UNIVERSITY OF AMSTERDAM

SYSTEM AND NETWORK ENGINEERING

REASEARCH PROJECT 1

Automated Windows lab Deployment

A method of deploying, installing and configuring a windows lab environment

Vincent van Dongen
Fons Mijnen

Supervisors
Marc Smeets
Mark Bergman

February 12, 2017

Abstract

Realistic test environments are essential for (security)testers and researchers. Therefore, it is necessary to quickly deploy a realistic testlab. Many products and techniques exist to automatically deploy Windows systems. However, these automated deployment tools only deploy basic Windows systems and do not create a realistic test environments, such as:

- Active Directory Domain Controllers.
- Site links.
- Replication Schemes.
- Domains and sub domains.
- Mail- ,DNS- ,Web- ,DHCP-, File-server.
- User database with users, groups, Organization Units.

Also, these tools do not generate or create traces of system and user behavior, such as:

- Groups and user account located in the user-database.
- Files located in user folders.
- Mailboxes with email included.
- Client applications.
- Log and event files.

This research investigates whether or not it is possible to automate the deployment of a realistic test lab. By using the currently available techniques and tools such as configuration management tools, image deployment tools, virtual machine snapshot techniques, and cloud templates it is only partially possible to automatically deploy a realistic testlab. Therefore, we have created a new model for lab deployment and tested the model in an experiment. The prototype uses a combination of different kinds of techniques and methods and is mainly programmed using Powershell scripts. The results of the experiment show that the proposed model is more efficient then current techniques for automated lab deployment. The prototype is able to automatically deploy a realistic testlab. The prototype can also create traces of user interaction. However, some user behavior can not be simulated.

Acknowledgements

We would like to express our gratitude and appreciation towards our supervisors Marc Smeets and Mark Bergman. Their assistance and guidance during the whole research period have been really valuable to us.

Contents

1 Introduction

It professionals, students, and researchers use test labs for a variety of reasons. Many products and techniques exist to automatically deploy Windows systems. However, these automated deployment tools only deploy Windows systems and dont create a realistic test environment. Therefore, manual configuration is required to create a useful testlab which can require a lot of time. Also, configuration errors can occur and technical knowledge is required to build a test lab. Moreover, these tools require a lot of user input, which extends the deployment time. A testlab does not have these kinds of limitations when it has already been automatically installed and configured.

Furthermore, the currently available products and techniques that exist to automatically deploy Windows servers do not generate or create traces of system/user behavior. For example, deployment tools do not generate log files, event files, browser history and connection logs. These kinds of user and system traces can be very useful for security researchers and IT professionals. They can test new vulnerabilities, measuring the impact of security breaches or run samples of malware in a testlab. A realistic Windows testlab can be defined by the following elements:

- Groups and user account located in the user-database.
- Files located in user folders.
- Mailboxes with email included.
- Client applications.
- Log and event files.

This research investigates whether or not it is possible to automate the deployment of a realistic Windows test lab. For this research project, we have proposed the following question:

Is it possible to automate a fast and easy rollout of a realistic Windows test environment with minimal user interaction?

To answer this research question, we have defined the following sub-questions:

1. What kind of techniques and methods exists to deploy and configure a Windows testlab
2. What is the most suitable option to automate the deployment and configuration of a Windows testlab?
3. What kind of techniques and methods exists to simulate system and user behavior on Windows machines?
4. What is the most suitable option to automate the simulation of system and user behavior on Windows machines?

This research is divided into multiple phases. First, the current techniques to deploy and configure a testlab are investigated. During the investigation, we also focus on the limitations of these products. Secondly, the current techniques for user simulation are investigated. Next, we propose a new model for Windows testlab deployment. In order to prove the model we will build a prototype to create an automated realistic testlab. We will then conducted experiments to verify our prototype. After that, we compared the existing methods and techniques with the proposed model and analyzed the results. Finally, we discussed the results and answered the research questions.

1.1 Related work

Very little existing research is done into this field. Although many commercial and open-source products exist to deploy an infrastructure, as well as products that aim to manage configurations in infrastructures, almost no research has been done to combine these products. Therefore, we use the research reports that describes different techniques to deploy a visualized environment. [11] Also, we studied existing products with the purpose of automated deployment of virtual machines. Existing commercial products like Puppet, SCCM, Vmware Deployment, and Azure have been studied and reused in this research.

Almost no research has been done related to building a test environment with traces of use like log files, events, and connections. Only a few unofficial blogs, tutorials, and web pages describe a few techniques to create traces of use.

1.2 Defining a testlab

Before the experiment can start, it is necessary to define what exactly a Windows testlab consists of. It depends on the purpose of the testlab which and how many servers should be installed. This testlab is designed for security testers. Below we have defined a list of what we think the testlab should include:

- Active Directory for a user-database.
- Multiple Active Directory Domain Controllers:
  - Three different domains.
  - Each domain contains 1 Domain Controller.
  - All the three domains are located in the same Forest.
  - All the Domain Controllers have a two-way trust and replication scheme.
- Email Server.
- Domain Name Server.
- Web Server.
- Client computer systems.
- DHCP server for releasing dynamic IP-addresses for clients.
- SMB share for sharing files and personal folders of users.
- Internet access for clients.
- Traces of user and system behavior:
  - Groups and user account located in the user-database.
  - Files located in user folders.
  - Mailboxes with email included.
  - Client applications.
  - Log and event files.

Below are the functional requirements displayed:

- Relatively fast deployment (less than 12 hours).
- Minimal user interaction.
- Functionality to automatically update Windows servers and client Operating Systems.
- Definable parameters such as domain names, IP-addresses and users/groups.
- The total costs should be as low as possible.
- The total amount of disk space should be as low as possible.

2 Currently existing techniques and product

In this chapter, the existing techniques and product related to this research will be discussed. First, the current techniques to deploy and configure a testlab will be discussed and, secondly, the current techniques to simulate user and system behavior will be discussed.

