

Lecture 30: Mounting Targeted Attacks with Trojans and Social Engineering — Cyber Espionage

Lecture Notes on “Computer and Network Security”

by Avi Kak (kak@purdue.edu)

April 20, 2016

12:23am

©2016 Avinash Kak, Purdue University



Goals:

- Can a well-engineered network be broken into?
- Socially engineered email lures
- Trojans and the gh0stRAT trojan
- Cyber espionage
- Exploiting browser vulnerabilities

CONTENTS

	<i>Section Title</i>	<i>Page</i>
30.1	Is It Possible to Break into a Well-Engineered Network?	3
30.2	Trojans	8
30.3	The gh0stRAT Trojan	14
30.4	Cyber Espionage	22
30.5	Cyber Espionage Through Browser Vulnerabilities	28

30.1: IS IT POSSIBLE TO BREAK INTO A WELL-ENGINEERED NETWORK?

- Consider an agent **X** who is determined to break into a network with the intention of stealing valuable documents belonging to an organization and for the purpose of conducting general espionage on the activities of the organization.
- Assume that the targeted organization is vigilant about keeping up to date with the patches and with anti-virus software updates (Lecture 22). We also assume that the organization's network operates behind a well-designed firewall (Lectures 18 and 19). Additionally, we assume that the organization hires a security company to periodically carry out vulnerability scans and for penetration testing of all its computers (Lecture 23).
- We further assume that the computers in the targeted organization's network are not vulnerable to either the dictionary or the rainbow-table attacks (Lecture 24).
- In addition, we assume that **X** is physically based in a different country, which is not the same country where the organization's

network is. Therefore, it is not possible for **X** to gain a James Bond like physical entry into the organization's premises and install a packet sniffer in its LAN.

- **Given the assumptions listed above, it would seem that the organization's network cannot be broken into. But that turns out not to be the case. Any network, no matter how secure it is from a purely engineering perspective, can be compromised through what is now commonly referred to as "social engineering."**
- Here is a commonly used exploit to compromise an otherwise secure network through social engineering:
 - Let's assume that an individual named **Bob** Clueless is a high official in an organization named **A** in the US and that this organization manufactures night-vision goggles for the military. Pretend that there is a country **C** out there that is barred from importing military hardware, including night-vision goggles, from the US. So this country decides to steal the design documents stored in the computers of the organization **A**. Since this country does not want to become implicated in cross-border theft, it outsources the job to a local hacker named **X**, who is obviously promised a handsome reward by a quasi-government organization in **C**. **C** supplies **X** with all kinds of information (generated by its embassy in the US)

regarding **A**, its suppliers base, the cost structure of its products, and so on. On the basis of all this information, **X** sends the following email to Bob Clueless:

To: Bob Clueless
From: Joe Smoothseller
Subject: Lower cost light amplifier units

Dear Bob,

We are a low-cost manufacturer of light-amplifier units. Our costs are low because we pay next to nothing to our workers. (Our workers do not seem to mind --- but that's another story.)

The reason for writing to you is to explore the possibility of us becoming your main supplier for the light amplification unit.

The attached document shows the pricing for the different types of light-amplification units we make.

Please let me know soon if you would be interested in our light amplifier units.

Attachment: light-amplifiers.doc

- When Bob Clueless received the above email, he was already under a great deal of stress because his company had recently

lost significant market share in night-vision goggles to a competing firm. Therefore, no sooner did Bob receive the above email than he clicked on the attachment. What Bob did not realize was that his clicking on the attachment caused the execution of a small binary file that was embedded in the attachment. This resulted in Bob's computer downloading the client `gh0st` that is a part of the `gh0stRAT` trojan.

- Subsequently, **X** had full access to the computer owned by Bob Clueless.

[As is now told, **X** used Bob's computer to infiltrate into the rest of the network belonging to the organization — this was the easiest part of the exploit since the other computers trusted Bob's computer. It is further told that, for cheap laughs, **X** would occasionally turn on the camera and the microphone in Bob's laptop and catch Bob picking his nose and making other bodily sounds in the privacy of his office.]

- I would now like to present a summary of the different steps/facets of a classic social engineering attack. This listing is taken from <http://www.f-secure.com/weblog/archives/00001638.html>:

1. You receive a spoofed e-mail with an attachment
2. The e-mail appears to come from someone you know
3. The contents make sense and talk about real things (and in your language)
4. The attachment is a PDF, DOC, PPT or XLS

5. When you open up the attachment, you get a document on your screen that makes sense, but you also get exploited at the same time
6. The exploit drops a hidden remote access trojan, typically a Poison Ivy or Gh0st Rat variant
7. You are the only one in your organization who receives such an email
8. You work for a government, a defense contractor or an NGO

