assigned by the DL in which it was created? It seems that if the asset being referenced is unique and will be present in exactly the same form in both domains, then the original ID should be preserved in order to ensure users know that this is the same asset. This implies that the import function of a DL needs

to offer the choice of creating an asset as new or as existing, and in the latter case, allow the import of the original ID (in addition to the asset itself and its metadata.)

This scenario also introduces the risk of ID collision if the IDs generated by each DL are not appropriately designed. To address this, various options are possible. A central ID assignment system could be used. However, given that DLs are expected to be deployed on air-gapped networks, this solution is likely impractical. Another option is to use Globally Unique IDentifiers (GUIDs) such as those specified in RFC 4122. A proper implementation of this RFC can ensure DLs can independently generate asset IDs with an acceptably low risk of collision. Finally, the Asset ID can be structured as containing a component that indicates the DL that generated the other part of the ID which uniquely identifies the asset.

The issue is further complicated when assets are modified. As the DL keeps all assets under version control, there is a requirement to maintain knowledge of the relationships between the various versions of an asset and present this to the users. It is possible to embed the version number in the Asset ID much like it is possible to identify the original DL. Such an ID would take the form [DL ID] + [Asset ID] + [Version ID]. However, with this approach another issue arises when an asset is modified simultaneously in two different DLs.

A few additional rare but plausible scenarios further add complexity to the situation. While a more detailed analysis is required to make a final recommendation, this study determined that probably the best solution address all of the related issues stemming from the full range of deployment options envisaged in the long run would be to use a two-part ID composed of a GUID for the DL in which the asset was originally created and a GUID for the asset itself, as well as to maintain backward and forward knowledge of versions (i.e. keep the the parent GUID and the children GUIDs in an asset's metadata). The DL should also provide the ability to export, import and visualize the full version history (metadata optionally with the asset itself) so that the full picture of an asset's transition over time can be rebuilt using the information distributed across all of the DLs in which versions of the asset were created.

8. DL Deployment Models

There are a few options possible for the deployment of the DL. For the NCR project, the deployment model is quite simple: a single DL deployed and operated by the Estonian Ministry of Defence along with all other components. In the longer term however, it is possible to allow other entities to deploy and operate additional DLs (and other components as well), and to deploy DLs in different air-gapped security domains. This section describes the different deployment models possible and discusses related issues. A key aspect to consider is the component's operational authority. In the longer term, it is envisaged that different organizations could deploy and operate different DLs to serve different communities.

8.1 Single DL

The single DL deployment model is the simplest and most basic deployment and is expected to be the first model to be implemented in the context of NCR. In this model, a single instance of the DL is operated by the Estonian Ministry of Defence and provides services to all of EDF-CR or NCR clients. Establishing trust between all components is a relatively simple matter, and the management of user accounts can be done by the Estonian MoD or, as previously discussed, through an appropriate decentralized authentication approach.

8.2 Multiple DLs

The multiple DLs deployment model consists of several DLs operated by different entities. In this model, users can choose to use one or more of the DLs. This model could be used to address security and/or intellectual property issues. For example, NATO or a NATO Nation may choose to operate a DL to store its own software to be used in exercises.

In this deployment model, the various components of the full NCR capability must be able to operate with more than one DL. For example, in the AMS the users would be able to select assets from anyone of the available DLs. Similarly, assets could be deployed from these DLs onto one or more cyber range. Finally, different instances of AMS operated by different entities on different cyber ranges could use a DL. The multiple DLs model introduces the need for trust to cross organizational boundaries and will require organizations to exchange certificates and/or secrets in order to establish this trust.

Finally, in the presence of several DLs, users may export and import assets across several DLs and the recommended practice for the various use cases must be clearly defined in the training program. When several DLs exist, there could be value in minimizing duplication. It may be possible for DLs to exchange the metadata about the assets they hold and to check for duplicates. If found, some of these could be referenced in other DLs rather than stored in every DL. When selected by a user, the DLs would communicate to transfer the asset as needed. Through the definition of user groups known to all DLs, this process could be largely automated.