2.1 Current techniques to deploy and configure a testlab

A variety of products, methods, and techniques already exists to deploy and configure a testlab, such as Puppet, SCCM and WDS. All the different products, methods and techniques can be roughly divided into four groups:

1. Configuration Management
2. Image Deployment
3. Virtual Machine Snapshot
4. Cloud Templates

Configuration management

Configuration management refers to a discipline for evaluating, coordinating, approving or disapproving, and implementing changes in computer systems that are used to construct and maintain software systems. Configuration management tools like Puppet ensures that the right configuration is deployed across many computer systems. The system resources, system configurations and their state are described in a system-specific catalog, which is applied to the target computer system. Configuration management tools like Puppet and Ansible are designed for deploying Configuration files to computer systems. However, some configuration tools are able to deploy computer systems. Configuration management tools already require an existing environment. This means that another infrastructure has to be created in order to use the configuration management tools. [5]

Image deployment tools

Image deployment tools are used to quickly and effectively install operating systems to servers and clients. In order to install an operating system, an installation image is required, which contains the Operating System. Also, a boot image is required to boot a system to perform an operating system installation. A boot image contains the Windows Deployment Services (WDS) client and the Windows Preinstallation Environment (Windows PE), which is basically a mini operating system used to connect the system to the WDS server and provide the means to select and install a WDS installation image. There are several methods to deploy an operating system [7]:

- PXE-boot: In this method of deployment, the operating system image and a Windows boot image are sent to a distribution point that is configured to accept PXE boot requests.
- Multicast deployments conserve network bandwidth by concurrently sending data to multiple clients instead of sending a copy of the data to each client over a separate connection.

- Pre-staged media deployments let you deploy an operating system to a computer that is not fully provisioned. The pre-staged media is a Windows Imaging Format (WIF) file that can be installed on a bare-metal computer by the manufacturer or at an enterprise staging center that is not connected to the Configuration Manager environment [9].

The most suitable solution would be the PXE-boot because network connectivity is not an issue for a testlab and the computer system are fully provisioned, which means that pre-staged media deployments are not necessarily.

Image deployment tools are designed for deploying Operating systems. But these tools are not able to deploy configuration files. Its, however, possible to create an installation image which has already installed software and applications. These images are called Capture Images. Before a captured image is used, a system with an operating system is customized by adding applications, custom configurations, and other system changes that are required for the testlab. When the system is ready for imaging, additional steps must be taken to ensure a successful capture, such as clearing out application and specific Registry keys. This means that manual configuration is still required afterward.

Virtual machine snapshot

Another technique is called virtual machine snapshots. A virtual machine snapshot is a file-based representation of the state of a virtual machine at a given time. All changes made after the snapshot was taken may be based on that snapshot information (incremental changes). When a virtual machine snapshot is created, the hypervisor pauses the virtual machine and creates a special disk image. From that moment, the hypervisor will write changes only to the extra disk image. Therefore, the base disk image file is not modified. The hypervisor still uses the original disk to read files. When a snapshot is reverted back to the moment when the snapshot was created, the hypervisor only removes the disk image files that contains all the changes. This method could be very useful in order to build a testlab. A testlab only has to be installed and configured manually once. At that point, a snapshot will be created that will be the starting point for any testlab. When someone used the testlab and has made changes in within the testlab, the default state of the testlab could easily revert back by using snapshot.

There is, however, a very big downside of using snapshots. Microsoft strongly recommends to not make use of any virtual machine snapshots within Domain Controllers. Creating and then reapplying a snapshot to a virtualized Domain Controller can create a situation known as Update Sequence Number (USN) rollback. The USN is used as a sequence number to determine which domain controller has the most recent version of the database. When restoring a Domain Controller by using snapshots that were taken earlier, the USN is then also restored. After the restoration, the DC then goes about making changes again to its database, incrementing the USN. But, this time the USN is not correct. This replication problem will occur when only a few Domain Controllers are rolled back through a snapshot. [6]

Another disadvantage of using virtual machine snapshots is that there is no efficient way to update software and Operating Systems. Manually updating every virtual machine is the only proper solution to update software and Operating Systems, which require a lot of time.

Templates

Finally, the last technique relies on templates. Within cloud solutions, such as Amazon AWS and Microsoft Azure, it is possible to deploy and configure a Virtual Machine based on a template. For each Virtual Machine, a unique template can be created. This means that the end-user only has to execute the templates to build an up-to-date testlab. However, these cloud solutions are only able to configure basic settings. It is for example not possible to automatically create and setup an Active Directory Forest or define the Active Directory Sites. Furthermore, the costs of virtual machines are relatively high compared to a dictated server. This is mainly due to the fact that the cloud providers charge a high amount of the total cost to keep the Virtual Machine up-and-running by providing redundancy. This is obviously not necessarily when you are running a testlab. Moreover, Microsoft automatically updates the Operating Systems. This could be a disadvantage for security testers that want to use an outdated Operating System to test security vulnerabilities. Finally, within most cloud providers it is not possible to create client VMs like Windows 7, 8 or 10. The client machines are compulsory to create a realistic testlab. [2]

In conclusion, none of the products, methods, and techniques are able to efficiently deploy and configure a testlab. Also, the product that can configure systems, are not able to configure sophisticated features such as Active directory schemes, replication connections and sites, and services. Either a combination of techniques are required or a completely different solution has to be developed.

2.2 Current techniques to simulate user and system behavior

After a testlab has been deployed and configured, some traces of user and system setup behavior has to be added to the testlab. This distinguishes a testlab from a realistic testlab. Below are a few examples displayed of traces of user and system setup behavior:

- Groups and user account located in the user-database.
- Files located in user folders.
- Mailboxes with email included.
- Client applications.
- Log and event files.

Almost every IT infrastructure has an Active Directory user-database. Many tools exist to create these user accounts. Examples of such tools are Sysmalogic, Firstattribute, and Netwrix. A few of those tools have a feature that adds parameters to accounts. An example of a parameter is whether or not an account is expired or disabled. The same tools are able to create groups and attach users to the group. These tools require an XML or CSV input that should be provided by the end-user.

Next, users often download, create or generate files such as doc(x), pdf, zip and ISO files. We have found no tools applications available that generates such files. Furthermore, there are also no tools available that generates search browser history, cookie session or stored web page credentials.

Many organizations use a mail server. Therefore, a mail server is included in the testlab. The mail server should contain emails and mailboxes in order to simulate user and system behavior. An application called Microsoft Exchange Server Load Generator is used to generate the traffic. It uses simulated exchange user mailboxes to generate load traffic. The called Microsoft Exchange Server Load Generator was designed to validate deployment settings and identify bottlenecks, but it can also be used to simulate user traffic [6].

Most clients have additional software installed on the client machine such as Microsoft Office, Skype, Mozilla Firefox, and Adobe reader. There are many existing products to deploy the additional software. The software can also be deployed through GPO‘s (Group Policy Objects).

Finally, log and event files should be added to the testlab in order to make the testlab more realistic. A logfile is a file that records either events that occur in an operating system or other software. Within Windows Operating system, its allowed to create and add logfiles. This can be easily accomplished by using scripting tools. However, its not allowed to alter the timestamp of the log. This means that the system time is always applied as a timestamp. There are currently no tools available to alter the timestamp [4].