30.2: TROJANS

- From the standpoint of the programming involved, there is not a whole lot of difference between a bot and a trojan. We talked about bots in Lecture 29. [The word “trojan” that you see here is all lowercase. However, in the literature, you are more likely to see the word as “Trojan” or “Trojan Horse” — after the Trojan Horse from the Greek epic “The Aeneid.” But as this word is acquiring a currency of its own in computer security circles, I think, sooner or later, it will become a more generic noun and that the security folks will refer to the malware simply as a “trojan.”]
- The main difference between a trojan and a bot relates to how they are packaged for delivery to an unsuspecting computer. There could be a certain randomness to how a bot hops from machine to machine in a network. For example, as you saw with the `AbraWorm.pl` worm in Lecture 22, a bot may simply choose to scan a random set of IP addresses each day and, when it finds a machine with a certain vulnerability, it may install a copy of itself on that machine.
- On the other hand, a trojan is intended for a more targeted attempt at breaking into a specific machine or a specific set of machines in a network.

- Also, a trojan may be embedded in a piece of code that actually does something useful, but that, at the same time, also does things that are malicious. So an unsuspecting person may never realize that every time he/she is clicking on an application, in addition to producing the desired results, his/her computer may also be engaged in harmful activities.
- It is sobering to realize that email attachments and other applications (that one typically finds on the desktop of a run-of-the-mill computer today) are **not** the only hosts for trojans. *As we describe below, trojans may also come buried in what is downloaded for the updating of the more system-oriented software in your computer.*
- The CERT advisory, whose first page is shown on page 12, mentions a version of the `util-linux` package of essential linux utilities that had a trojan embedded in it; this corrupted package was inserted into the archive `util-linux-2.9g.tar.gz`. The archive was placed on at least one official FTP server for Linux distribution at some point between January 22 and 24, 1999. It is possible that this corrupted archive was distributed to other mirror sites dedicated to the distribution of the Linux operating system. *[As the CERT advisory mentions, this specific trojan consisted of a modification to the `/bin/login` file that is used for logging in users. The trojan code would send email to, presumably, the intruders, providing them with information related to the user logging in, etc.]* The full text of the advisory is available at <http://www.cert.org/advisories/CA-1999-02.html>.

- The same CERT advisory also talks about messages of the following sort that were emailed to a large group of recipients in January 1999:

Date:
From: "Microsoft Internet Explorer Support" IESupport@microsoft.com
To:
Subject: Please upgrade your Internet Explorer

Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052


As a user of the Microsoft Internet Explorer, Microsoft Corporation provides you with this upgrade for your web browser. It will fix some bugs found in your Internet Explorer. To install the upgrade, please save the attached file ie0199.exe in some folder and run it.

For more information, please visit our web site at www.microsoft.com/ie/

As you can see, this spam message is written to look like it came directly from Microsoft to your computer. As you would infer on the basis of what was presented in the previous section, **this email is a classic example of using social engineering to break into a machine.** According to the information posted at <http://www.f-secure.com/v-descs/antibtc.shtml>, when the trojan ie0199.exe that came with the email messages was run, it extracted two files from its body: mprexe.dll and sndvol.exe. The trojan then registered the dll with the Windows registry so that it would be run at every reboot of the machine. When the dll was run, it executed the sndvol.exe file, which caused the infected machine to contact one of the following Bulgarian web sites: <http://www.btc.bg>, <http://www.infotel.bg>, and <http://ns.infotel.bg>.

CERT Advisory CA-1999-02 Trojan Horses



<http://www.cert.org/advisories/CA-1999-02.html>



search

GO customize Publications Catalog

Software Assurance Secure Systems Organizational Security Coordinated Response Training

[Publications Catalog](#)
[Historical Documents](#)
[Authorized Users of "CERT"](#)
[US-CERT Vulnerability Notes Database](#)
[Vulnerability Disclosure Policy](#)
[Courses](#)



CERT[®] Advisory CA-1999-02 Trojan Horses

Original issue date: February 5, 1999
Last revised: March 8, 1999
Minor typographical corrections

A complete revision history is at the end of this file.

Systems Affected

Any system can be affected by Trojan horses.

Overview

Over the past few weeks, we have received an increase in the number of incident reports related to Trojan horses. This advisory includes descriptions of some of those incidents ([Section II](#)), some general information about Trojan horses ([Sections I and V](#)), and advice for system and network administrators, end users, software developers, and distributors ([Section III](#)).

Few software developers and distributors provide a strong means of authentication for software products. We encourage all software developers and distributors to do so. This means that until strong authentication of software is widely available, the problem of Trojan horses will persist. In the meantime, users and administrators are strongly encouraged to be aware of the risks as described in this document.

I. Description

A Trojan horse is an "apparently useful program containing hidden functions that can exploit the privileges of the user [running the program], with a resulting security threat. A Trojan horse does things that the program user did not intend" [[Summers](#)].

