University of Piraeus
Department of Digital Systems

MSc in Security of Digital Systems

Master Thesis

Designing a free Data Loss Prevention system

Dimitrios Koutsourelis

March 2014

Supervisors

Sokratis Katsikas, Professor

University of Piraeus

Spyros Papageorgiou, Captain (OF-5)

Hellenic National Defense General Staff/ Directorate of Cyber Defense

Abstract

Data Loss is an everyday threat within all organizations. The leaking of confidential data ultimately results in a loss of revenue. A quarter of all Data Loss is widely attributed to the use of applications such as Voice, Email, Instant Messaging, Social Networks, Blogs, P2P and Videos. A common approach, adopted to prevent Data Loss is the use of sophisticated data detection algorithms and the deployment of systems to control applications, attempting to access the outside world [1]. Many vendors currently offer data loss prevention products for no small amount of money.

This thesis focuses on providing all the necessary information needed to fully comprehend the terms that wrap around data loss prevention and illustrating some of the most common mechanisms and techniques used by the available software. Also, two publicly available, free, data loss prevention software tools, MyDLP and OpenDLP, are presented and analyzed. In the third part of this project, a system architecture is presented, designed to combine the two previously mentioned software in a way such that they co-exist inside an information system, complementing each other, providing solid data loss prevention services.

Table of Contents

List of Figures[1]

[1] All images representing a network architecture were created using Microsoft Visio 2013.

Images presenting the operation of OpenDLP and MyDLP are screenshots taken while running the DLP software under test conditions.

1 Introduction

1.1 Background

Over the last decade, enterprises have become increasingly reliant on digital information to meet business objectives. On any given business day, significant amounts of information fuel business processes that involve parties both inside and outside of enterprise network boundaries. There are many paths for these data to travel and they can travel in many forms—e-mail messages, word processing documents, spreadsheets, database flat files and instant messaging are only a few examples. Much of this information is innocuous, but in many cases a significant subset is categorized as "sensitive" or "proprietary," indicating that this information needs to be protected from unauthorized access or exposure. This need can be externally driven by privacy and other types of regulation, or internally driven by business objectives to protect financial, strategic or other types of competitive information.

Most enterprises employ safeguards to control sensitive information. Often, however, these controls are inconsistent and are managed at different points in the enterprise with different levels of diligence and effectiveness. The result is that despite their efforts, organizations around the globe leak large amounts of sensitive information. These leaks create significant risk to enterprises, their customers and business partners with the potential to negatively impact a firm's reputation, compliance, competitive advantage, finances, customer trust and business partnerships. Concerns over this need, to better control and protect sensitive information, have given rise to a new set of solutions aimed at increasing an enterprise's ability to protect its information assets. These solutions vary in their capabilities and methodologies, but collectively they have been placed in a category known as data loss prevention (DLP). While still an adolescent technology, DLP is seeing an increase in adoption and an increasing number of products entering the market [2].

1.2 Objectives

This thesis focuses on providing necessary information and explanation of the basic terms regarding data loss prevention, in a simple and comprehensible way. Its main purpose is to present a solid and complete DLP solution based solely on free software. A DLP system with specific architecture is proposed, which combines and uses under full collaboration, two, free, publicly available, independent DLP tools: MyDLP and OpenDLP. Moreover, a solution to automate the operation of the previously mentioned software is presented, providing functionality without the need of any user's intervention.

1.3 Structure

The first (1st) chapter (Chapter 1) of this thesis highlights the need to protect information, as well as the importance and value of DLP systems. The objectives and the main purpose of the project are also defined.

The second (2nd) chapter (Chapter 2) lays the foundation for effective understanding of the DLP concept, what makes it tick and its key components, along with descriptions and presentations of the methods and techniques used.

Chapter 3 addresses the software tools that were chosen and used to provide DLP services. Sub-chapter 3.1 focuses on OpenDLP and sub-chapter 3.2 on MyDLP. For each tool, the basic components and functions are presented in order to fully comprehend the way they operate and differ from each other.

Chapter 4 introduces a DLP system design, which is based on the combination of OpenDLP and MyDLP. The need for automation is discussed and several limitations that were met during the implementation phase are addressed.

The conclusions and a review of the system proposed can be found in chapter 5.

2 Data Loss Prevention - DLP

2.1 What is Data Loss Prevention?

Data Loss Prevention is a recent type of security technology. While tools such as firewalls and Intrusion Detection and Prevention systems (IDS/IPS) look for anything that could pose a threat to an organization, DLP focuses on identifying sensitive data. It looks for content that is critical to an organization [3].

Specifically, through policies a DLP system automatically makes sure no sensitive data is stored, sent or accessed where it shouldn't be, while still allowing users to use the

tools and services they choose and need to fulfill their tasks. Unlike traditional white and black-listing mechanisms, the DLP only blocks the actions where sensitive data is involved, e.g. sending e-mails is perfectly acceptable, but not if they contain sensitive data [4].

The term data loss prevention (or sometimes data "leak" prevention) is often used to describe a specific product or technical solution which addresses the need to prevent loss of intellectual property or otherwise sensitive information. DLP itself encompasses many facets of information security which often intersect various technological boundaries. Understanding, that Data Loss Prevention is a process, and not a product, is the first step in building an effective DLP solution.

2.2 Definition

For a more formal definition, Securosis – A leading research and advisory firm that has been following the development of DLP closely, defines Data Loss Prevention as followed:

"Products that, based on central policies, identify, monitor, and protect data at rest, in motion, and in use, through deep content analysis."

Thus the defining characteristics are:

- Deep content analysis
- Central policy management
- Broad content coverage across multiple platforms and locations [5].

2.3 Various Related Terms

The terms "data loss" and "data leak" are closely related and are often used interchangeably, though they are somewhat different. When media containing sensitive information is lost and subsequently acquired by unauthorized party, Data loss incidents turn into data leak incidents. However, a data leak is possible without the data being lost in the originating side.

Some other terms associated with Data Loss Prevention are listed, as can be seen in SANS Institute DLP Basics whitepaper:

- Data Loss Prevention/Protection
- Data Leak Prevention/Protection
- Information Loss Prevention/Protection
- Information Leak Prevention/Protection
- Extrusion Prevention
- Content Monitoring and Filtering
- Content Monitoring and Protection [3].

2.4 Data Loss Threats

Loss of data comes in many different forms. Knowing how it occurs helps developing the appropriate prevention measures. Data loss can be categorized in the following types:

- Accidental data loss.
- Insider attacks.
- External attacks.

2.4.1 Accidental data loss

As soon as data leaves company property, control over it is lost and data is considered unrecoverable. A typical cause for accidental data loss is employees unfamiliar with company policies. Frequently, they are unaware of the sensitivity and importance of the documents they are working with, or they overestimate their own knowledge regarding computer security.

Common examples of accidental data loss are:

- Sending documents containing sensitive data, via e-mail without any form of encryption.
- Saving sensitive documents to an unencrypted, non password-protected remote storage device for further transportation.
- Unsecure uploading of files to online services, often available publicly and discoverable with a simple search (e.g. cloud services)

Another common cause that results in insecure handling of sensitive material is employees' lack of proper training. Average users do not usually know how to mark a file as sensitive or encrypt it.

One of DLP's greatest strengths is combating accidental data loss. Strong detection policies and proper integration of such a system can easily stop accidental leaks. Under a correctly implemented DLP system's control, data only flows through authorized channels, defined by company policies. Sensitive data classification and encryption is also automated.

2.4.2 Insider Attacks

An insider attack is a malicious attack perpetrated on a network or computer system by a person with authorized system access.

Insiders that perform attacks have a distinct advantage over external attackers because they have authorized system access and also may be familiar with network architecture and system policies/procedures. In addition, there may be less security against insider attacks because many organizations focus on protection from external attacks. An insider attack is also known as an insider threat [6].

A DLP system is not an access control system and should not be seen as a replacement to one. For a more effective protection against insiders' attacks, but also external attacks, a DLP system should be always used in conjunction with other security systems, such as Firewalls and IDSs. Most importantly it is up to the companies themselves to make sure users only have access to what they need in order to do their job, and separate data according to level of sensitivity [7].

"The role of a DLP comes into play when a user is working with sensitive data. The DLP does not disallow the user to view the sensitive content (which is what an access control system would do), but instead makes sure the user does not treat the data in an irresponsible manner (e.g. exporting sensitive data outside company's area of control). Having a DLP in place might also discourage users from committing data theft. If they know such a system is in place, they will also recognize an inherent risk for being detected when conducting data theft" [4].

2.4.3 External Attacks

An external attack is any attempt to destroy, expose, alter, disable, steal or gain unauthorized access to physical or logical resources, or make unauthorized use of an asset. An external attack usually is perpetrated by someone with bad intentions so they are what most people consider as hacker attacks. The motives behind these attacks are mainly of financial nature.

The effectiveness of a DLP system in these types of attacks depends mainly on the attacker's knowledge of a system and the functionality of the attack. With the use of special patterns, for detecting and stopping data-stealing malware, or other similar threats, an alert might be triggered, resulting in the detection of the breach.

The existence of a DLP system might have some effect on detecting and stopping remote attacks, but again having a firewall, IDS, anti-virus, conducting employee security awareness training and applying good security practices goes a lot further in remediating this threat, than just installing a DLP system [8].

2.5 Content Awareness

DLP solutions have the ability to analyze deep content using a variety of techniques. Content analysis, also known as content awareness, is one of the main characteristics of DLP and is very different from analyzing context. The difference between content and context can be easily made comprehensible by providing the following simple parallelism.

"*Think of content as a letter, and context as the envelope and environment around it. Context would include anything, short the content of the letter itself. For example letter's source, destination, size, recipients, sender, header information, metadata, time, format. Context is highly useful and any DLP solution should include contextual analysis as part of an overall solution. A more advanced version of contextual analysis can be done, which involves deeper analysis of the content, its environment at the time of analysis, and the use of the content at that time. By using such advanced methods it is able to identify information like, which business unit the current holder of the envelope belongs to, its virtual location, what parsing/reading application is used and so on.*

Content awareness on the other hand, involves peering inside containers and analyzing the content itself. Its advantage is that while context is used, there is no restriction by it. If a piece of sensitive data needs protection, this protection is needed everywhere — not just in obviously sensitive containers. It's the data that should be protected, not the envelope. For this reason, it makes a lot more sense to open the letter, read it, and decide how to handle it. This is more difficult and time consuming than basic contextual analysis".

2.5.1 Content Analysis Techniques

"The first step in content analysis is capturing and opening the files containing the data in need of protection. The DLP engine then needs to parse the context and dig into it. For plain text messages this is easy, but when a peek inside binary files is necessary things get more complicated. All DLP solutions solve this using file cracking. File cracking is the technology used to read and understand the file, even if the content is buried multiple levels down. For example, it's not unusual for the cracker to read an Excel spreadsheet embedded in a zipped Word file. The product needs to unzip the file, read the Word doc, analyze it, find the Excel data, read that, and analyze it. Other situations get far more complex, like a .pdf embedded in a CAD file. Many of the products on the market today support around 300 file types, embedded content, multiple

languages, double byte character sets for Asian languages, and pulling plain text from unidentified file types. All serious tools have quite a bit of proprietary capability in addition to the embedded content engine.

Some tools support analysis of encrypted data if enterprise encryption is used with recovery keys, and most can identify standard encryption and use that as a contextual rule to block/quarantine content. Note that this is often limited to email and certain file transfers or types, and you need to ask carefully if you require more expansive encryption detection (e.g., for encrypted .rar files on unusual ports or protocols)" [5].

Several methods to analyze content exist, each with its own uses:

- **Content Analysis using Regular Expressions:** The most common analysis technique. It analyzes the content for specific rules that can be easily configured, mainly used for detecting easily identified pieces of structured data like numbers that meet credit cards format, healthcare records etc. It is also called "Rules-based analysis".