In conclusion, many tools and software exist to create some form of traces of user and system behavior. However, these tools can only generate traces for a few applications. This means that by using currently existing tools and product, its only partially possible to create a realistic testlab.

2.3 Conclusion

We have investigated the currently available techniques and methods to deploy/configure a testlab and the currently available techniques and methods to create a realistic testlab. The different techniques and methods can roughly be divided into four groups. However, by using one these techniques and tools its only partially possible to automatically deploy a realistic testlab. Therefore, some requirements cannot be fulfilled by using currently available techniques and require manual interaction that was defined in chapter 1. This is also applied to tools and techniques that create a realistic testlab.

3 New Windows testlab deployment model

Considering the proposed models and techniques in chapter 2.1 and 2.2 a new model for lab deployment is needed in order to create an efficient lab deployment tool. This new model will be built with the intention of creating a tool that automates a fast and easy rollout of a realistic Windows test environment with minimal user interaction. This model will use a combination of existing techniques, functions in virtualization, and automation of Windows systems. The model will be defined as an abstract method of creating a tool to do lab deployment.

In order to test the new model, we will create a prototype of the model. This prototype will be used to benchmark the performance of the model. Furthermore, we are able to compare our model with other existing deployment tools and determine whether or not this model is viable and how it performs compared to other solutions.

3.1 Model specification

We have divided our model into 8 phases. We have defined these 8 phases based on the distinct techniques and functions each phase has. Each phase will have an independent input and output. Therefore, the model is capable of dynamically adding and removing phases. The phases are defined as followed:

1. Lab definition
2. Deployment of server and OS installation.
3. Server provisioning.
4. Software installation and configuration on servers.
5. Deployment of clients and OS installation.
6. Client provisioning.
7. Software installation on clients.
8. Log file and user behavior emulation.

The model will be specified to adhere to a set of rules. First of all, the model should be able to deploy a testlab as defined in chapter 1.2. Moreover, the model should also be extensible to add different operating systems if needed. Therefore, the phases that divide the testlab should be as independent as possible. For example, if router support is needed within the testlab, it should be able to define and create extra phases in the model that add an extra router.

The lab will be deployed in a virtualized environment. This is done by a hypervisor. An hypervisor is a software architecture that allows administrators to create virtualized machines. This setup has a few advantages. First of all, the hypervisor is able to run scripts that deploy and configure Virtual Machines. Secondly, the images can be transferred within the file system. Therefore, no network is required. Thirdly, the codes and commands are being send over the virtual network. This means that any packet loss is limited to a minimum.

3.1.1 Phase 1: lab definition

Before lab deployment can start the lab should first be defined. The specific language of storing the lab information shouldn't be important. It could be for example stored in XML, SQL, or CSV. The lab definition will require a name so it can be found and the lab configuration can be read by the different phases. This will reduce the amount of information required to transfer between phases and thus increase independence.

A subnet is required in order to allow connectivity between systems in the lab. In order to ensure that no IP range conflicts arise, we propose the use of a 10.0.x.0/24 network. This allows a single machine to deploy 255 labs of up to 253 machines, allowing a theoretic possibility of 64515 machines across all labs. Finally, domains and sub-domains for the Active Directory scheme configuration should be also defined in the lab definition.

In this phase, the system specifications can be defined for the virtual machine, such as Hard-drive size, RAM Memory, CPU cores, Operating system, and other machine specifications. At the very least we propose that operating systems and programs to be installed should be defined for each machine.

3.1.2 Phase 2: Deployment of server and OS installation

In order to deploy Windows images differencing disks will be used. This means that for each image to would be used to deploy a server, a *golden image* will be created. Furthermore, for every Virtual Machine, a new disk will be created and attached to one of the *golden images*. The images have a parent-child relation where all the changes and modification related to the golden disk will be written to the differential disk [8]. The architecture used by differencing disks is shown in figure 1. Differencing disks can increase manageability, due to the fact that the Virtual Machines share a similar configuration, and can dramatically reduce the amount of disk space required on the Hypervisor. It is imperative that the parent image remains read-only since every image will be based on this parent image.

Windows operating system installations include many unique elements per installation. Since the differencing disk will include these unique elements each child disk will also contain these. However, this would create conflict where unique elements would be the same across all children of a *golden image*. Therefore these identifiers need to be generalized for every child disk. The tool Sysprep can be used in order to generalize unique elements in an Operating System. Sysprep seeks to solve these issues by generating new computer names, unique SIDs, and custom driver cache databases during the Sysprep process. Sysprep is used in the final step to make a golden disk. Before the system will shut down, the tool Sysprep is run on the system and will automatically shut down

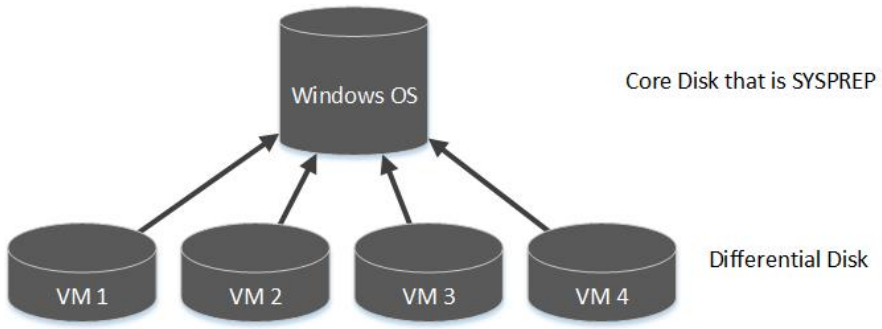

Figure 1: Overview of the differencing disks architecture

the computer system. When a differential disk is attached to the golden disk and the system has been turned on, the Sysprep tool automatically performs the final step to generalize unique elements.

By using differencing disks in addition to Sysprep *golden images* each lab machine uses minimal disk space by using (the same) base images. From here it can be populated by adding programs and other settings This method of deployment has a couple of advantages.

First and foremost it allows a high level of automation. The golden images can be automatically prepared and Sysprep for later deployment. A *unattend.xml* file can be attached during the Sysprep setup. The *unattend.xml* file is used to automate Windows installation, but can also be used to perform the final steps of Sysprep. Within the *unattend.xml* file, it is possible to define parameters for the system such as time zones, default password, and activation keys.

Another valuable advantage of this method is that it is able to maintain an automated library of Windows images with old updates. By automating Sysprep and Windows updates, it is possible to automatically update an image and Sysprep the golden image again. This updated image can then be stored in a library.