Trojan horses rely on users to install them, or they can be installed by intruders who have gained unauthorized access by other means. Then, an intruder attempting to subvert a system using a Trojan horse relies on other users running the Trojan horse to be successful.

II. Recent Incidents

Incidents involving Trojan horses include the following:

False Upgrade to Internet Explorer

Recent reports indicate wide distribution of an email message which claims to be a free upgrade to the Microsoft Internet Explorer web browser. However, we have confirmed with Microsoft that they do not provide patches or upgrades via electronic mail, although they do distribute security bulletins by electronic mail.

- The lessons to be learned from the above CERT are:
 - Unless you are using a respected package manager such as the Synaptic Package Manager to install software updates, make sure that you are downloading the software from a trusted source, that it has a digital signature obtained through a cryptographically secure algorithm, and that you can verify the digital signature of the software you are downloading. *When trojans are embedded in system files, the file size is often left unchanged so as to not arouse suspicion. So the only way to verify that a file was not tampered with is through its digital signature.*
 - Validating the digital signature should involve also validating the public key of the signer.
 - **Never, never click on an email attachment if you are not absolutely sure that the message is authentic — even if it looks authentic.** If you were not expecting the sort of message you are looking at (even if it appears to be from someone you know), it is best to not open the attachment without establishing the provenance of the message. In most of our day-to-day interactions, this is not a problem since the context of our interaction with the others immediately establishes the authenticity of the email.
- To further underscore the role played by socially engineered email (especially those emails that include attachments containing mal-

ware) in infiltrating networks, here is a quote from the abstract of a recent report by Nagaraja and Anderson from the University of Cambridge (a detailed reference to this report is given in Section 30.4):

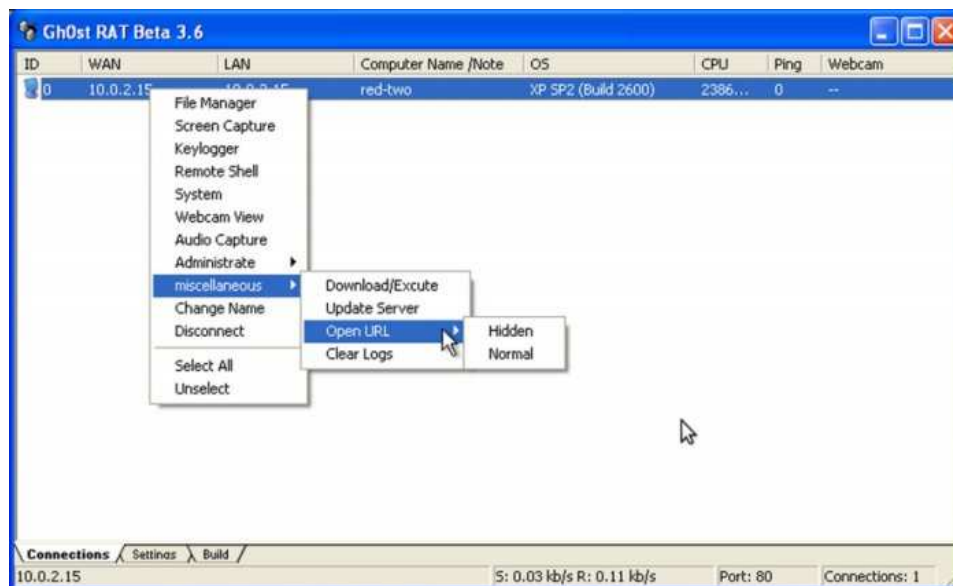
“This combination of well-written malware with **well-designed email lures**, which we call social malware, **is devastatingly effective**. The traditional defense against social malware in government agencies involves expensive and intrusive measures that range from mandatory access controls to tiresome operational security procedures. **These will not be sustainable in the economy as a whole**. Evolving practical low-cost defenses against social-malware attacks will be a real challenge.”

30.3: THE gh0stRAT TROJAN

- This is probably the most potent trojan that is currently in the news. That is not surprising since when a machine is successfully compromised with this trojan, **the attackers can gain total control of the machine, even turn on its camera and microphone remotely and capture all the keyboard and mouse events.** In addition to being able to run any program on the infected machine, **the attackers can thus listen in on the conversations taking place in the vicinity of the infected machine and watch what is going on in front of the computer.**
- The trojan, intended for Windows machines, appears to be the main such trojan that is employed today for cyber espionage.
- The “RAT” portion of the name “gh0stRAT” stands for “Remote Administration Tool”.
- An attacker controls such a trojan with a “RAT Management Tool” that consists of a graphical user interface (GUI) on which the attacker can see the Windows registry of the infected machine,

the currently running processes, the list of installed applications, the current network connections, etc. The GUI also has graphical controls that the attacker can use to turn on and off the camera and the microphone on the infected machine, to send corrupted emails to those whose addresses can be found in the infected machine, to capture keyboard events, etc.