Below is presented an example of regular expressions used for detecting numbers that match credit card format:

```
(\D|^)\%?[Bb]\d{13,19}\^[\-\/\.\w\s]{2,26}\^[0-9][0-9][01][0-9][0-9]{3}
```

- **Fingerprinting:** This method works by comparing the inspected file to a group of files in a database, already identified for containing sensitive data. If a match occurs the file is marked as a sensitive file. A bit by bit comparison is often used, but an even more effective way commonly used is by comparing the hash values of the files. It is mostly used for media files and other binaries where textual analysis isn't necessarily possible, such as photos, audio, movies, and certain proprietary design and engineering files.
- **Partial Document Matching:** It is used for protecting sensitive documents, or similar content with text, and source code. Generally, unstructured content that is known to be sensitive. What is checked is the complete text of the document, or even excerpts as small as a few sentences. The technique that is used is known as cyclical (or overlapping) hashing, where you take a hash of a portion of the content, offset a predetermined number of characters, then take another hash, and keep adding hashes of document segments until done. Outbound content is run through the same hash technique, and the hash values are compared for matches.
- **Statistical Analysis:** This category includes a wide range of statistical techniques which vary greatly in implementation and effectiveness. The most well known and

commonly used among statistical analysis techniques are machine learning algorithms, which seek to automate the identification of sensitive contents through training. DLP systems that use machine learning algorithms work similar to a spam filter and include a learning phase where the system is trained to what it should consider as sensitive data.

- **Conceptual/Lexicon:** The conceptual/lexicon method of detection uses a combination of dictionaries and rules. It mixes together word usage patterns commonly associated with specific concepts, for example sexual harassment, racist statements etc. Because of the loose nature of the rules, this technique is not very reliable as it is prone to both false positives and false negatives.
- **Categories:** Templates that contain one or more of the previously mentioned techniques in combination with each other to detect sensitive data. Often pre-built categories exist in many DLP systems with rules and dictionaries for common types of sensitive data.

2.6 DLP System Architecture and Data Protection Types

The goal of DLP is to protect content throughout its lifecycle. There are three basic types of protection in DLP systems, related to the primary states of information: Data at Rest, Data in Motion and Data in Use.

A complete DLP system must combine and implement all three data protection technologies in the most efficient way. The basic components of a DLP system are the Endpoint DLP which is installed directly on a workstation and keeps track of how data is stored, the Network DLP which is often placed between the LAN and WAN networks, as a proxy and monitors network traffic and the DLP Server or Central Management Console which manages both of the above components and is mainly responsible for policy deployment. The infrastructure of such a DLP system is presented in Figure 1, followed by a description and explanation of the basic data protection types and components of a DLP system.

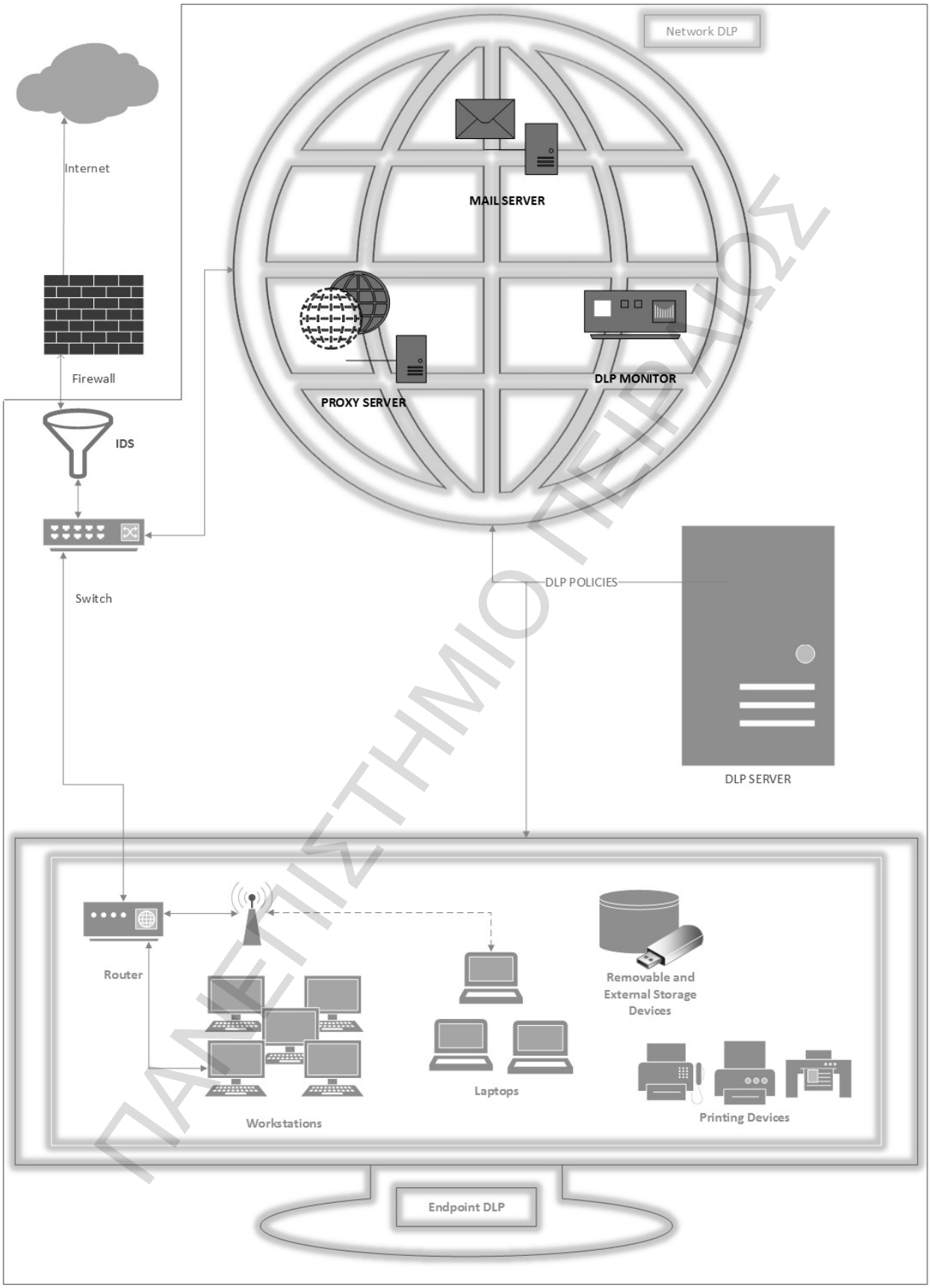


Figure 1: Basic DLP system architecture

2.6.1 Data at Rest (Endpoint DLP)

A basic function of DLP solutions is the ability to track specific types of information and locate where they are stored throughout the enterprise. This function is also called content discovery and means that the DLP solution must have the ability to seek out and identify specific file types, whether they are on file servers, storage area networks (SANs) or even end-point systems. Once found, the DLP solution scans the content of these files to determine whether specific pieces of information are present, such as credit card or social security numbers. To accomplish these tasks, most DLP systems utilize applications which are deployed remotely and are used to log onto each end system and "crawl" through data stores, logging the location of specific information sets based on a set of rules that have been entered into the DLP management console. Such applications are known as "crawlers".

Collecting this information is a valuable step in allowing an organization to determine where its key information is located, whether its location is permitted within existing policies, and what paths these data might travel that would violate information policies (Figure 2) [2].

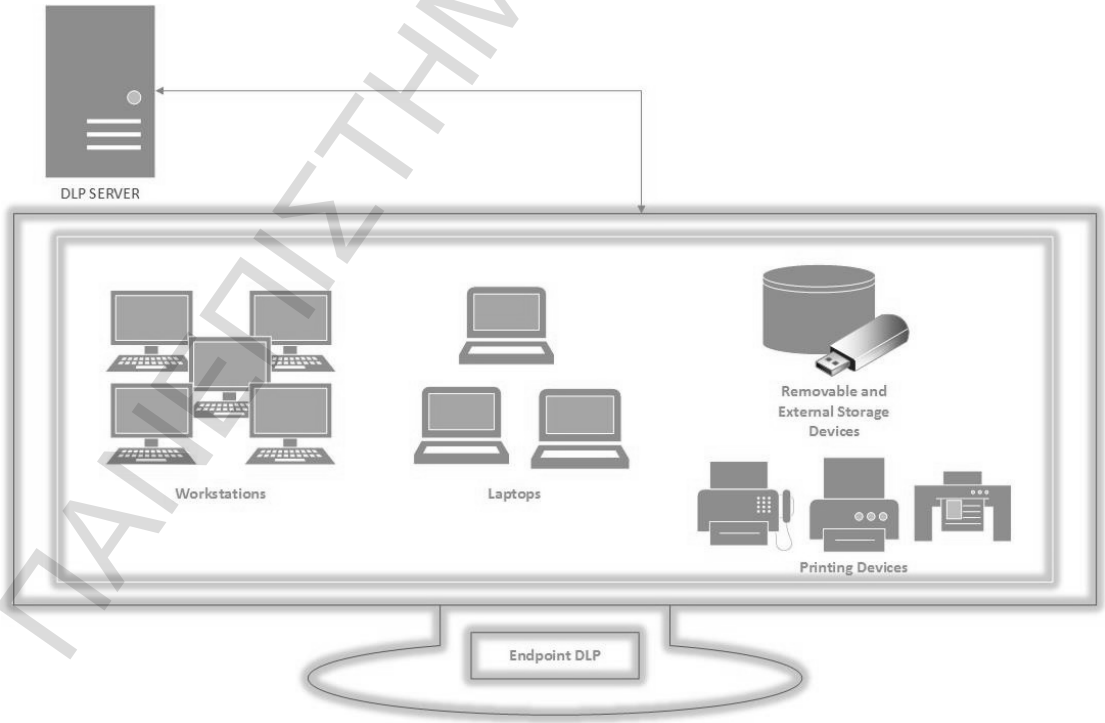


Figure 2: Endpoint DLP architecture

2.6.2 Data in Motion (Network DLP)

Data in motion is data that is being transmitted over a network. To monitor data movement on enterprise networks, DLP solutions use specific network appliances or embedded technology to selectively capture and analyze network traffic. Files that are sent across a network are usually broken into packets. *"To inspect the information being sent across the network, the DLP solution must be able to: passively monitor the network traffic, recognize the correct data streams to capture, assemble the collected packets, reconstruct the files carried in the data stream, and then perform the same analysis that is done on the data at rest to determine whether any portion of the file contents is restricted by its rule set."* This ability of DLP systems follows the principles of content awareness as described in Chapter 2.5 and the process is known as deep packet inspection (DPI). DPI does not only check the basic header information of a packet (which is similar to the "to" and "from" information found on a postal envelope), but also reads the contents within the packet's payload (akin to the letter within the postal envelope). This DPI capability allows the DLP system to inspect data in transit and determine contents, source and destination. If sensitive data are detected flowing to an unauthorized destination, the DLP solution has the capability to alert and optionally block the data flows in real or near real time, again based on the rule set defined within its central management component. Based on the rule set, the solution may also quarantine or encrypt the data in question. An important consideration for network DLP is that the data must be decrypted before the DLP solution can inspect the data. Either the DLP solution must have the capability to do this itself (by having this feature and the necessary encryption keys) or there must be a device that will decrypt the traffic prior to its inspection by the DLP module, and re-encrypt once the data have been inspected and allowed to pass [2] (Figure 3).