8.3 DLs Deployed in Multiple Domains

As outlined in the TBCE, the NCR capability foresees the implementation of a cyber range in the classified domain in the second phase of the investment. While this capability will not be delivered by the procurement activity supported by this study, the associated high-level requirements are nevertheless considered to ensure its findings are future-proof. Specifically, this study considers that an activity may span several security domains that are not directly interconnected, each with its own DL. For such activities, it is assumed that the planning will take place in the AMSs in each domain, with the overall activity broken down into components according to their confidentiality requirements. It is to be expected that the assets to be deployed In the environment implemented on the classified domain will likely include a large number of unclassified assets held in the unclassified DL. This situation is portrayed in Figure 5 below, which additionally shows the existence of a DL on the Internet (itself a different security domain). A description of the dataflows in this diagram is provided in Table 2. Figure 5 also assumes that the AMS plays a unique, central role in the management of cyber range activities, as is understood from the TBCE. This implies that the other components of the CR (e.g. Situational Awareness Capability, Data Collection Capability, Network Traffic Simulation Capability, etc.) will interact with the DL through the AMS. On Figure 5, they are therefore considered part of the cyber range boxes.

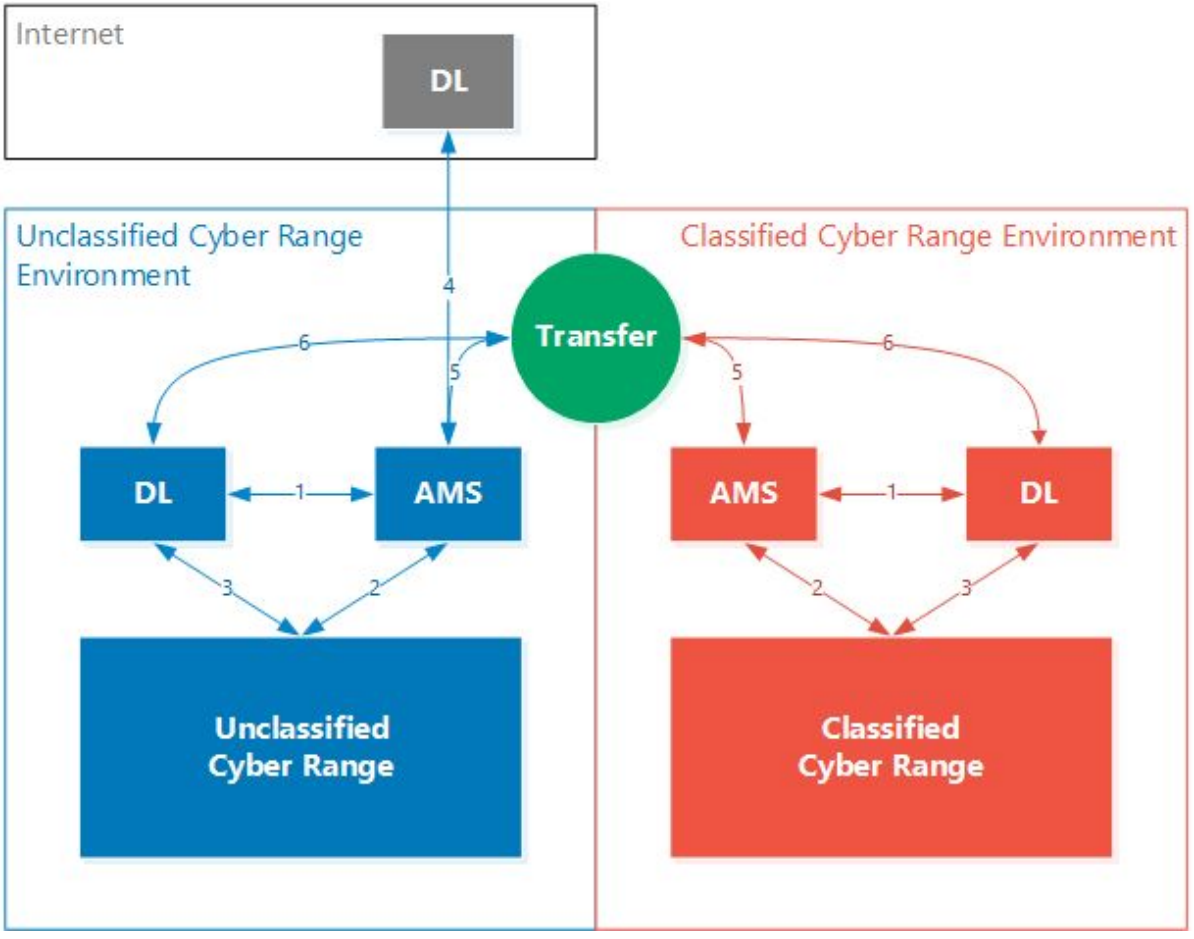

Figure 5 - Multi-Domain Deployment

Dataflow	Description
1	AMS users are able to access the DL and manage assets stored in it from the AMS.
2	AMS users are able to arrange for the deployment of assets onto the cyber range
3	On instructions provided via the AMS, assets in the DL can be directly transferred to the cyber range.
4	AMS users are able to access other DLs, possibly provided by different organizations
5	AMS users need to be able to push activity details, including how DL assets are to be used in an activity, to a different security domain. How these transfers are to be implemented and whether they are authorized in both directions is subject to security accreditation by the relevant security authorities. Such transfers might be difficult or even impossible for some organizations.
6	Subject to security accreditation, it must be possible to transfer assets across DLs in different security domains. It may be that transfers are only approved from low to high by some organizations, or that transfers in either direction are implemented in different ways (e.g. diode from low-to-high, manual transfer via security officer from high-to-low).

Table 2: Dataflows in a Multi-Domain Deployment