3.1.3 Phase 3: Server provisioning

After the *golden images* are deployed, the servers will be provisioned. In order to connect the server to the network the server will get a static IP address in a 10.0.x.0/24 range where 10.0.x.1 will be the IP address of the virtual switch on the hypervisor. Through this network interface, the hypervisor can access the lab environment and issue commands. Without any setup or configuration, a server will always get an IP address from the 'Automatic Private IP Addressing' (APIPA) range [2]. This IP range is automatically given to windows systems when the system can not find a DHCP server to retrieve an IP address from. The network interface on the hypervisor has both a 10.0.x.1 address and a 169.254.x.1 address on the same interface. Therefore, the hypervisor is able to contact the servers with a APIPA address and a 10.0.x.0/24 address. When the connection is established, the hypervisor sends a command to the server to change its IP address, computer name, and local admin password. The x in the IP addresses are defined in the lab configuration in Phase 1.

3.1.4 Phase 4: Software installation and configuration of servers

With the servers provisioned and updated, software can now be installed on the server. Since software can depend on other software the order of installation can be defined. The model will simply install a program from the defined list in phase 1 in sequential order. Installers like EXE and MSI files can be transferred to the machine then run with specific arguments, ISO and IMG files can be mounted to the VM and then run. The model allows for all types of installation formats to be used.

3.1.5 Phase 5: Deployment of client and OS installation

Client deployment is very similar to server deployment. It can be started at the same time as the server deployment but will have to wait with continuing to phase 6 until the Domain Controllers are configured. The clients, just like the servers are deployed from differencing disks. The considerations and advantages mentioned in phase 2 also apply to the client deployment.

3.1.6 Phase 6: Client provisioning

Clients are given IP addresses through DHCP. One of the Virtual Machines runs a DHCP service. The clients systems will get a dynamic IP address from the DHCP server. The DHCP server can be polled to retrieve the total amount of addresses leased. Once the number of leased addresses equals the number of clients on the system, the installation is done. Next, the hostname, DNS server, and local password are configured. After that, the clients reboot and are joined to a domain.

3.1.7 Phase 7: Software installation on client

Software installation on the clients is done by copying the installer files to the client and then running it with Administrator privilege. By using this technique, it is also possible to install older software. In phase 1 different versions of software can be defined for different clients. A database which holds the different versions of the software can be contacted and provide the client with the preferred version of the software. This database should also contain arguments needed to perform a quiet installation of the software.

3.1.8 Phase 8: Log file and user behavior emulation

The final phase generates log files and creates a realistic test environment. On both server and clients, events will be created that create log files. Windows log files are hard to alter since they are under constant use by the logger service and thus immutable while Windows is running by any process except by the logger service itself. Some software is available that injects malicious code into the event logger service which allows the user to alter or remove windows log events [10]. This software works only on Windows 2000, and no known software is currently available that allows the mutation of Windows log events.

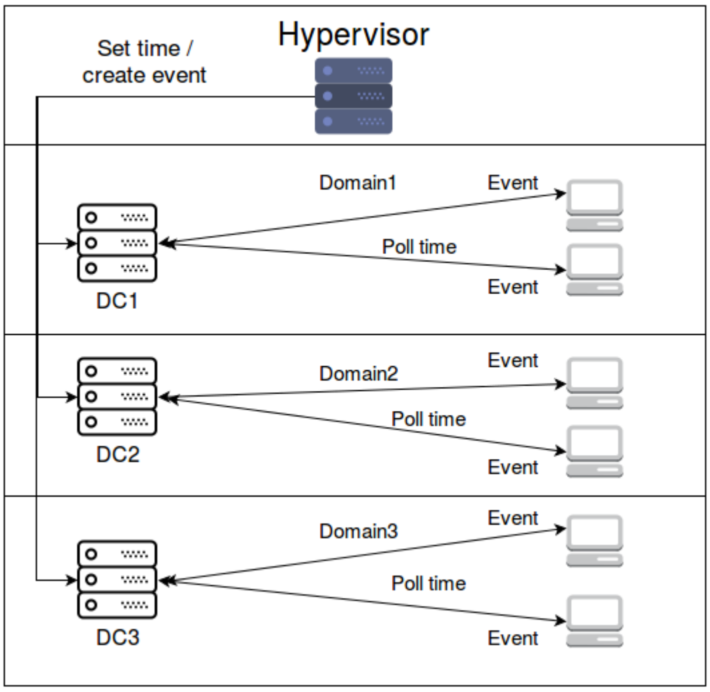

Figure 2: Overview of log emulation

In order to simulate logs of many years of use, we propose altering the time and creating events between time alterations. By using the domain controller servers as time synchronization servers the time of clients and servers can be altered and then create events after that time change. We propose the use of a separate running script on each client that polls the server every N seconds. Between these seconds, the clients can generate events that are logged. For example, client programs can be started or server services can be stopped and started. Furthermore, due to the fact that every machine is virtualized external events like power offs can be virtualized.

Using this model, the hypervisor will send events to the domain controllers where they will jump a random number of minutes in a certain range and then create events. In parallel, the client will be polling the servers for the *"current"* time and create events. This allows the model to speed up log creation and create log files for years over a couple of nights depending on the acceleration of the time. An overview of this model is detailed in figure 2

3.1.9 Final model

Using the model in the eight phases proposed above, we can roll out a test lab with minimal user interaction in an efficient way. Since this model is specifically designed for testlab deployment, it is more efficient than current deployment services and in theory should be faster and less work intensive to build. A final overview of the workflow of the phases is shown in figure 3