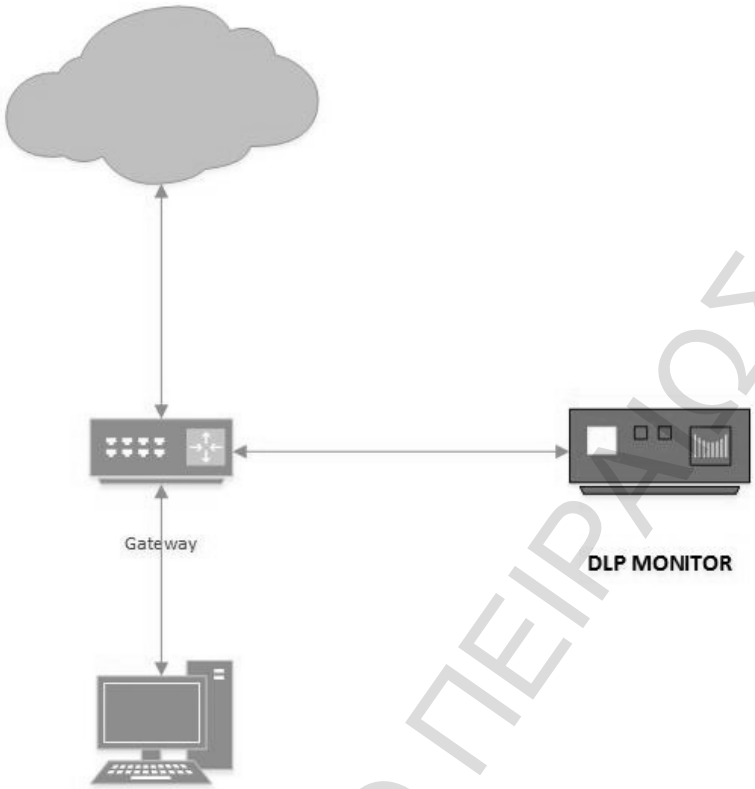


Figure 3: Data in Motion DLP architecture

2.6.3 Data in Use

Data in use is perhaps the most challenging aspect of DLP. *"Data in use primarily refers to monitoring data movement stemming from actions taken by end users on their workstations, whether that would entail copying data to a thumb drive, sending information to a printer, or even cutting and pasting between applications*." DLP solutions typically accomplish this through the use of a software programs known as agents, which are usually controlled by the same central management console of the DLP solution [2].

2.6.4 Central Management

The DLP server is the core of a DLP system. Its responsibilities are many, including Endpoint agent deployment and agent management tasks, like policy deployment, software patching and log collection, software updates, common definitions and licensing information from the vendor's server, logs. It also forwards critical alerts to system administrators, generates reports based on collected data and provides tools to create and manage DLP policies.

Many of the points above can usually be accessed and managed through an administration interface. This is commonly a web application accessible from any browser, although command-line interfaces also exist. The DLP server usually comes in the form of a hardware or virtual appliance[2] [4].

[2] A virtual appliance is a virtual machine image designed to run on a virtualization platform (e.g., Virtual Box, Xen, VMware Workstation, Parallels Workstation)

3 Data Loss Prevention Tools

The tools that are used in this project, OpenDLP and MyDLP, to provide Data Loss Prevention services are presented and analyzed in the following paragraphs.

3.1 OpenDLP

OpenDLP is a free and open source, agent- and agentless-based, centrally-managed, massively distributable data loss prevention tool released under the GNU General Public License (GPL) [9]. Given appropriate Windows, UNIX, MySQL, or MSSQL credentials, OpenDLP can simultaneously identify sensitive data at rest on hundreds or thousands of Microsoft Windows systems, UNIX systems, MySQL databases, or MSSQL databases.

Figure 4: OpenDLP Logo [15]

OpenDLP has two basic components:

- A web application to manage Windows agents, Windows/UNIX/database agentless scanners and scan results.
- A Microsoft Windows agent used to perform accelerated scans of up to thousands of systems simultaneously.

The agent is written in C programming language and has no .NET [10] requirements. It runs on Windows 2000 and later systems, as a low priority service so users do not see or feel it. The agent resumes automatically upon system reboot with no user interaction needed. It securely transmits results to the central management web application at user-defined intervals over two-way-trusted SSL connection. In order to identify sensitive data

it uses Perl Compatible Regular Expressions (PCREs) [11] for pattern matching and can read inside compressed files (e.g zip, rar, 7z). Furthermore, it limits itself to a percent of physical memory of the machine running it, so there is no thrashing when processing large files.

The web application on the other hand distributes, deploys and starts agents over Network Basic Input/Output System or Server Message Blocks (Netbios [12]/SMB [13]). When the scan is done, it automatically stops, uninstalls, and deletes agents. It can pause, resume, and forcefully uninstall agents in an entire scan or on individual systems. It can also initiate agent-less scans, where the DLP server and not the client, does all the processing work. Scan results are received concurrently and securely from hundreds or thousands of deployed agents, over a two-way-trusted SSL connection. The web interface gives users the ability to create Perl-compatible regular expressions (PCREs) for finding sensitive data at rest, create reusable profiles for scans that include whitelisting or blacklisting directories and file extensions, review findings and identify false positives and finally export results as XML files. It is written in Perl with MySQL backend and has also the capability of deploying Windows agents through existing Meterpreter [14] sessions [15].

OpenDLP only deals with one area of potential data loss: the Endpoint (data at rest). It is only capable of searching for regular expressions found in plaintext. One of its main limitation is that encryption defeats this tool. Since encryption converts plaintext into an unreadable form, regular expression scanning is rendered useless [16].

OpenDLP's main interface looks like the following:

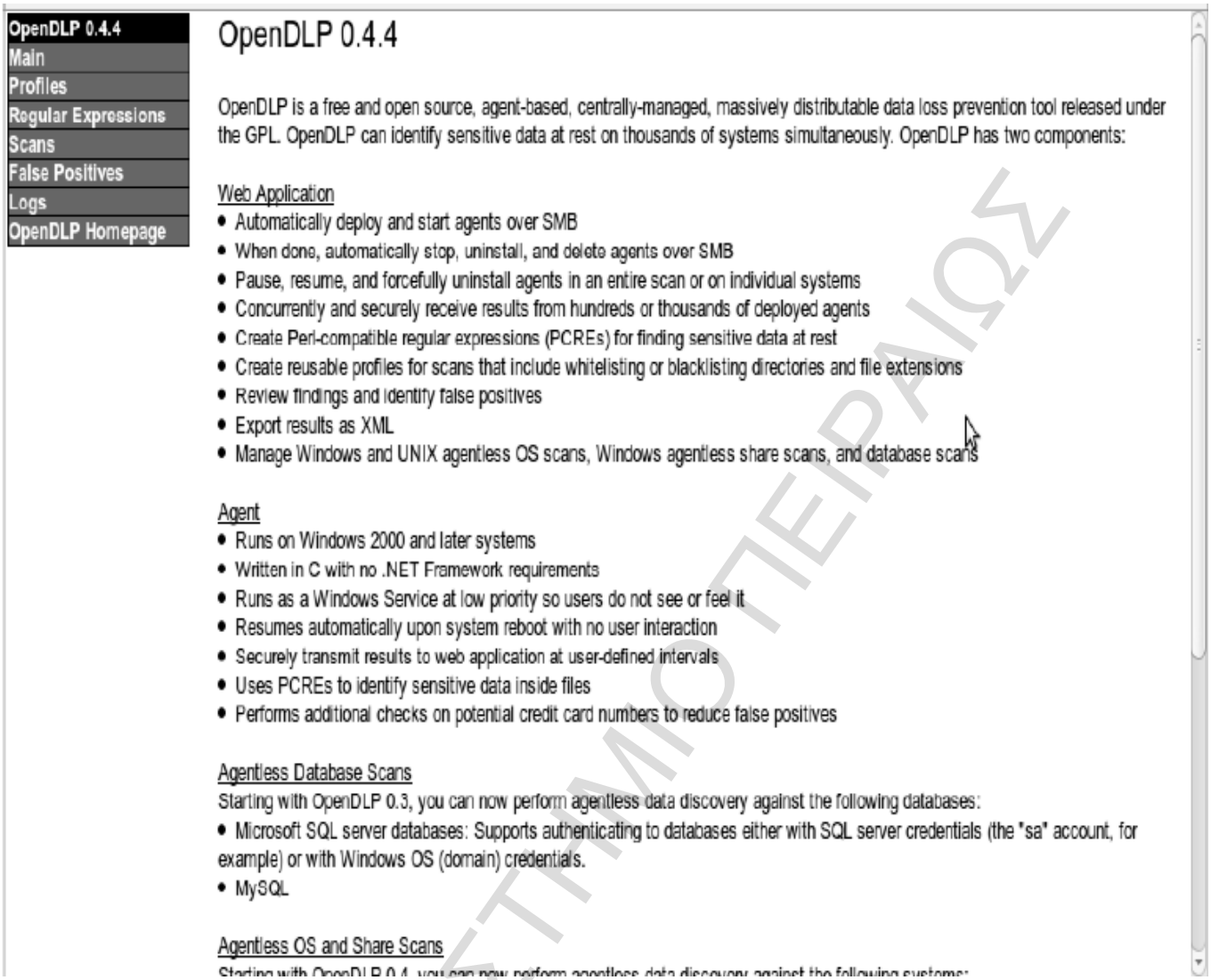


Figure 5: OpenDLP's main page

3.1.1 Workflows

OpenDLP can be used in different ways, but the basic workflow is as follows:

- Review the provided Regular Expressions for data to look for (Figure 6).

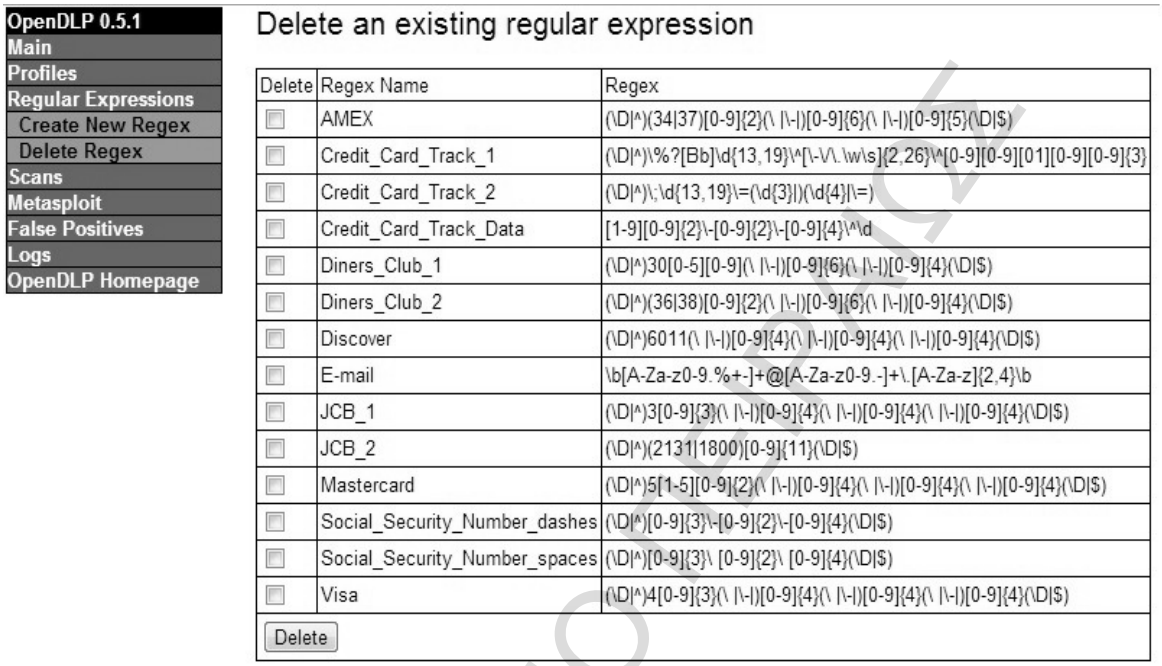


Figure 6: OpenDLP Regular Expressions management

- Create a profile with authentication credentials and policy settings (Figure7).

Figure 7: New OpenDLP Scan Profile

Profiles are used to define the scan types to be done as well as to provide and store the credentials necessary to perform the scan. OpenDLP uses the following profile types:

- Windows Filesystem (agent)
- Windows Filesystem (agentless over SMB)
- Windows Network Share (agentless over SMB)
- UNIX Filesystem (agentless over SSH)
- Microsoft SQL Server (agentless)
- MySQL (agentless)

To scan a Windows file system with an agent a local or domain administrator account is necessary.