This use case introduces the question of whether the content of the unclassified DLs should be replicated in the classified DL. Two factors impact the decision on which approach to take. The first is the total cost of storage in the classified domain, and the second is the ease of transferring files from the unclassified domain to the classified one. If storage is cheap and transfer is difficult, the best approach is to replicate the assets across both domains, and implement a mechanism to regularly transfer the new additions and changes made in the unclassified DL across to the classified DL. If storage is expensive but there exist an easy transfer mechanism, it is best to only transfer the metadata about assets between the domains, and pull the assets across as they are needed.

The other issue that is introduced by a deployment of DLs across air-gapped domains is the handling of changes of assets or their metadata for assets that are in more than one domain. If the changes is allowed to be made in the classified domain, the updated asset would need to be transferred to the unclassified domain. In many cases, the transfer of unclassified data from the classified to the unclassified domains is much harder than the

opposite. If this is the case, it may be beneficial to prevent users from modifying unclassified assets in the classified domain, instead forcing them to make the changes in the unclassified domain and having the updated asset transferred again. Either approaches has associated pros and cons that are inherent to the operation of multiple security domains.

Regardless of where the changes can be made, a synchronization mechanism will be needed. This synchronization mechanism will need to work across air-gapped networks, and will possibly need to gracefully handle the case in which the data being synchronized has been modified on both sides simultaneously. This synchronization mechanism would ideally be triggered when any update is made on the data in a DL. This problem can be greatly simplified if knowledge about the existence of the classified DLs is itself deemed unclassified and can be shared with the unclassified DLs. It can be further simplified if the results of a synchronization are also deemed unclassified and thus can be shared back with the unclassified DLs.

In the absence of specific information about the above aspects, this study cannot make recommendations but only outline possible approaches.

8.4 Federated Cyber Ranges

The TBCE also foresees a third phase for the implementation of the NCR capability, namely covering the federation of cyber ranges operated by NATO and NATO Nations. The federation of cyber ranges allows users to organize much large activities such as large multinational exercises and geographically dispersed verification tests, for example. It also facilitates the harmonization of security policies, the management of access rights, and provides a framework for sharing a greater set of assets and cyber range resources.

The implementation of a federation of cyber ranges does not necessarily require significant additional development. Given that it mostly has to do with harmonizing policies and distributing authentication and access control, if the proper technologies and approaches are selected at the outset, a federation can be easily created, particularly if the participants reuse the same software. It is therefore critical that the technologies used to implement the DL’s authentication and access control functions be selected with the needs of future federation in mind.