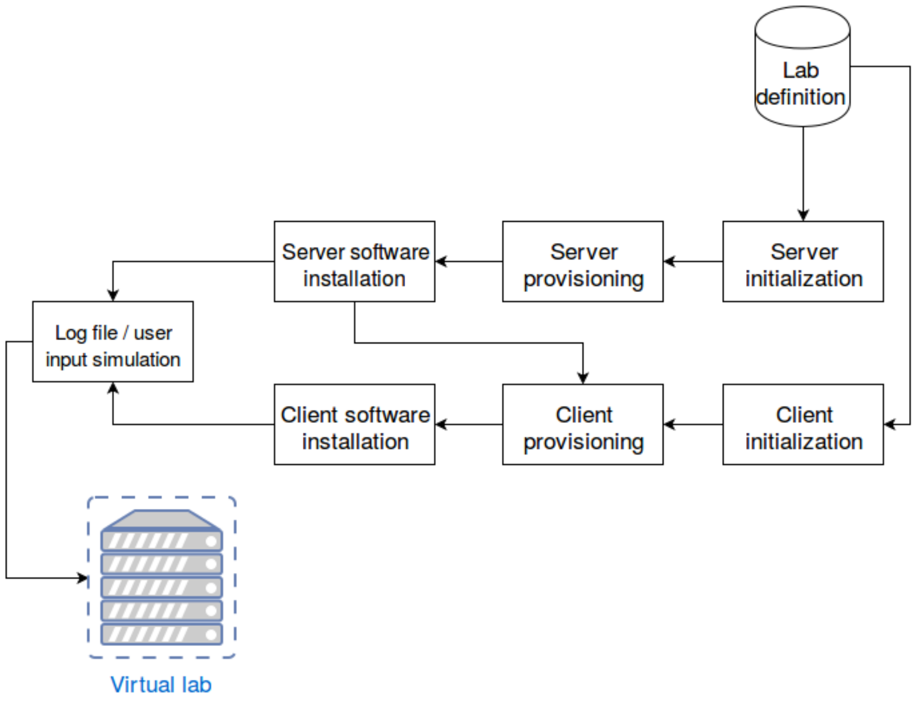

Figure 3: Overview of the 7 phases of testlab deployment

3.2 Prototyping the model

In the previous paragraph, a model was defined that deploys a *zero-touch* testlab. In order to verify the model we have created a prototype which is described in the next paragraph. The complete source code of the prototype is located in appendix A.

3.2.1 Prototype resource specification

The prototype will be build and tested on a server with the following specification:

- Intel(R) Xeon(R) CPU E3-1240L v5 @ 2.10GHz 4 cores
- 16GB RAM
- 100GB disk
- Windows server 2012R2 OS

3.2.2 Underlying architecture

The prototype is built on a Windows machine using Hyper-V as an hypervisor. Since the prototype is built on a Windows machine, we use the native windows scripting language Powershell. By using build-in Powershell functions, we can perform all the required functions of phase 2, 3, 4, and 5. Powershell is also capable of sending commands to a remote Windows machine. PowerShell Remoting allows a computer system to run individual PowerShell commands or access full PowerShell sessions on remote Windows systems. It is similar to SSH for accessing remote terminals on other operating systems. Another interesting consideration is the extensibility that Powershell offers. We want to develop a prototype that supports Windows Operating Systems, but it should also be extensible enough to support other operating systems like Linux or Mac. Powershell has been open sourced in 2016 and Microsoft is actively developing Linux and MAC support for Powershell. [12] These considerations make Powershell the obvious choice for the prototype.

Powershell also supports a feature that directly installs Windows features. all the features that have to be installed can be added to an XML file. These XML files are transferred to the server and then read and executed. Instead of installing features by using XML files, it is also possible to install features by using DVD drive. Virtual Machine techniques enable a Virtual Machine to create a virtual drive on which the DVD drive can be mounted on.

3.2.3 Prototype build

The prototype uses a variety of PowerShell scripts to do the lab deployment. Several components need to be present in order to deploy the testlab. The lab definition is stored in an XML file containing the name of the lab, IP range and (sub)domain names. Each lab machine has a list of programs, a parent disk, and a domain name. A library of images also has to be available to use the system. When the prototype is installed on a new machine, the library will have to be built manually at first. As discussed in chapter 3.1.2, the image library can be automated to a certain extent.

It is possible to automate the entire process of Windows updates. Therefore, it is possible to keep the images up to date with the newest patches. This method relies on a 3 step process shown in figure 4. First, the *golden image* is added to a Virtual Machine and started. This Virtual Machine will have a static IP and access to the outside world. When the Virtual Machine is booted up, Windows updates will be started remotely. Once done, the machine will be completely up to date and powered down. Now the VHD file is copied to a staging area where it will be added to a new VM. In this staging area it will be booted up, be given a *unattend.xml* file and then Sysprep is run. After the Sysprep is done, the image will be given a date when it was imaged and added to the library of images. Using this technique updating the images can be done automatically without needing other software or manually downloading the updates. This task can be scheduled and manually run when needed to keep images up to date.

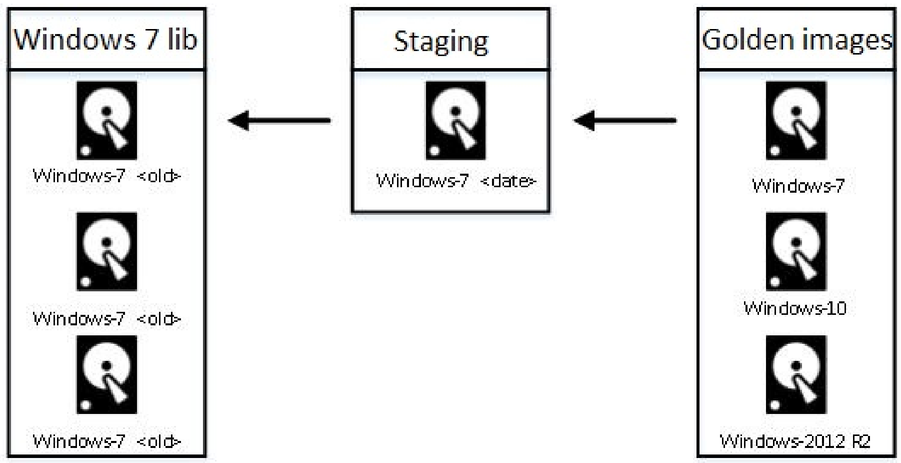

Figure 4: Automated windows updates overview with a windows 7 image

When starting the prototype, the file *create_lab.ps1* is called. This script only requires two inputs to do the entire lab deployment. A lab name, which it uses to find the corresponding lab entry in the *labs_config.xml* file and a password, which has to be provided by the user. This script will be the central script that calls other scripts in sequence. It calls the *init_servers.ps1* scripts to perform phase 1 and 2 of the deployment model. Once it finishes deployment it starts Domain Controller installation.

When the servers finish the installation of domain controllers, is starts the script *install_dc.ps1*. According to the lab specification in chapter 2.1, the testlab contains multiple Domain Controllers located in 3 different domains that exist in the same forest. Domain controllers can be installed through PowerShell. All the mandatory parameters for promoting a Domain Controller can be included in the script. However, when all the Domain Controllers are being configured, replication errors may occur. In order to solve these issues, the replication time has to be altered. Moreover, some waiting timers have to be included in the script to successfully configure a Domain Controller. Finally, its also possible to create and define Microsoft Active Directory Sites. This can also be accomplished through Powershell. [3]. After the Domain Controllers are installed, users are added based on the *users.csv* file. Meanwhile, the clients are initialized with the *init_client.ps1* script during the installation and configuration of the Domain Controllers.