- Start a scan by providing IP address or a list of IPs (Figure 8).

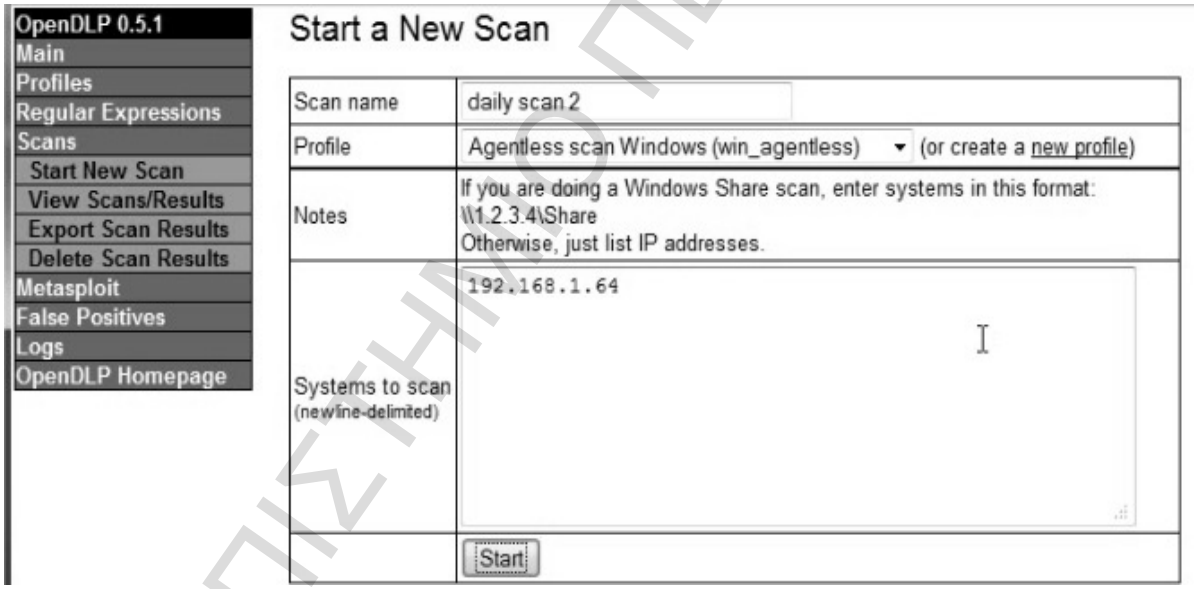
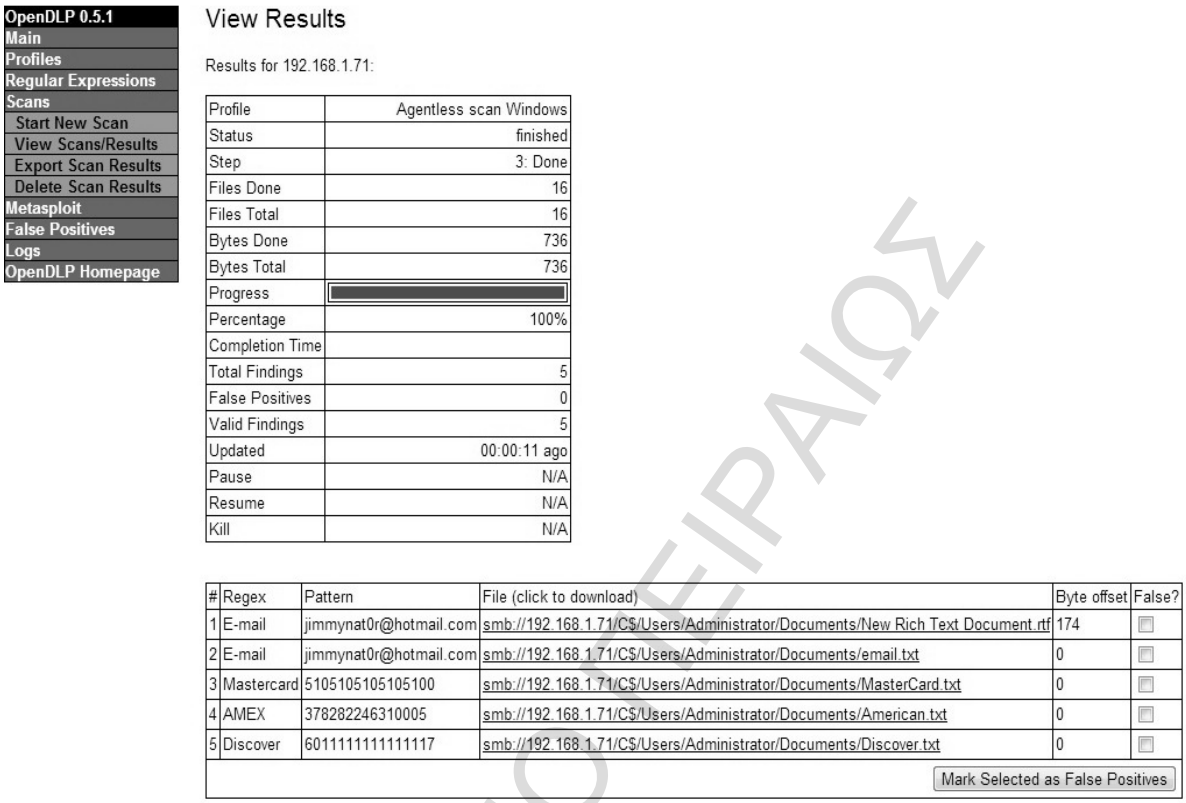


Figure 8: OpenDLP Scan Start

- Review the scan results and mark false positives (Figure 9).



OpenDLP 0.5.1
Main
Profiles
Regular Expressions
Scans
Start New Scan
View Scans/Results
Export Scan Results
Delete Scan Results
Metasploit
False Positives
Logs
OpenDLP Homepage

View Results

Results for 192.168.1.71:

Profile	Agentless scan Windows
Status	finished
Step	3: Done
Files Done	16
Files Total	16
Bytes Done	736
Bytes Total	736
Progress	
Percentage	100%
Completion Time	
Total Findings	5
False Positives	0
Valid Findings	5
Updated	00:00:11 ago
Pause	N/A
Resume	N/A
Kill	N/A

#	Regex	Pattern	File (click to download)	Byte offset	False?
1	E-mail	jimmynat0r@hotmail.com	smb://192.168.1.71/C$/Users/Administrator/Documents/New Rich Text Document.rtf	174	
2	E-mail	jimmynat0r@hotmail.com	smb://192.168.1.71/C$/Users/Administrator/Documents/email.txt	0	
3	Mastercard	5105105105105100	smb://192.168.1.71/C$/Users/Administrator/Documents/MasterCard.txt	0	
4	AMEX	378282246310005	smb://192.168.1.71/C$/Users/Administrator/Documents/American.txt	0	
5	Discover	6011111111111117	smb://192.168.1.71/C$/Users/Administrator/Documents/Discover.txt	0	

Mark Selected as False Positives

Figure 9: View Scan Results

- Report any suspect business sensitive or compliant data found.
- Work with the information owners and Office of Information Security to develop a remediation plan [17].

3.2 MyDLP

MyDLP is an open-source software (server and client) application that can monitor confidential data stored in servers and systems across the network. It can prevent anyone from accidentally sending confidential data out of the network or deliberately stealing it via email, web, instant messaging, external storage devices, printers, etc. MyDLP development project has made its source code available under the terms of the GNU General Public License.

MyDLP supports data loss prevention mechanisms for data in motion (Network), data in use and data at rest (Endpoint) [18].

Figure 10: MyDLP logo [18]

MyDLP consists of the following components:

- **MyDLP Network**: Network server, which is used for high load network operations such as intercepting TCP connections and hosting MyDLP network services.
- **MyDLP Endpoint**: Remote agent, which runs on endpoint machines in order to inspect end user operations such as copying a file to an external device, printing a document and capturing screenshots.
- **MyDLP Web User Interface**: Management interface for system administrators to configure MyDLP. It pushes relevant parts of system configuration to both MyDLP Network and MyDLP Endpoint.

MyDLP has two editions, Community and Enterprise. While the Community edition of MyDLP is free of cost, the Enterprise edition has a price-tag (based on the number of users). The Community edition offers comprehensive DLP capabilities, but lacks many features found in the Enterprise edition [19]. The basic limitations of MyDLP free solution, compared to the full edition are the following:

- MyDLP agents can be installed only in systems using Windows OS. No Support for Linux/Unix/other operating systems.
- Lack of automatic Content Discovery/ Endpoint-Workstation Discovery.
- No Microsoft Active Directory Integration [20].
- Email Notifications to the administrator of the system when alerts arise are disabled.
- No Syslog [21] Integration.

The full list of the differences between Community and Enterprise Edition can be found in the following web address: http://www.mydlp.com/features/.

In order to better comprehend the services provided by MyDLP, images of the web interface and its basic operations are presented.

3.2.1 Policy Management

The policy tab is used to define policies (Figure 11). The policy objects tree on the left is used to drag and drop predefined or custom, user made, objects into policy rules. On the right side there is the rule table which represents the DLP policy. The rule types supported in the community edition are: **Web Rules**, which are used to monitor and

control web traffic, **Mail Rules** with which emails are controlled and monitored, **Removable Storage Rules**, which are used to control data moved to removable memory devices such as USB memory sticks, removable hard drives, smart phones etc, **Printer rules,** to control print jobs and **Screenshot Rules**, to prevent print screen function while a sensitive application is running.

It is important to note that in order for Web rules to work, client traffic has to be redirected to the MyDLP Server. It's conceivable that a proxy server with the appropriate configuration has to be set up.

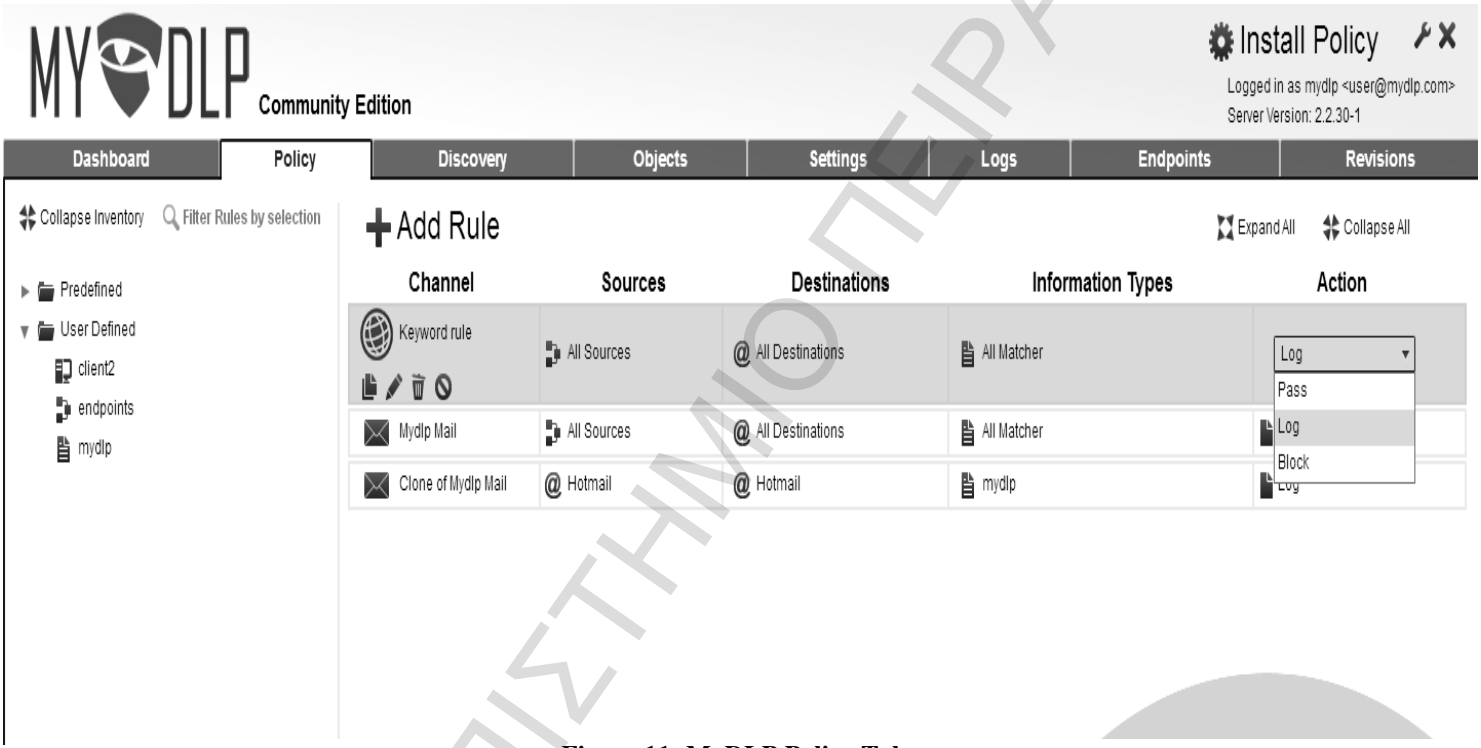


Figure 11: MyDLP Policy Tab

Each Rule has a five (5) part structure:

- **Channel**, which indicates the type of the rule and its name.
- **Sources** field, which restricts the rule on a certain user or a user group which can be denoted by IP address, network, Active Directory element or an email address depending on the rule type.
- **Destinations**, which can be a domain, directories or application names.
- **Information Type** field represents the information to be searched in the related data channel during inspection. There many types of information types and custom information types can be also defined.
- **Actions** field shows the desired action to be taken when defined information type found on a data channel. Available actions in the free edition are Pass, Block and Log. **Pass** action allows information to pass through data channel freely without any logs. This action is available for all rule types. **Log** action allows information to pass through data channel but generates event log. This action is not available for screenshot rules. Finally, **Block** action prevents information to pass through data channel and generates event log. This action is not available for removable storage inbound rules [22].

3.2.2 Endpoint Management

MyDLP Interface provides a way (through the Endpoints Tab) to check information about the systems in which the MyDLP agent has been installed and also control the behavior of the agent in the registered terminals (Figure 12). Through the Web UI, the administrator can see whether a system is online or offline, the IP addresses of the controlled systems, their computer name, the user account they are logged in and under which they have registered in the DLP server, the version of the agent and details about their first and last active session. The MyDLP admin can also search for specific endpoints and issue command “Truncate” to clear endpoints’ database on the server.

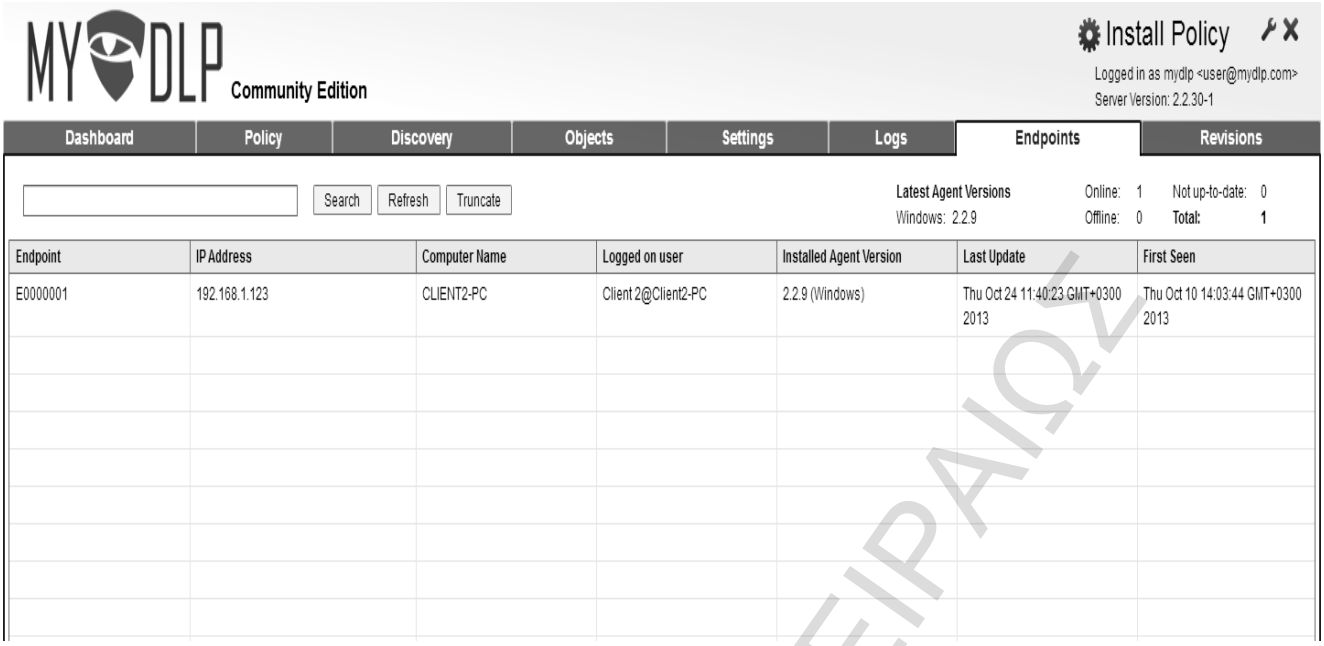


Figure 12: Endpoints MyDLP Tab

3.2.3 Logs

The administrator of the system can monitor all DLP related events in the logs tab (Figure 13). On the top side there is the log tool bar. Using the log tool bar a user is able to search for logs in a specific time period. On the middle there is the log table. Logs listed in log table have the following structure:

1. **Date**: Data and time of the event.
2. **Source**: Source of data.
3. **Destination**: Destination of data.
4. **Policy**: Related policy rule.
5. **Details**: Details about rule [22].

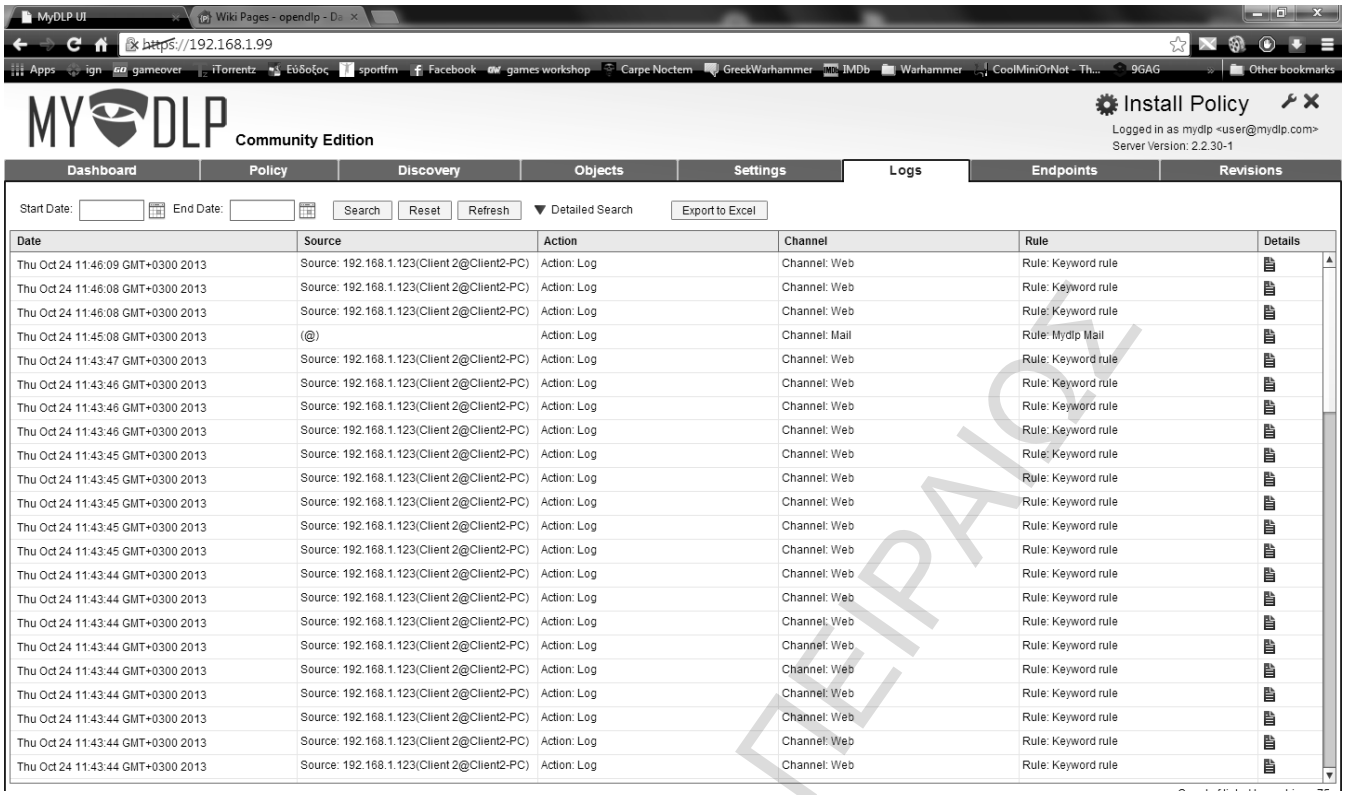


Figure 13: Logs Tab

3.2.4 MyDLP Block Request Example

An example of how MyDLP deals with user actions that meet a web rule with the "Block" action can be seen in the following figure (Figure 14). Further details about the reasons of the limitations can be provided by clicking the underlined "blocked" weblink.

Figure 14: Access Denied by MyDLP

4 Designing a free DLP solution using OpenDLP and MyDLP

As already stated in the Objectives section, found in the first chapter, the main aim of this thesis is to design a solid and open DLP solution by combining, in the most effective way, the two previously presented open DLP tools: MyDLP and OpenDLP. The combination of these tools is done in a way such that they complement each other, counterbalancing their weaknesses.

In order to better understand how these tools can be combined, it is necessary to re-examine which sectors of the three primary states of data (Chapter 2.6: Data at Rest, Data in Motion, Data in Use)each of them can cover.

At a first glance, MyDLP seems to be a more notable and complete solution, providing answers for all three data states. However, due to the limitations present in the Community Edition, it has been measured and found wanting. The free version of MyDLP does an excellent job coming up against unauthorized network traffic and limiting the non-authorized use of flash drives and printers, but when it comes to Endpoint protection it faces many restrictions and its weakness is evident, due to the lack of content discovery. It also supports only systems running Windows OS.

On the other hand, OpenDLP, as previously stated, only deals with one area of potential data loss: the Endpoint.

Therefore, the proposed DLP system, employs OpenDLP at first stage, as Endpoint Protection, utilizing its content discovery capabilities to identify what data needs protection and where it is located. Consecutively, (only for systems using Windows OS) any sensitive data found is carried forward to MyDLP server where it is parsed and appropriate policies are created to cover the needs for data in motion and data in use.

It is also recommended that client machines use static IP addresses and more importantly, MyDLP and OpenDLP are hosted in two different, independent machines in order to avoid having a single point of failure. A simple image of such a system infrastructure is presented below (Figure 15).