Finally, it may be that different virtualization technologies will be used in the implementation of the different cyber ranges within the federation. When it comes to virtual machine images, these differ from one virtualization technology to the next. It may be beneficial to allow an asset to exist in different forms while still having the same metadata. This could possibly be implemented via the version control functionality by supporting different branches for each of the supported virtualization technology. With this approach, a DL would become a truly interoperable component of a federation of cyber ranges.

9. Recommendations

Based on the interviews conducted and the analysis made of the elements and issues presented above, this study makes a number of recommendations regarding the DL, mainly in terms of deployment and interrelationships with other components of the NCR, and regarding the various requirements a DL should meet. The recommended requirements are grouped as follows: functional, security, and performance.

9.1 The DL in its Ecosystem

The DL should be seen as an independent component of the NCR which interacts with all other components through defined interfaces. This will allow the ability to deploy several DLs by different entities and in different security domains, thus making it future-proof in terms of the second and third phases of the NCR capability. In terms of overall system integration, the definition of these interfaces (DL to AMS and DL to CR) should be jointly maintained by the developers of the DL and these other components, and should be able to evolve somewhat independently from the components themselves. In particular, it is necessary for the DL to be able to interoperate with different versions of these components as not all of them will be upgraded simultaneously. A DL will therefore need to support several versions of the interface specifications at the same time.

In addition to direct machine-to-machine interfaces with AMS and the NCR, the DL could also have interfaces with other components, such as the user and traffic simulation components for example. For these other components, two approaches are possible: direct interaction through an agreed interface, or the use of generic storage space on the cyber range to transfer assets between the DL and these components. An example of a direct interaction would be when the operator of the traffic simulation system retrieves traffic traces from the DL through direct operation of the traffic generator. Specifying and implementing a new interface to meet this kind of requirement should be relatively easy given the simplicity of the functionality (mainly store and retrieve) and the possibility to reuse API functions developed for the other interfaces. However an “interface” is more than just a specification implemented by both components, it is also a documented practice supported by training for users of both components. In the end, the decision of which approach to take will largely depend on the functionality offered by these other components.

Finally, the use of the DL may not evolve to its fullest extent by itself. It takes extra effort to make an asset reusable, and users my not initially see the value in this effort, at least not before a critical mass is reached. In addition to deploying the DL, it may be necessary to invest into training and promotion programs for the use of the DL in order to help reach this critical mass sooner.

9.2 General Requirements

1. The DL shall be integrated into the overall NCR.

2. The DL software shall be provided with the full documentation necessary for its long-term maintenance, a version control system, a testing environment, and a corpus of unit and system tests that can be used to validate future code updates.
3. The DL shall have clearly documented interfaces with other NCR components maintained under version control.
4. The DL shall be provided with manual for both users and system administrators.
5. The DL shall be provided with a “train the trainer” program including all necessary training material, covering in different modules system development, system installation, system administration, basic user training, and advanced user training.
6. The DL shall not rely on software that requires a paid license (whether perpetual or renewable) without prior approval from the Estonian MoD.
7. All supporting documents such as but not limited to design documents, help files, user manuals, and training manuals shall be provided in their native, easily editable formats.
8. The DL shall run at least on the following operating systems:
    a. Red Hat
    b. CentOS
    c. Arch Linux
9. The DL shall be fully installable through an automated installer.
10. The DL shall provide a command-line interface that exposes its full functionality and that can be used from a script.
11. The DL shall make maximum use of the Resource Description Framework (RDF) and semantic web technologies.
12. The DL documentation and training material shall describe the best practice for the usage of the DL including explaining how to develop reusable assets, how to break apart assets into more reusable components, how to develop scripts so that they are more easily reused, etc…
13. The DL shall provide storage with at least two input/output operations per second, latency, and throughput performance category.
14. The DL shall allow asset owners and POCs to selected which storage performance category to use on a per asset basis.
15. The DL shall provide the means to automatically move assets which are not pinned by their owner or POC to a specific storage performance category to a different one based on an analysis of recent and expected asset usage.