In order to support some form of mail in the Testlab, Microsoft Exchange is used. Unlike all the other features mentioned above, Exchange server cant be installed with the Server Manager deployment cmdlets. Invoking the installer with arguments is also not possible because of sandboxing issues with remote Powershell calls. Therefore, we use the *schtasks.exe*, a task scheduling application, to break out of the sandbox and install Exchange. Next the script *install_features.ps1* is called. This installs standard Windows features like DNS, DHCP, and IIS through the *install-windowsfeature* cmdlet.

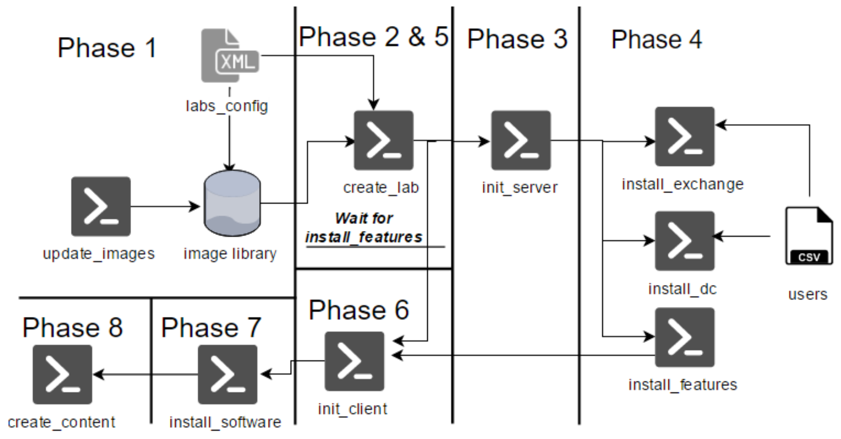

Figure 5: The prototype workflow

Now that the servers are completely installed, the clients can be provisioned. The Hypervisor polls the DHCP server on current leases and unblocks after all clients have leases. It then provisions the clients by adding domains, computer names and local admin passwords. After provisioning, the script *install_programs.ps1* is run. This fetches the programs and program versions listed in the *labs_config.xml* file. In the *programs.xml* file, the programs, versions, extensions, and other arguments that are required to silently install software are called. The installers are transferred from the Hypervisor to the client machine and then run with the corresponding arguments.

As explained earlier in this report, it is possible to create log files with PowerShell. However, it is not possible to change or alter the timestamp of log entries. When a log event is created, it uses the current system time as the timestamp. Therefore, by altering the current system time, it is possible to use a different timestamp. As explained in chapter 3.1.8 we can leverage the Domain Controllers to do the time manipulation. We offload scripts to the clients, these scripts both poll the Domain Controllers every 5 seconds and generate events for log files. From the Hypervisor, we call the Domain Controllers to jump a certain amount of minutes in lockstep and generate events on the servers.

Beside logfile generation, we use a script with a small dictionary to generate user files, folders, and content using random names and file extensions. Finally, emails with random content are sent by using Exchange command line functions.

3.3 Conclusion

In order to verify the proposed model, we have created a prototype that automates a fast and easy rollout of a realistic Windows test environment with minimal user interaction. This solution uses a different kind of techniques and methods compared to the existing ones. The prototype proves the viability of our proposed deployment model that was defined in chapter 3.1. The complete overview of resources and scripts that were used in the prototype can be seen in figure 5 The results of the experiment in the next chapter.

4 Results and Comparison

In this paragraph, we will evaluate the findings and results of the experiment. Next, we will compare the results with already existing tools, methods, and products. Finally, we discuss the results.

4.1 Findings and evaluation

It is possible to automatically deploy a realistic testlab with the techniques and methods that were described in the previous paragraph. However, not every technique or method worked as it should be. Below are the findings and results of the experiment displayed:

1. It is possible to automatically deploy a realistic testlab.
2. Powershell was designed for maintenance and not for configuration management. Therefore, not all installation or configuration parameters can be defined in Powershell.
3. Microsoft Exchange has no function for remote installation. In order to install Exchange the remote Powershell environment has to be broken out of.
4. Within Windows operating systems, user are not able to alter timestamps in logfiles and other system- or user traces.
5. The average installation time is approximately 5.5 hours for 8 virtual machines.
6. To deploy a basic testlab consists of 8 servers and 3 clients machines, the total disk usage is approximately less than 200 GB.
7. We have found no other way to force the replication between Domain Controllers lower than 15 minutes.

After conducting the experiment many times it was apparent that Microsoft Exchange server crashes often or does not completely install the Exchange application. Many error handling techniques had to be included in the deployment script in order to install the Exchange server.

The Domain Controllers can be installed without a problem. However, a sophisticated script is required to create a domain and join a forest. One of the most important part of the script that configures the domain Controllers, it that a waiting timer of 15 minutes has to be included in the script. Otherwise, replication errors will occur when an extra Domain Controller will join the Forest.

Traces of user and system behavior can be accomplished by installing additional software, creating users/groups/Organization Units/mailboxes, creating Shares, file servers, and create user file and mails. Log files can be generated by time synchronizing with domain controllers. However, it is apparent that not all user actions can be emulated within the model. Some standard user behavior like a desktop logins can not be done with scripting. Time synchronisations are also stored in the log files creating entries that should not be there. Therefore, we conclude that the creation of log files is still lacking and can not be done until a method is found that allows users to alter log events.

Finally, the required user input is minimal to deploy the testlab. Before a testlab can be created, a CSV file has to be completed with user account and groups names. All the account, organization names and groups that are defined in the CSV file will be created in the user database. After that, an XML file has to be completed with parameters for the servers. Finally, the PowerShell script can be executed

4.2 Comparison between already existing products and our prototype

In this paragraph, we compare the results with the already existing products. First, we compare the results with configuration management tools. Secondly, we compare the results with Image deployment software.Thirdly we compare the results with the virtual snapshots method and finally with Cloud Templates.

When comparing configuration management tools with our prototype, it's clear that our solution can configure more parameters than the configuration management tools. For example, configuration management tools can only do basic domain controller installation and configuration but not some of the more complex configuration tasks like replication time. However, our prototype is capable of completely configuring Domain Controller and joining the Active Directory Forest.