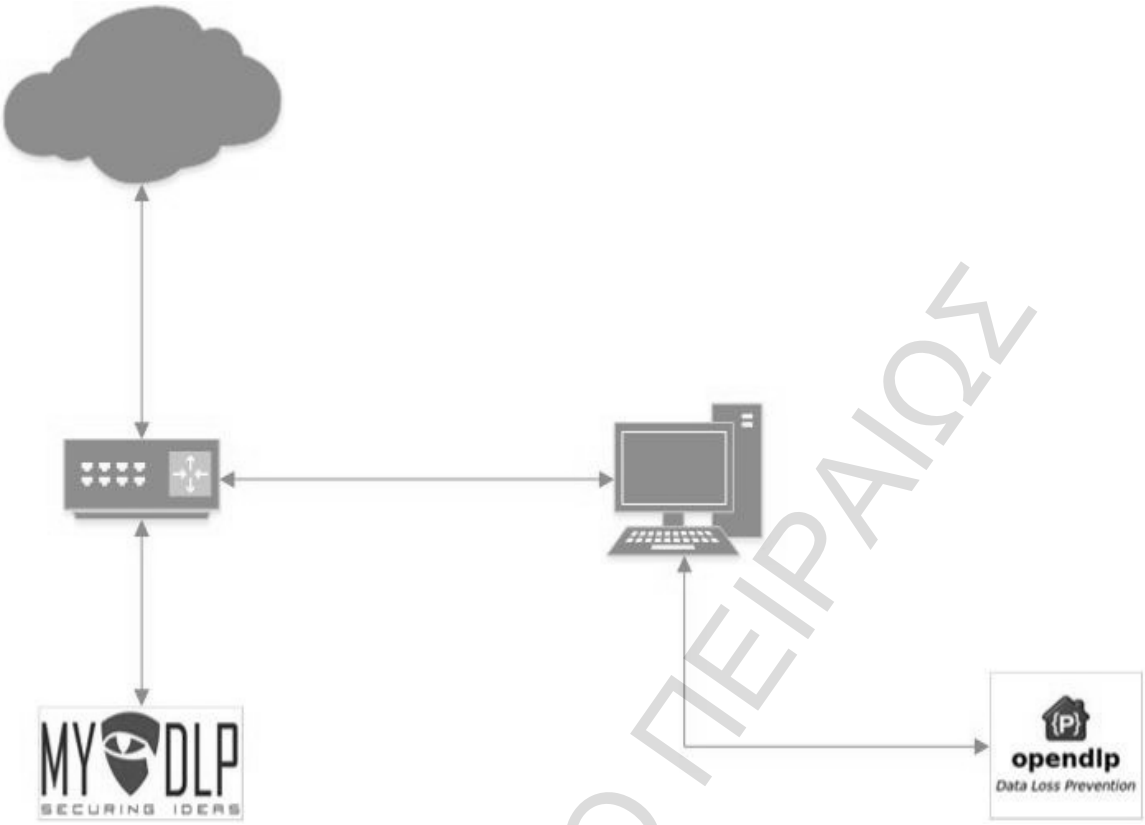


Figure 15: MyDLP- OpenDLP combination

4.1 Human factor – The weak link

Apart from the limitations and weaknesses present in the way the DLP tools function, there is another important obstacle that needs to be surpassed. Human error and negligence has to be minimized or reduced as much as possible. Even the best protection mechanisms fail and become useless if they are not properly operated, configured and maintained.

Regarding the proposed system above, it is obvious that there is a constant need for human presence and interference, for monitoring the state of the system, initiating scans, checking results and updating security policies. Neglecting to perform a sensitive data scan to the systems under protection using OpenDLP, results to inability to locate what data should be protected and where they are stored. Consequently, no rules for that specific data can be created by MyDLP. The existence of more generic rules alone, in MyDLP, would certainly limit the impact, but it would also reduce the level of protection that can be achieved, if exact rules were used.

4.2 The Need for Automation

The answer to the previously mentioned problem is presented in three steps:

1) Automate the scan initiation procedure in OpenDLP, 2) create and automate a mechanism to compare OpenDLP's scan results. Any possible changes detected in the sensitive data of a system are pointed out to the system administrator. 3) Automate the rules creation procedure in MyDLP, so that no constant human intervention is needed.

To achieve procedures' automation, several scripts that use the programming language Python, have been written and are listed in the Appendices. The main idea behind the DLP services automation is having the scripts run periodically, using some kind of event scheduling mechanism.

For this thesis, the automation scripts are stored in each DLP Server and are executed at regular time intervals using Linux's job scheduler Cron [23].

Cron example: run script.py, every day of the week, every month, at 2 am

```
0 2 * * * /home/test/script.py
```

```
# * * * * *  command to execute
# ┬ ┬ ┬ ┬ ┬
# │ │ │ │ │
# │ │ │ │ └──── day of week (0 - 7) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same as 0)
# │ │ │ └────── month (1 - 12)
# │ │ └──────── day of month (1 - 31)
# │ └────────── hour (0 - 23)
# └──────────── min (0 - 59)
```

Figure 16: Cron job scheduler basic syntax [23]

It is very important to point out that the objective of this automation is not to set the web interfaces (this counts for both web UIs) inactive and useless. Therefore, calling and executing "raw" scripts of corresponding services (e.g. script responsible for scan execution) through the DLP servers is not an option. What the scripts do is summon "phantom users" to navigate through the web interfaces with no human interaction needed. In other words, the scripts mimic the behavior of a real user when using the web consoles.

4.2.1 OpenDLP Automation

Automation in OpenDLP was possible by using a popular web testing tool, called Selenium [24] [25] and specifically the Selenium Webdriver Application Programming Interface (API) [26].

4.2.1.1 Selenium

Selenium is a browser automation tool. It can be controlled by many programming languages and supports a large number of web browsers (e.g. Firefox, Chrome, PhantomJS etc). Selenium navigates through webpages interacting with the HTML elements of the webpage.

Figure 17: Selenium Logo [24]

By running the appropriate script[3], a browser window spawns (testing purposes only), executing the exact same actions that a human user would. The automated "phantom user" logs into OpenDLP's management console and initiates a new scan based on already saved profiles (Chapter 3.1.1 – Workflows), providing also all the necessary information needed (IP address of system, list of IP addresses, or network Shares). When the scan is complete the results are exported in XML format, always automatically and are saved in the specified directory.

For testing purposes alone, the script in "Appendix A" uses Mozzila Firefox as its web browser. In a real environment, where the Linux server runs the script and no Graphical User Interface (GUI) is available, Firefox Webdriver is to be replaced with the headless [27] version of Firefox browser [28], or even better with PhantomJS [29] webdriver.

[3] Script for OpenDLP's automation can be found in Appendix A

4.2.2 Scan Comparison Automation[4]

After the scans results have been exported and saved, a mechanism is initiated, in order to check if changes were detected in the sensitive data of the scanned systems and report them to the administrator of the system. Changes detection is based on a series of comparisons, between key elements of successive scan results (every time, the current scan is compared to the immediately preceding one). For this project scans are considered to run a daily basis.

As previously mentioned, scans are saved in XML format. XML documents form a tree structure [30] that starts at "the root" and branches to "the leaves". Scan results are presented as "leaves", or "children" of the XML tree. An example of an exported OpenDLP XML scan file is shown in Figure18. Each branch represents a different scan result. Result details are included in each branch's children.

Figure 18: OpenDLP scan XML tree

The exported XML documents are parsed using the lxml etree [31]module. Four (4) basic result elements are extracted and used as criteria for the aforementioned comparisons. These elements are:

[4] Script for Scans Comparison can be found in Appendix B

- Type of detected data (e.g. Email, Visa, Master Card numbers etc)
- Pattern of data (e.g. 0510510510510510 for Master Card numbers)
- File containing scanned data.

(e.g. smb://192.168.1.64/c$/Users/Administrator/Documents/MasterCard.txt)

- MD5 [32]checksum value. (e.g. e2740266aab85558996a9a87fc561c0e)

In order to present results in a more comprehensible format, detected data are sent to the system admin structured as an array. Such an example is presented(Figure 19).

OpenDLP scan 2014-01-30

type	filtered_pattern	file	md5
Mastercard	5105105105105100	smb://192.168.1.64/c$/Users/Administrator/Documents/MasterCard.txt	e2740266aab85558996a9a87fc561c0e
Discover	6011111111111117	smb://192.168.1.64/c$/Users/Administrator/Documents/Discover.txt	bbc0ba70efd34473f1519050b3c0cab5
AMEX	378282246310005	smb://192.168.1.64/c$/Users/Administrator/Documents/American.txt	a5d172d46fc909f47e337374892330e8
E-mail	jimmynat0r@hotmail.com	smb://192.168.1.64/c$/Users/Administrator/Documents/New Rich Text Document.rtf	2b20acd6b2717edfbb869b87f818d9e1
E-mail	jkweston@microsoft.com	smb://192.168.1.64/c$/Windows/ehome/en-US/epgtos.txt	04e6dea4847cf9e2b6420833534ad19d
E-mail	jimmynat0r@hotmail.com	smb://192.168.1.64/c$/Users/Administrator/Documents/email.txt	b0d930a1f1b84f96860bcb893b8d859e
E-mail	budhajeewa@gmail.com	smb://192.168.1.64/c$/Program Files/7-Zip/Lang/si.txt	a13d1406d26dde62c4fb361a06a92ae3
E-mail	gomikochar@yahoo.com	smb://192.168.1.64/c$/Program Files/7-Zip/Lang/pa-in.txt	a946ddab1a37e43d6d28f6f43a5a1320
E-mail	drka2003@mail.ru	smb://192.168.1.64/c$/Program Files/7-Zip/Lang/be.txt	09f545e8f0567fe3600a4f70f25c036a

9 entries were found

Figure 19: Detected Data

Results comparison is based primarily on the md5 values collected for each data. As a secondary comparison criterion, the filename of the file containing the data is used.

According to the above, when comparing the file and md5 values found in the preceding scan to the ones found in the current scan, if the filename and md5 values don't exist in the current scan's results, it is stated that the file containing the specific detected data has been deleted. Alternatively, if the filename and the md5 value are both found in the results of the current scan it is stated that the file remains unchanged. Finally, if the file exists in the results of the current scan, but the md5 value has changed, the file is noted as being modified. Regarding the detection of new data, saved in already existing, or newly created files, the filename and pattern values of the current scan file are compared to the ones of the preceding scan file. Any new data entries or files detected are pointed out.

When the comparison of scans is finished, a report is sent to the administrator of the system, listing any findings. An example of such a report is presented in Figure 20.

```
 LIST OF CHANGES

1 file smb://192.168.1.64/c$/Users/Administrator/Documents/MasterCard.txt -----> has not been changed
2 file smb://192.168.1.64/c$/Users/Administrator/Documents/Discover.txt -----> has not been changed
3 file smb://192.168.1.64/c$/Users/Administrator/Documents/American.txt -----> has not been changed
4 file smb://192.168.1.64/c$/Users/Administrator/Documents/New Rich Text Document.rtf -----> has not been changed
5 file smb://192.168.1.64/c$/Windows/ehome/en-US/epgtos.txt -----> has not been changed
6 file smb://192.168.1.64/c$/Users/Administrator/Documents/email.txt -----> has not been changed
7 file smb://192.168.1.64/c$/Program Files/7-Zip/Lang/si.txt -----> has not been changed
8 file smb://192.168.1.64/c$/Program Files/7-Zip/Lang/pa-in.txt -----> HAS BEEN DELETED
9 file smb://192.168.1.64/c$/Program Files/7-Zip/Lang/be.txt -----> HAS BEEN MODIFIED

NEW ENTRIES

New file created: ['smb://192.168.1.64/c$/Windows/winsxs/x86_microsoft-windows-ehome-epgtos.resources_31bf3856ad364e35_6.1.7600.16385_en-us_cd9872fe4248fb0d/epgtos.txt']
New file created: ['smb://192.168.1.64/c$/Program Files/7-Zip/Lang/pa-int.txt']
New entry: gomikochar!!@yahoo.com
New entry: drka200345@mail.ru
```

Figure 20: Scan Comparison Report

4.2.3 MyDLP Automation

The final step of the proposed automation operation is automating the rules creation procedure in MyDLP, based on the findings of OpenDLP content discovery.

Unlike the automation technique used for OpenDLP's web interface (using Selenium Webdriver), MyDLP's automation is completely different, due to its webpage's architecture. MyDLP's web console is based on Apache Flex [33] [34], formerly known as Adobe Flex. Flex is a software development kit (SDK) for the development and deployment of cross-platform rich Internet applications based on the Adobe Flash [35].