9.3 Functional Requirements

Graphical User Interface (GUI)

1. The DL shall provide a web GUI based on HTML5 and Javascript. Other technologies can be used with prior approval from the Estonian MoD.
2. The DL shall provide its web GUI via HTTPS.
3. The DL web GUI shall provide a system administration module that exposes all configuration options and through which the system administrator can manage the DL.

4. The DL web GUI shall provide a general user module that allows users to fully exploit its full functionality.
5. The DL shall provide a warning to the system administrator for any configuration change or operation that can affect operations (such as making the system unavailable, disconnecting users, etc.) when made via the GUI, as well as provide the administrator the ability to cancel the change.
6. The DL configuration GUI shall provide the ability to make several changes to the configuration and to apply them all together, where relevant.
7. The DL shall provide helpful advice for configuration changes that may affect performance or that should be done in a particular order.
8. The DL shall provide separate help functionality for the users and for the system administrators, providing both in-context help, as well as a user manual and a system administration manual delivered via the web GUI, with a table of content that is intuitive and easily navigable.
9. The help functionality in the web GUI shall provide an adequate search interface.
10. The DL shall provide a community-based wiki that can be used by users to provide overview information about related assets. The DL shall make it easy to reference and embed asset metadata in the wiki.

System Management

1. The DL's configuration shall be stored in files that can be edited directly by a user using a basic text editor.
2. The DL shall reload its configuration from files and rotate its log files when it is sent the SIGHUP signal.
3. The DL shall shut down in an orderly fashion upon receiving the SIGTERM signal.
4. The DL shall collect operations performance data (metrics) and provide tables, charts and graphs via its GUI of various aspects of its performance over various periods of time in order to allow the administrator to identify performance issues and report on system usage.
5. The performance metrics shall cover storage usage as per the asset classification system, user activity including response times for regular operations, and throughputs achieved during data transfers, amongst other relevant metrics.
6. The DL shall provide the means to export various user-specified performance reports, in both PDF and comma-delimited format.
7. The DL shall provide an alert notification functionality for resource usage and any other type of error or failure, integrated with the syslog auditing system.
8. The DL shall provide an automated mechanism to apply patches, handling error conditions appropriately.
9. Upon update, the DL shall run a verification and consistency check through all of the stored assets to ensure assets are fully compatible with the new version. It shall be possible to select whether to run this check during operations, or to block all user access while it is run. If the former, this check shall run with reasonable impact to DL performance.

Asset Management

1. The DL shall allow for the management of the full lifecycle of assets, including maintaining information on its owner and POC and status (e.g. draft, ready, error, obsolete).
2. The DL shall include license information in a an asset's metadata.
3. The DL shall allow for the storage and retrieval of any type of digital asset.
4. The DL shall provide a version control system for assets that allows user to commit changes to assets, capturing a descriptive comment by the user.
5. The DL version control system shall allow for a set of changes to be submitted atomically (i.e. to update both an asset and its metadata in a single operation, or to update several assets at the same time).
6. The DL shall provide globally unique identifiers that can be externally referenced for each version of an asset.
7. The DL shall maintain information about adjacent asset versions (the parent of a asset as well as its children) for each asset in its metadata.
8. The DL shall provide a full-featured, customizable asset classification system that includes the following functional components:
    a. Top-level taxonomy
    b. Controlled tagging
    c. Freeform tagging
    d. Ontology-based metadata
9. The DL shall provide the means to configure the asset classification system to make the various functional components and their various aspects either optional or mandatory.
10. The DL asset classification system shall automate metadata capture as much as possible.
11. The DL shall provide the means to put a NATO security classification to each asset and separately to its metadata.
12. The DL shall provide the means for users to generate a set of wiki pages for each asset, as well as to associate attachments with the asset. These elements form part of the asset's metadata.
13. The DL asset classification system shall include capture of asset usage information and shall provide an API through which the AMS and NCR can report upon details of asset usage.
14. The DL shall generate a configurable indexing system so that the text content of assets can be indexed. The DL shall make it possible to turn indexing on or off on a per asset type basis (top-level taxonomy).
15. The DL shall provide the means to search for assets using the asset classification system (leveraging the asset indexing), covering all functional components and allowing for boolean search logic as well as pattern search (e.g. regular expressions).
16. The DL shall provide an administrator with the means to manage the classification system and facilitate the implementation of changes that affect existing assets stored in the DL.