The automation, flexibility and extensibility of the proposed model makes it a better choice then existing deployment methods listed in chapter 2. In comparison to traditional Windows deployment tools like WDS it requires no existing AD, DHCP and DNS server to do the deployment. Furthermore traditional deployment tools will create a new complete disk for every VM instead of a differentiating disk. This means 20GB extra disk space per disk, which will quickly be noticeable when deploying labs as demonstrated in figure 6. Furthermore, these tools will push entire images over the virtual switch to the virtual machines creating a bottleneck of transfer speed where differentiating disks use the file system itself for deployment as well as unnecessary overhead.

Another method that can be compared with our method is the use of snapshots. There are two advantages between the snapshot method and our method. First of all, Microsoft strongly recommends to not make use of any virtual machine snapshots within Domain Controllers. Replication problem will occur when only a few Domain Controllers are rolled back through a snapshot. Secondly, there is no efficient way to update software and Operating Systems. Manually updating every virtual machine is the only proper solution to update software and Operating Systems, which require a lot of time.

Another consideration is the flexibility of the deployment. When using snapshots the deployment is essentially locked into a single configuration where all the machines will always have the same state. This in contrast to our model that allows for a much more dynamic deployment of a testlab. The configuration

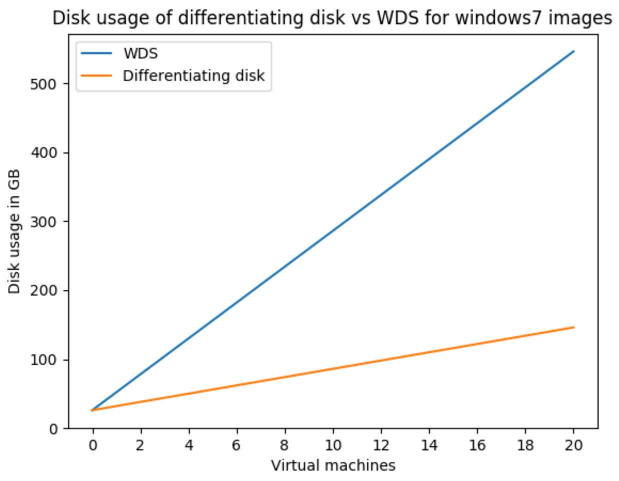

Figure 6: Disk usage comparison between WDS and Differentiating disks for Windows7 images

can be changed with each deployment and therefore allows for a more flexible deployment. However snapshots do allow for log file generation. Systems could simply be used for a while, or systems could be taken from some production VM and then snapshot. This would mean log files are already generated in the snapshot.

There are some differences between the proposed model and Cloud templates. Cloud templates in general contain Windows images that are easy to deploy and automate. However, these templates are generally updated regularly and therefore do not allow for easy deployment of older images. For security researchers it is sometimes necessary to test on older systems. The proposed model consists of an automated library that contains images which would allow researchers to deploy older images. Furthermore, the pricing list to rent multiple Virtual Machines are very high compared to buying a single server to run a hypervisor on.

4.2.1 Comparison table

In order to outline the difference between our model and the currently existing lab deployment methods we have build a comparison table. There are 8 different features which all the solutions are reflected on. Each feature is explained below:

1. Is able to deploy and install computer systems: It should be possible to deploy client systems such as Windows 7, Windows 8 and Windows 10. But also server systems such as Windows Server 2012.
2. Is able to configure computer systems: With this feature, it should be possible to configure hostnames, DNS servers, IP-address, and local administrator passwords. Moreover, it should be also possible to configure software.
3. Is able to build automatically an Active Directory Forest: Active Directory Forests should be able to be created. Furthermore, it should be possible

to deploy domain Controllers within different domains inside a forest. Finally, it should be possible to alter and create sites and replication schemes for these domains.

4. Is able to Automatically update windows servers and clients: The Operating Systems of clients and servers should be automatically updated. This also applies to the software that was installed on the systems.
5. Is able to create clients, groups and network shares that are connected to Active Directory. It should be able to create clients and groups. These groups should be allocated to network shares and other groups.
6. Only require user interaction at the start of the deployment: One of the requirements that were defined in chapter 1 was that the user input should be as minimum as possible. This feature states that only at the start of the deployment the end user must provide user input to the application or script.
7. Possibility to extend the framework to support other Operating Systems. It should be possible to add other operating systems to the testlab or other types of software for the computer systems.
8. Definable parameters such as IP-addresses and users: Lab specific configurations like domain Names, hostnames, IP-addresses, groups, and users should be definable before starting the lab deployment.

	CM	WDS	Snapshots	cloud templates	Our model
1. Deploy and install computer systems	✓	✓	✓	✓	✓
2. Configure computer systems	✓		✓	✓	✓
3. Build automatically an Active Directory Forest			✓		✓
4. Automatically update Windows servers and clients					✓
5. Create clients, groups and network-shares that are connected to Active Directory	✓		✓		✓
6. Only require user input at the start of the deployment	✓		✓	✓	✓
7. Possibility to extend the framework to support other Operating Systems	✓		✓	✓	✓
8. Definable parameters such as IP-addresses and users	✓			✓	✓

Table 1: A comparison of all the existing products as well as our model

4.2.2 outcome of the comparison table

Below, we discuss the outcome of every feature of table 1.

1. Is able to deploy and install computer systems: Every product is able to deploy a computer system.
2. Is able to configure computer systems: The configuration tools, snapshot method, the cloud template method, and our method are able to configure computer systems. Within WDS, it is possible to attach a configuration files. However, these configuration files only configures the host name, time zone, and password of computer systems.
3. Is able to build automatically an Active Directory Forest: Only the snapshot method and our method are able to automatically build an Active Directory Forest. However, as described in chapter 2, Microsoft strongly suggest to not use snapshots for domain controllers. The WDS tool and the cloud template are able to install a Domain Controller service but not configure it and create a domain.
4. Is able to Automatically update Windows servers and clients: Only our solution is able to automatically update Windows server and clients by using the image library technique as described in chapter 3. The cloud template method automatically update Windows servers but it is not possible to use an older version of Windows.
5. Is able to create clients, groups and network shares that are connected to Active Directory. WDS is not able to to create users and groups. Furthermore, the cloud template method is not able to automatically create a connection to network shares.
6. Only require user interaction at the start of the deployment: Within configuration management tools, it is possible to define everything in special defined systems catalogs. Within Snapshots, all the configuration has already been done. Therefore, only the roll/back function has to be applied. For the cloud template, it possible to define everything in the templates and upload it to the cloud provider. Finally, within our solution, it is possible to define everything in XML files.
7. Possibility to extend the framework to support other Operating Systems. The WDS application is only capable to extend the framework to support

Windows Systems. This means that no other vendor can be added to the framework.