The problem presented by the use of a Flash application, is that the Selenium Webdriver is ineffective, since there is no way to navigate through the webpage's HTML source code. Furthermore, disassembling the application and extracting the HTML code is not an easy task and the results are unreliable.

The approach used to cope with the above hindrance is a bit unconventional. Automation is achieved by using image recognition techniques to identify and control GUI components. The tool used is called Sikuli [36] and it is mainly used when there is no easy access to a GUI's internal or source code.

4.2.3.1 Sikuli

Sikuli is an open source tool to automate and test GUI using image recognition technology. It identifies existence of particular areas of a user interface, like a button, based on the fact that it looks like a certain image and then performs the user defined set of actions like clicking on it. It not only allows the user to control actions which take place in a browser like Selenium/Webdriver, but can also be used to control other desktop

applications. Sikuli Script is a visual scripting API using Jython [37]. Sikuli includes an integrated development environment (IDE) for writing visual Sikuli scripts with screen-shots [38] It can be also used with other IDEs that support the Jython language (e.g. Netbeans, Eclipse). Scripts written in Sikuli can be exported as executable files (Sikuli executable files use the .skl extension) and be called from other scripts.

The script provided in Appendix C is an example of MyDLP's automation using Sikuli. The script is written in Jython using Sikuli's IDE. When executed, it reads the sensitive data that were detected by OpenDLP's scan and then uses MyDLP's web console to create a custom user object containing the above findings. Moreover, it creates and installs a removable storage rule (Chapter 3.2.1 – MyDLP Policy Managent), configured to block the transfer of the previously detected data, to removable storage drives.

4.2.3.2 Sikuli's Limitation

Due to the image recognition technology that Sikuli uses, the script has to be run by systems that support a graphical environment. Since MyDLP runs on Ubuntu Server which doesn't support a graphical interface, the script has to be executed remotely and have its results displayed to another machine.

5 Conclusion

It's clear after all, that a data loss incident can have a significant impact in a company's proper operation and profitability. Inevitably, customers can lose confidence to the company, which means lost business. Data can leave the corporate network through a number of exit points. To mitigate the risk of data loss, companies should look for a DLP solution that monitors and blocks content distribution effectively without limiting functionality.

The number of Data Loss Prevention solutions, that are available commercially, increases day by day. However, the cost of most tools is very high and for many organizations the available options are prohibitive in terms of price.

The DLP system proposed in this thesis is based entirely on free products. Furthermore, the proposed and described automation plan, not only increases the system's capabilities, but aims to limit and reduce, human error and negligence, which constitute the greatest points of failure of such a system. However, restrictions still exist and the proposed DLP system is far from perfect. As mentioned before, OpenDLP is

defeated when facing encrypted data and MyDLP's automation is difficult, due to its web console's architecture. Sikuli doesn't completely deal with this problem, as Linux servers don't support a graphical interface.

As a final note, there is always room for improvement, but the proposed system can provide a satisfactory level of protection by combining the strengths of each tool, for no cost.

6 Bibliography

[1] Shenick Network Systems Limited, «Test Data Loss Prevention systems with diversifEye™,» Shenick Network Systems Limited, 2010.

[2] ISACA, "Data Leak Prevention," ISACA, 2010.

[3] Prathaben Kanagasingham, Sans Insitute, "Data Loss Prevention," Sans Insitute, 2008.

[4] T. Torsteinbø, Data Loss Prevention Systems and Their Weaknesses, University of Agder, 2012.

[5] Securosis, L.L.C, "Understanding and Selecting a Data Loss Prevention Solution," Securosis, 2010.

[6] Techopedia, "What is an Insider Attack – Definition from Techopedia," [Online]. Available: http://www.techopedia.com/definition/26217/insider-attack.

[7] B. Ruppert, «Protecting Against Insider Attacks,» SANS Institute, 2009.

[8] U. T.Mattson, How to Prevent Internal and External Attacks on Data - Securing the Enterprise Data Flow Against Advanced Attacks, 2008.

[9] "GNU General Public License," [Online]. Available: https://www.gnu.org/copyleft/gpl.html.

[10] "Microsoft .Net Framework," [Online]. Available: http://www.microsoft.com/net.

[11] "Perl Compatible Regular Expressions," [Online]. Available: http://www.pcre.org/.

[12] "Network Basic Input/Output System," [Online]. Available: http://en.wikipedia.org/wiki/NetBIOS.

[13] "Server Message Block," [Online]. Available: http://en.wikipedia.org/wiki/Server_Message_Block.

[14] "Meterpreter," [Online]. Available: http://www.offensive-security.com/metasploit-unleashed/About_Meterpreter.

[15] "OpenDLP," [Online]. Available: https://code.google.com/p/opendlp/.

[16] "OpenDLP: Data loss prevention tool," 3 May 2010. [Online]. Available: http://www.net-security.org/secworld.php?id=9226.

[17] University of Maine System, "Data Loss Prevention using OpenDLP," [Online]. Available: http://www2.maine.edu/pdf/DataLossPreventionusingOpenDLP.pdf.

[18] "MyDLP," [Online]. Available: http://www.mydlp.com/why-mydlp/.

[19] R. K, "Open Source DLP – Data Leak/Loss Prevention Application: MyDLP," [Online]. Available: http://www.excitingip.com/3950/open-source-dlp-data-leakloss-prevention-application-mydlp/.

[20] "Active Directory," [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/aa746492(v=vs.85).aspx.

[21] "Syslog," [Online]. Available: http://en.wikipedia.org/wiki/Syslog.

[22] MyDLP, *MyDLP Administration Guide, Version 2.0,* MyDLP, 2012.

[23] "Cron scheduler," [Online]. Available: http://en.wikipedia.org/wiki/Cron, https://wiki.archlinux.org/index.php/cron.

[24] "Selenium HQ," [Online]. Available: http://docs.seleniumhq.org/.

[25] "Python Selenium," [Online]. Available: https://pypi.python.org/pypi/selenium.

[26] "Application Programming Interface," [Online]. Available: http://en.wikipedia.org/wiki/Application_programming_interface.

[27] "What is a headless browser," [Online]. Available: http://blog.arhg.net/2009/10/what-is-headless-browser.html.

[28] "Installing headless Firefox in Ubuntu Server," [Online]. Available: http://www.installationpage.com/selenium/how-to-run-selenium-headless-firefox-in-ubuntu/.

[29] "PhantomJS," [Online]. Available: http://phantomjs.org/.

[30] "XML tree," [Online]. Available: http://www.w3schools.com/xml/xml_tree.asp.

[31] "Processing HTML and XML in Python," [Online]. Available: http://lxml.de/.

[32] "MD5 value," [Online]. Available: http://en.wikipedia.org/wiki/MD5.

[33] "Flex," [Online]. Available: http://flex.apache.org/.

[34] "Apache Flex," [Online]. Available: http://en.wikipedia.org/wiki/Apache_Flex.

[35] "Flash," [Online]. Available: http://en.wikipedia.org/wiki/Adobe_Flash.

[36] "Sikuli," [Online]. Available: http://www.sikuli.org/.

[37] "Jython," [Online]. Available: http://www.jython.org/.

[38] "SIKULI – Visual Technology to automate and test GUIs," [Online]. Available: http://xebee.xebia.in/index.php/2012/05/23/sikuli-visual-technology-to-automate-and-test-guis/.

7 Appendices

7.1 Appendix A - OpenDLP Automation

```
__author__ = 'Dimitris Koutsourelis'
import datetime
from datetime import timedelta
from selenium import webdriver
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.keys import Keys
import time
import os

#create firefox profile to use for file download without pop-up window
profile = webdriver.FirefoxProfile()
profile.set_preference('browser.download.folderList', 2) #save location of a file
profile.set_preference('browser.download.dir', '/Path/of/file/directory/')
#directory to save
file
profile.set_preference('browser.download.manager.showWhenStarting', False)
profile.set_preference('browser.helperApps.neverAsk.saveToDisk', 'text/xml') #instant s
ave to disk, no pop-up window when saving xml document files
driver = webdriver.Firefox(profile)


#  daily scan counter is kept in a text file
with open("/Path/of/counter/textfile/directory/", "r+") as readmyfile:
    counter = readmyfile.read()

#counter check, weekly reset and scan delete
if int(counter) == 0:
    driver.get('https://dlpuser:OpenDLP@192.168.1.98/OpenDLP/deletescan.html')
    scan_delete_list = driver.find_elements_by_xpath("//input[@type='checkbox'][@nam
e='scanname']")
    for i in scan_delete_list:
        scan_delete = driver.find_element_by_xpath("//input[@type='checkbox'][@name='
scanname']")
        scan_delete.click()
        scan_delete_confirm = driver.find_element_by_xpath("//input[@type='submit'][@
value='Delete Scans']")
        scan_delete_confirm.click()
        driver.get('https://dlpuser:OpenDLP@192.168.1.98/OpenDLP/deletescan.html')
    counter = 7

#send authentication credentials while fetching scan service
driver.get('https://dlpuser:OpenDLP@192.168.1.98/OpenDLP/startscan.html')

#fill scan name
scanname = driver.find_element_by_name("scanname")
scanname.send_keys("daily scan %s" % (counter))

```

```
#choose scan profile to use
scan_profile = Select(driver.find_element_by_name('profile'))
scan_profile.select_by_value("Agentless scan Windows")

#choose targets
systems = driver.find_element_by_name("systems")
systems.send_keys("IP Address of target system", Keys.TAB, Keys.ENTER)
driver.get(('https://192.168.1.98/OpenDLP/viewresults.html'))
time.sleep(300)

#export scan results
driver.get('https://192.168.1.98/OpenDLP/exportscan.html')
radioscanselect = driver.find_element_by_xpath("//input[@type='radio'][@name='scan name']")
radioscanselect.click()
radioscanselect.click()
export_scan = driver.find_element_by_xpath("//input[@type='submit'][@value='Export Scan Details']")
export_scan.click()
driver.implicitly_wait(30)

#daily scan counter increased and written in text file
writemyfile = open("/Path/of/counter/textfile/directory/", "w")
counter = int(counter) - 1
counter = str(counter)
writemyfile.write(counter)

writemyfile.close()
driver.close()

# add scan date to exported scan results
dt = datetime.date.today()
yesterday = dt - timedelta(1)
os.chdir(('/Path/of/file/directory/'))

for filename in os.listdir(os.getcwd()):
    if filename.startswith('OpenDLP-daily'):
        os.rename(filename, "OpenDLP scan" + " " + "%s" % (dt) + ".xml")
```