17. The DL shall provide the means to group assets together into a single aggregate asset.
18. When handling an aggregate asset, the DL shall provide the means to address the aggregate asset or the assets in it individually.
19. The DL shall provide functionality to recover assets deleted by users (e.g. “recycle bin”) for a configurable period of time.
20. The DL shall manage the version control history of an asset over time, including using the version history imported from other DLs, presenting it in the web GUI over a timeline, as a human-readable text, and as exportable JSON data.
21. The DL shall allow users to store assets in an encrypted format.
22. The DL shall provide automated functionality to detect duplicates and similar assets to their respective POCs and the administrator.
23. The DL shall provide the means to import and export assets including their metadata and version history
24. The DL shall provide the means to automatically deploy assets on the NCR, integrating seamlessly with the NCR’s system to provide the maximum automation possible.
25. The DL shall collect and manage metrics related to asset usage, such as details of access to asset and asset metadata (search hits, viewing, export, deployment, etc) including parameters about these events (user, time, connection mechanism and details, etc.)
26. In collaboration with the NCR, the DL shall provide an auto-update support mechanism by which assets are deployed according to individually configurable frequency to the NCR into a space connected to the Internet so that auto-update features of software will execute by themselves. The updated asset shall be stored back as a new version, and the DL shall document the process as well as whether changes were indeed made in the version description.

Interoperability

1. It shall be possible to deploy a DL independently of all other components of the NCR.
2. The DL software shall be updatable independently of other NCR components.
3. Every version of DL software shall maintain compatibility with all other DL software versions and interface specifications released within the last three years.
4. The DL shall be able to establish connections to any number of other components (e.g. several AMS, several cyber ranges, etc…)
5. The DL shall provide a machine-to-machine, RESTful Application Programming Interface (API) based on JSON and whose specifications are well documented and maintained under version control, passing the API version number in every API call. This API shall expose the full functionality of the DL.
6. When a call is made on its API using a version that it does not support, the DL shall gracefully negotiate the closest available version, and if none are available, gracefully terminate the request.
7. The DL shall provide functionality to maintain knowledge of other NCR components (other DLs, AMSs, cyber ranges, etc…), including user-friendly name, software and

API version numbers, URI and PoC information at minimum, whether or not they are directly reachable.

8. The DL shall implement an automated pairing mechanism with other NGCR components through which basic information is exchanged between the two components. This pairing shall be possible through direct a HTTPS connection with mutual authentication and via a data import and export function that supports pairing of components that are not directly connected.
9. The DL shall re-initiate pairing whenever its status changes (e.g. new URI, new POC, updated software or configuration, etc.)
10. When importing an asset from another DL with which it is not yet paired, a DL shall initiate pairing. If pairing cannot be done, the DL shall prompt the user for minimum information about that DL and maintain that information until pairing is successful.

Logging

1. The DL shall generate event logs that can be integrated into an rsyslog system using its functionality to the fullest while maintaining wide compatibility with the syslog standard (RFC 5424).
2. The DL shall present different configuration levels for audit logging including none, minimal, normal, high, and debug. At high and debug audit logging levels, performance requirements can be reasonably negatively impacted (i.e. not be met).
3. The DL shall use a configurable value for the facility number to use in its messages.
4. The DL shall precede its messages with “DL-ABCD” where ABCD is a configurable identification string.
5. The DL shall make appropriate use of the severity levels.