8. Definable parameters such as IP-addresses and users: WDS cannot configure computer systems. Therefore, it also not possible to create definable parameters. Snapshots use the parameters that were defined when the testlab was created. Therefore, it also not possible to create definable parameters.

The combination of currently available techniques resulted in a less effective deployment tool compared to our model. A combination of tools would result in more overhead and less efficient lab deployment for a number of reasons. Primarily these tools are specified to do more things then just lab deployment and therefore contain a large set of tools that are unused. Furthermore, because of the way these tools are build to handle many different kinds of deployment they are inefficient when compared to our proposed model. For example, instead of using the file system directly, a combination of tools would push the entire Windows images through the virtual network to the virtual machines increasing overhead. Some other reasons include:

1. More parameters can be defined.
2. Less user input is required during deployments and installation.
3. Possibility to extend the framework to support other Operating Systems.
4. Includes the possibility to automatically update images.

4.2.3 Summary

Compared to other techniques, tools, products and methods, our solution is able to automatically deploy a realistic testlab effectively. Where the current techniques, tools, products and methods are only able to partially automatically deploy a realistic testlab. Furthermore, the proposed method in this research is a more efficient way of deploying a realistic testlab compared to a combination of deployment tools.

In comparison with other existing tools and products, our solution can deploy virtual machine more efficiently and can do more sophisticated configuration. Also, the limited amount of user interaction is something that differs our solution with existing tools and products.

5 Conclusion

We have investigated the currently available techniques and methods to deploy and configure a testlab. The different techniques and methods can roughly be divided into four groups:

1. Configuration management
2. Image deployment
3. Virtual machine snapshot
4. Cloud templates

By using one these techniques and tools it is only partially possible to automatically deploy a realistic testlab. Some requirements cannot be fulfilled by using current techniques and require manual interaction that was defined in chapter 1. Therefore, we have created our own model to automatically deploy a realistic testlab. Our model uses the following phases and methods to deploy a realistic testlab:

1. Specify lab
2. Deployment of server and OS installation
3. Server provisioning
4. Software installation and configuration on servers
5. Deployment of clients and OS installation
6. Client provisioning
7. Software installation on client
8. Log file and user behavior emulation

In order to verify our model, we have created a prototype. Compared to other the currently existing techniques, tools, products, and methods our prototype is able to automatically deploy a realistic testlab. Where the other tools and methods are only able to partially automatically deploy a realistic testlab.

In order to compare our method with the currently existing tools and products, we created an comparison table. The outcome of the comparison table was that our method supports all requirements and the currently existing tools and products do not. Despite the fact that the snapshot technique and the cloud templates method support almost as many requirements, they have a few distinct disadvantages. The snapshot technique doesn't automatically update the Operating Systems in the testlab. For security researchers, it's sometimes preferred to have an older Operating System in order to test vulnerabilities on different versions of Operating Systems. A big disadvantage within the cloud template solution is the pricing list to rent multiple Virtual Machines.

Also, the combination of currently available techniques resulted in a less effective deployment tool compared to our model due to the following reasons:

1. More parameters can be defined.
2. Less user input is required during deployments and installation.
3. Possibility to extend the framework to support other Operating Systems.
4. Includes the possibility to automatically update images.
5. Less overhead

Our prototype uses Powershell to deploy, install, and configure servers as well as clients. However, a few tasks in Powershell can not be done. Therefore, other tools are required to complete the task. Moreover, the requirement user input is minimal to deploy the testlab.

This research has shown that it is possible to automate a fast and easy rollout of a realistic Windows test environment with minimal user interaction by using the methods and techniques specified in our model. We have proven the viability of the model by building a prototype of the model. The prototype has shown that our model works and is more efficient and effective in Windows testlab deployment than current solutions.

6 Future Work

Further research can be done on different levels. First, the proposed solution can be improved by including more Windows Operating systems and other non-Microsoft operating system. Also, a more user-friendly interface could be built with error-detection mechanisms included. Moreover, more services and applications can be included in the proposed method.

The proposed testlab is mainly designed for security research are not any router, networking security appliances, network monitoring tools, VLANs or different networks present in the testlab. More research can be conducted on this topic.

Finally, log and event files should be added to the testlab in order to mers. However, the testlab has only a very basic network configuration. Theyake the testlab more realistic. In this research, we have only investigated whether or not it is possible to alter the timestamp. We have also altered the current time of client machine to verify that the timestamp is different. More research can be conducted to determine what the side effects are when the time continuously changes within the testlab. Furthermore, a group calling themselves The Shadow Brokers publicly released a cache of NSA hacking tools. Screenshots of the sale suggest that the Shadow Brokers group has in its possession a tool thats capable of editing and tampering with Windows event logs [1]. This tool might become available in the near future and could be added as a feature to the solution.

References

[1] Chris bing, cyberscoop, januari 12 2017. https://www.cyberscoop.com/shadow-brokers-leak-nsa-linked-microsoft-hacking-tools/.

[2] Dhcp and automatic private ip addressing. https://technet.microsoft.com/en-us/library/cc958957.aspx.

[3] Microsoft developer, active directory powershell, august 18, 2009, m. ali. https://blogs.msdn.microsoft.com/adpowershell/2009/08/18/.

[4] Microsoft developer network, event logging and viewing, from chapter 3, microsoft windows 20012 administrator's pocket consultant by william r. stanek. https://msdn.microsoft.com/en-us/library/bb726966.aspx.

[5] Software engineering institute, configuration management, carnegie mellon. http://www.sei.cmu.edu/productlines/frame_report/config.man.htm.

[6] Technet, microsoft exchange load generator, december 12, 2007. https://technet.microsoft.com/en-us/library/bb508893(v=exchg.80).aspx.

[7] Technet, microsoft system center, introduction to operating system deployment in configuration manager, may 14, 2015. https://technet.microsoft.com/en-us/library/gg682108.aspx.

[8] Technet, microsoft, using differencing disks. https://technet.microsoft.com/en-us/library/cc720381(v=ws.10).aspx.

[9] Windows server 2012 unleashed by andrew abbate; michael noel; rand morimoto; omar droubi; guy yardeni; chris amaris published by sams, 2012.

[10] Winzapper, a windows 2000 alterion tool.

[11] S. Sardesai, S. Khan, D. Kumar, G. Parupudi, and V. Deo. Operating system deployment methods and systems, September 23 2004. US Patent App. 10/667,123.

[12] Powershell team. Powershell on linux and open source!, 2016. https://blogs.msdn.microsoft.com/powershell/2016/08/18/powershell-on-linux-and-open-source-2/.

Appendices

Apendix A The prototype in its current form can be found on github: `https://github.com/alfuananzo/Stained-Glass`