7.2 Appendix B – Scans Comparison

```
1.  _author__ = 'Dimitris Koutsourelis'
2.  import datetime
3.  from datetime import timedelta
4.  from lxml import etree
5.  import os
6.  from tabulate import tabulate
7.  import smtplib
8.  from email.MIMEMultipart import MIMEMultipart
9.  from email.MIMEBase import MIMEBase
10. from email.MIMEText import MIMEText
11.
12. dt = datetime.date.today()     # set today's date
13. yesterday = dt - timedelta(1)   # set yesterday's date
14. os.chdir('/Path/of/scan/files/directory/')
15. scan1 = etree.parse("OpenDLP scan" + " " + "%s" % (yesterday) + ".xml") # read and anal
yze today's and yesterday's scan files,
16. scan2 = etree.parse("OpenDLP scan" + " " + "%s" % (dt) + ".xml")       # exported as xml fi
les
17.
18. root1 = scan1.getroot()
19. systems_xml1 = scan1.find('systems')
20. system_xml1 = scan1.find('.//system')
21. results_xml1 = scan1.find('.//results')
22. result_xml1 = scan1.findall(
23.     './/result') # find all yesterday's scan results elements, which are children of "result" el
ement
24.
25. # a list consisting of scan results for yesterday's scan is created. The values that are save
d are:
26. matrix1 = [(child.find('.//type').text, # type of object found e.g email, visa, mastercard et
c
27.             child.find('.//filtered_pattern').text,
28.             # pattern of object found e.g if type is email, filtered pattern is mail@mail.com
29.             child.find('.//file').text, #file were objects were found.
30.             child.find('.//md5').text) for child in result_xml1] # md5 value of object
31.
32. root2 = scan2.getroot()
33. systems_xml2 = scan2.find('systems')
34. system_xml2 = scan2.find('.//system')
35. results_xml2 = scan2.find('.//results')
36. result_xml2 = scan2.findall('.//result') #find all today's scan result elements.
37.
38. #list consisting of scan results for today's scan is created!
39. matrix2 = [(child.find('.//type').text,
40.             child.find('.//filtered_pattern').text,
41.             child.find('.//file').text,
42.             child.find('.//md5').text) for child in result_xml2]
43.
```

```
44.with open("/Path/of/comparison/scan/files/directory/" + "%s" % (yesterday) + "vs" + "%
s" % (dt) + ".txt", "w") as comparison_write:
45.    comparison_write.write("OpenDLP scan" + " " + "%s" % (yesterday))
46.    comparison_write.write("\n")
47.    comparison_write.write(tabulate(matrix1, ["type", "filtered_pattern", "file", "md5"]))
48.    comparison_write.write("\n\n")
49.    comparison_write.write(str(len(matrix1)) + " entries were found")
50.    comparison_write.write("\n\n\n")
51.    comparison_write.write("OpenDLP scan" + " " + "%s" % (dt))
52.    comparison_write.write("\n")
53.    comparison_write.write(tabulate(matrix2, ["type", "filtered_pattern", "file", "md5"]))
54.    comparison_write.write("\n\n")
55.    comparison_write.write(str(len(matrix2)) + " entries were found")
56.    comparison_write.write("\n\n\n")
57.    comparison_write.write("----------------------------------------------------------------------------------
--------"
58.                           " \n\n LIST OF CHANGES")
59.    comparison_write.write("\n\n")
60.    comparison_write.close()
61.
62.comparison_result = str()
63.for i in range(len(
64.        matrix1)):#results comparison based primarilly on md5 values. Secondary comparis
on criterion is filenames
65.    for j in range(len(matrix2)):
66.        if matrix1[i][2] != matrix2[j][2] and matrix1[i][3] != matrix2[j][3]:   #if the filename a
nd md5 value don't exist in today's scan results, file has been deleted.
67.            comparison_result = ("file " + matrix1[i][2] + " -----> HAS BEEN DELETED")
68.        elif matrix1[i][3] == matrix2[j][3] and matrix1[i][2] == matrix2[j][2]: #if filename and
md5 value is found in today's scan results, the file is unchanged.
69.            comparison_result = ("file " + matrix1[i][2] + " -----> has not been changed")
70.            break
71.        elif matrix1[i][2] == matrix2[j][2] and matrix1[i][3] != matrix2[j][3]: #if filename exist
s in today's scan results, but md5 value has changed, file has been modified.
72.            comparison_result = ("file " + matrix1[i][2] + " -----> HAS BEEN MODIFIED")
73.            break
74.    with open("/Path/of/scan/comparison/files/directory/" + "%s" % (yesterday) + "vs" + "
%s" % (dt) + ".txt", mode='a') as comparison_write:
75.        comparison_write.write(str(i + 1) + " " + comparison_result + "\n")
76.    comparison_write.close()
77.
78.with open("/Path/of/scan/comparison/files/directory/" + "%s" % (yesterday) + "vs" + "%
s" % (dt) + ".txt", mode='a') as comparison_write:
79.    comparison_write.write("\n" + "NEW ENTRIES" + "\n\n")
80.    comparison_write.close()
81.
82.comparison_result2 = []
83.pattern_comparison_result = str()
84.for j in range(len(matrix2)):
85.    diff_filenames = []
86.    matrix1_filenames = [(matrix1[i][2]) for i in range(len(matrix1))]
```

```
87.  diff_patterns = []
88.  matrix1_patterns = [(matrix1[i][1]) for i in range(len(matrix1))]
89.  for i in range(len(matrix1_filenames)):
90.    if matrix2[j][2] == matrix1_filenames[i]:
91.      diff_filenames = [matrix1_filenames[i]]
92.  if matrix2[j][2] in diff_filenames:
93.    pass
94.  else:
95.    comparison_result2 = [matrix2[j][2]]
96.    print comparison_result2
97.    with open("/Path/of/scan/comparison/files/directory/" + "%s" % (yesterday) + "vs"
+ "%s" % (dt) + ".txt", mode='a') as comparison_write:
98.      comparison_write.write("New file created: " + str(comparison_result2) + "\n")
99.for j in range(len(matrix2)):
100.          for i in range(len(matrix1_patterns)):
101.            if matrix2[j][1] == matrix1_patterns[i]:
102.              diff_patterns = [matrix1_patterns[i]]
103.          if matrix2[j][1] in diff_patterns:
104.            pass
105.          elif matrix2[j][1] not in diff_patterns:
106.            pattern_comparison_result = matrix2[j][1]
107.            print pattern_comparison_result
108.
109.            with open("/Path/of/scan/comparison/files/directory/" + "%s" % (yesterday
) + "vs" + "%s" % (dt) + ".txt", mode='a') as comparison_write:
110.              comparison_write.write("New entry: " + pattern_comparison_result + "\n
")
111.            comparison_write.close()
112.
113.      gmail_user = "something@gmail.com"
114.      gmail_pwd = "password"
115.      to = "somethignelse@gmail.com"
116.
117.      def mail(to, subject, text, attach):
118.        msg = MIMEMultipart()
119.        msg['From'] = gmail_user
120.        msg['To'] = to
121.        msg['Subject'] = subject
122.
123.        mailServer = smtplib.SMTP("smtp.gmail.com:587")
124.        mailServer.ehlo()
125.        mailServer.starttls()
126.        mailServer.ehlo()
127.        mailServer.login(gmail_user, gmail_pwd)
128.        mailServer.sendmail(gmail_user, to, msg.as_string())
129.
130.        mailServer.close()
131.
132.      mail(to, "Today's scan results", " ", attach=
"C:\Users\Dimitris\Desktop\scan temp\OpenDLP comparison " + "%s" % (yesterday) + "vs" + "%s
" % (dt) + ".txt")
```

7.3 Appendix C – Sikuli Script

```
1. from sikuli import *
2. import datetime
3. from datetime import timedelta
4. import os
5.
6. dt = datetime.date.today()
7. yesterday = dt - timedelta(1)
8.
9. daily_comparison = open("/Path/of/scan/comparison/files/directory/" + "%s" % (yesterd
ay) + "vs" + "%s" % (dt) + ".txt")
10.lines = daily_comparison.readlines()
11.for i in range(len(lines)):
12.   if lines[i].startswith("New entry: "):
13.       entry_line = lines[i].strip("New entry: ")
14.       with open("/Path/of/data/to/be/imported/to/MyDLP/directory/MyDLP_keywords.t
xt",'a+') as daily_comparison_keywords:
15.           daily_comparison_keywords.write(entry_line)
16.daily_comparison_keywords.close
17.
18.click(Pattern("1391121903448.png").similar(0.72))
19.click(Pattern("FiKeywordGro.png").targetOffset(-18,-2))
20.click(Pattern("ViKeywordGro.png").targetOffset(148,0))
21.wait("EditKeywordG.png",60)
22.click(Pattern("EditKeywordG.png").targetOffset(10,-94))
23.type("%s" % (dt) + "Keywords")
24.click(Pattern("EditKeywordG.png").targetOffset(173,-13))
25.wait("EditKeywordI-1.png",60)
26.click(Pattern("EditKeywordI-1.png").targetOffset(-72,-24))
27.wait("EditKeywordI-2.png",60)
28.click(Pattern("EditKeywordI-2.png").targetOffset(-6,4))
29.click("EditKeywordI-1.png")
30.wait("Filename.png",60)
31.click("Filename.png")
32.click("Filename.png")
33.type("/Path/of/data/to/be/imported/to/MyDLP/directory/MyDLP_keywords.txt")
34.type(Key.ENTER)
35.wait("SaveCancel.png",120)
36.click(Pattern("SaveCancel.png").targetOffset(-35,1))
37.wait("SaveCancel-1.png",60)
38.click(Pattern("SaveCancel-1.png").targetOffset(-28,5))
39.click(Pattern("1390965309907.png").similar(0.87).targetOffset(0,-2))
40.click("UserDehnecl.png")
41.click(Pattern("UserDehned.png").targetOffset(148,0))
42.wait("CreateNewIte.png",60)
43.click(Pattern("CreateNewIte.png").targetOffset(-25,48))
44.wait("EditDialogNa.png",60)
45.click("Ilame.png")
46.type("%s" % (dt) + "Keywords")
47.click(Pattern("EditDialogNa.png").targetOffset(173,83))
```

```
wait("MatcherEditD.png",60)
doubleClick(Pattern("MatcherEditD.png").targetOffset(-57,-15))
type("Keyword Group")
type(Key.ENTER)
wait("MatcherEditD-1.png",60)
click(Pattern("MatcherEditD-1.png").targetOffset(22,-13))
type("%s" % (dt) + "Keywords")
type(Key.ENTER)
click(Pattern("Matchwholewo.png").targetOffset(-77,-46))
click(Pattern("Matchwholewo-1.png").targetOffset(-81,-8))
click(Pattern("SaveCancel-2.png").targetOffset(-36,7))
click(Pattern("SaveCancel-3.png").targetOffset(-36,11))

if exists("1391116734420.png",60):
    click("1391116734420.png")
elif exists(Pattern("AddRul.png").similar(0.82).targetOffset(-58,1),30):
    click(Pattern("AddRul.png").similar(0.82).targetOffset(-58,1))
    if exists(Pattern("add.png").targetOffset(-11,0),30):
        wait(Pattern("add.png").targetOffset(-11,0))
        click(Pattern("add-1.png").targetOffset(-13,1))
wait("CreateNewRul.png", 60)
click(Pattern("QRemovableSt.png").targetOffset(-74,0))
wait("Name.png", 60)
type("Name.png","Removable Storage rules for " + "%s" % (dt))
click("1391116196313.png")
wait("RemovableSto.png")
click(Pattern("RemovableSto.png").targetOffset(344,3))
click(Pattern("ActionPass.png").targetOffset(-70,16))
click(Pattern("PassPassLogB.png").targetOffset(-51,31))
click("UserDeHned-2.png")
click(Pattern("UserDehned-1.png").targetOffset(144,-1))
wait("CreateNewIte-1.png")
click(Pattern("CreateNewIte-1.png").targetOffset(-17,71))
wait("EditDialogNa-1.png")
click("EditDialogNa-1.png")
type("%s" % (dt) + "Keywords")
wait("SaveCancel.png",120)
click(Pattern("SaveCancel.png").targetOffset(-35,1))
wait("Q30JanKeywor.png")
click("Q30JanKeywor.png")
click(Pattern("fi1TQ30JanKe.png").targetOffset(109,2))
wait("EditDialogIt.png")
click(Pattern("EditDialogIt.png").targetOffset(173,83))
doubleClick(Pattern("MatcherEditD.png").targetOffset(-57,-15))
type("Keyword Group")
type(Key.ENTER)
wait("MatcherEditD-1.png",60)
click(Pattern("MatcherEditD-1.png").targetOffset(22,-13))
type("%s" % (dt) + "Keywords")
type(Key.ENTER)
click(Pattern("Matchwholewo.png").targetOffset(-77,-46))
```

```
click(Pattern("Matchwholewo-1.png").targetOffset(-81,-8))
        click(Pattern("SaveCancel-2.png").targetOffset(-36,7))
        wait("Name.png", 60)
        click(Pattern("Name.png").targetOffset(43,-6))
        type("%s" % (dt) + "Keywords")
        click(Pattern("SaveCancel-3.png").targetOffset(-36,11))
        dragDrop("Q30JanKeywor-1.png", "N0informatio.png")
        wait("RemovableSto-1.png",120)
        click(Pattern("rInstallPoli.png").targetOffset(-72,-1))
        os.remove("/Path/of/data/to/be/imported/to/MyDLP/directory/MyDLP_keywords.txt")
```