9.4 Security Requirements

1. The DL shall provide the means to mutually authenticate users and other NCR components.
2. The DL authentication service shall be modular and replaceable with a different one in the future.
3. The DL shall authenticate all interactions, whether through the web GUI, the API or the command line.
4. If using a PKI, the DL shall verify the validity of certificates in a configurable manner.
5. The DL shall provide the means to manage users, groups, communities, and roles.
6. The DL shall provide the means to assign users to groups, communities, and roles, groups to roles, and groups to communities.
7. The DL shall put all functionality under an Attribute-Based Access Control (ABAC) system.
8. The DL shall provide the means to manage the security policy of the ABAC system, taking into consideration the action, the user, the group, the role, the asset, the time, and the context, as necessary and relevant for secure operation.
9. The DL shall provide access controls to assets including:
    a. Read, add, modify, delete for assets
    b. Read, add, modify, delete for asset metadata

    c. Read, add, modify, delete for asset version history (both asset and metadata)
10. The DL shall make it possible to authorize these actions for users, groups and roles using the ABAC system.
11. By default, the DL shall allow the owner and POC of an asset to set permissions.
12. The DL shall provide a summary report of all permissions for all assets to the DL administrator, including minimal metadata, in order for the administrator to appreciate how the access control settings limit sharing.
13. The DL shall provide an integrity verification function for all assets that is integrated with the matching functionality provided in the NCR and the AMS so as to provide end-to-end integrity verification.
14. The DL shall provide malware scanning for all assets at time of creation and at a configurable frequency.
15. The scanning for malware shall not overly affect performance.
16. When malware is detected, the DL shall put the asset in quarantine and notify the administrator and POC of the asset.
17. The DL shall not allow the deployment of an asset that is in quarantine.
18. The DL shall allow the export of a quarantined asset upon specific administrator approval.
19. The DL shall allow the update of a quarantined asset so that a cleaned or inert version can be put in place. The version history shall include information about the quarantined version without making it accessible, except as per above.
20. The DL shall provide a configurable backup mechanism for both online and offline backups. The backup mechanism shall support incremental backups.
21. The DL shall provide the means to run a test restore of the backup at minimum annually and facilitate the verification that the restore is actually successful.
22. The DL shall be implemented in a way that supports high-availability through redundancy.

9.5 Performance Requirements

1. The DL shall be able to store at least 2^32 objects.
2. The DL shall allow for the storage of individual assets of up to 16 exbibytes in size.
3. The DL shall demonstrate a sustained ingest throughput of at least 80 MB/s when receiving a 2 GB file from a client on a Gigabit Ethernet connection.
4. The DL shall demonstrate a sustained delivery throughput of at least 80 MB/s when sending a 2 GB file to a client on a Gigabit Ethernet connection.
5. The DL shall be able to handle at minimum 256 concurrent user connections through its web GUI and command line interface and simultaneously handle 512 concurrent API connections with other CR components.
6. The DL shall provide responsive functionality at all time, informing users and other components of the estimated time needed for long-running operations (greater than 3 seconds) allowing them to cancel such operations.

References

1. Estonian Ministry of Defence. (2015). “EDF CYBER RANGE 2015- 2021”
2. Estonian Ministry of Defence. (2016). Study on “NATO NU CYBER RANGE TECHNICAL DESIGN AND NATO NS CYBER RANGE ARCHITECTURAL SOLUTION”
3. Estonian Ministry of Defence. (2017a) NATO Type B Cost Estimate for “CAPABILITY PACKAGE 9C0121 NATO CYBER RANGE”
4. Estonian Ministry of Defence. (2017b). Tehniline Kirjeldus. Lisa 1 Hankedokumentide „Küberharjutusvälja õppeteegi uuring“ juurde
5. Spafford, G. (2010). How to Set Up and Manage a Definitive Media Library. Accessed 27.12.2017. Available from: http://www.itsmwatch.com/itil/article.php/3887361/How-to-Set-Up-and-Manage-a-Definitive-Media-Library.htm
6. Fedora Commons robust, modular, open source repository system. (2017) Visited websites http://fedorarepository.org/ and https://wiki.duraspace.org/display/FF/Fedora+Repository+Home
7. Bytelife OÜ. (2014) vLM user manual v1.2

Annex A. Interview questionnaire

1. Why have the components not been reused so far?

2. Which use cases enable to reuse components developed on cyber range?

3. What types of artifacts would you want to see in the library? Please describe their properties and possible use cases.

4. Which are the requirements in order to reuse the components?

5. Which parts of the component usage can be automated?

6. Which other requirements need to be addressed in order to reuse the components?