- Shown in the figure below is an example of the GUI of the RAT management tool that goes with the `gh0stRAT` trojan. As the reader can see, the drop-down menu displayed includes buttons for controlling the camera, the microphone, etc., on the infected machine.



- Variants of this trojan allow the attackers to plug in their own additional features for further customizing its behavior.

- The `gh0stRAT` trojan was written originally by the hackers in China. So the original code has its comments and other embedded documentation (which are important to understanding code) mostly in Chinese. But now some open-source folks claim to have translated it into English — meaning that they claim to have translated the comment lines and the other documentation into English. [However, you will notice that that is not entirely the case. Much of the documentation that is included in the files is still in Chinese.] You can download the latest “English version” as an archive called `gh0st3.6_src.zip` from the URL

`http://www.opensc.ws/c-c/3462-gh0st-rat-3-6-source-code.html`

One of the coders at this web site says that “ *This is very poorly coded and most of it looks ripped. Anyhow, I tested it out on Vista and it compiled fine using MSVC++ with the Platform SDK. It has minor warnings but all functions still work properly.*”

- Just to give the reader a sense of the scope of `gh0stRAT`, shown on the next four pages is an indented listing of the subdirectories and the files in the source code directory for `gh0st3.6`. [At some point in the future, I plan to add to my description of the functionality of some of the more significant files in the directory tree — assuming it can be done at all.]
- The brief comments that follow the file names in the directory listing on the next several pages are just pure guesses on my part at this time — not at all to be taken too seriously. I hope to

refine my understanding of the code at some point in the near future.

- A compilation of this source code will give you a **Server** that an attacker can use to monitor the trojan on an infected machine. The trojan itself is compiled as the executable **gh0st**.
- Here is a listing of the files:

```

gh0st3.6_src/
  gh0st.dsw
  gh0st.ncb
  gh0st.opt

  Server/
    install/
      ReadMe.txt
      install.aps
      install.rc
      install.plg
      install.dsp
      acl.h
      RegEditEx.h
      resource.h
      StdAfx.h
      decode.h
      install.cpp
      StdAfx.cpp          => for including the precompiled header stdafx.h
      res/
        svchost.dll      => a well-known trojan module for remote access
                          (Note that this is not the same as svchost.exe
                           that is so basic to the operation of the Windows
                           platform. See Lecture 22.)

    svchost/
      ReadMe.txt
      svchost.plg
      svchost.aps
      svchost.rc
      svchost.dsp
      resource.h
      ClientSocket.h
      hidelibrary.h
      ClientSocket.cpp
      StdAfx.cpp
      svchost.cpp
      common/
        filemanager.h    => for file ops such saving, loading, moving, etc.
        KeyboardManager.h => for storing keystrokes, etc.
        AudioManager.h   => for recording microphone inputs from the trojan

```

```

hidelibrary.h    => for making folders invisible to a user
login.h
ScreenManager.h => header needed for the control GUI
until.h
inject.h
loop.h
Buffer.h
ScreenSpy.h     => for monitoring the screen of an infected machine
VideoCap.h      => for capturing camera config and for remote video capture
decode.h
install.h
Manager.h
ShellManager.h
VideoManager.h  => for capturing camera config and for remote video capture
Dialupass.h
KernelManager.h
RegEditEx.h     => sets and reads registry permissions (header)
resetssdt.h
SystemManager.h
AudioManager.cpp => for recording microphone inputs from the trojan
ScreenManager.cpp => needed for the control GUI
until.cpp
Buffer.cpp
ScreenSpy.cpp
VideoCap.cpp    => for capturing video remotely
install.cpp
Manager.cpp
ShellManager.cpp
VideoManager.cpp => for capturing video remotely
Dialupass.cpp   => for viewing passwords used for dialup
KernelManager.cpp => makes calls to cKernelManager for multithreading
SystemManager.cpp
RegEditEx.cpp   => sets and reads registry permissions
FileManager.cpp
KeyboardManager.cpp

sys/
makefile
RESSDT.c
RESSDT.sys
sources

gh0st/
ReadMe.txt
gh0st.clw       => contains info for the MFC class wizard
gh0st.plg       => compilation build log file
removejunk.bat
gh0st.rc
gh0st.aps
BuildView.h
KeyBoardDlg.h  => header for capturing keystrokes
StdAfx.h
AudioDlg.h     => for recording microphone inputs
MainFrm.h
SystemDlg.h
BmpToAvi.h
gh0stDoc.h
TabSDIFrameWnd.h
Resource.h
ThemeUtil.h
gh0st.h
Tmschema.h
ScreenSpyDlg.h => header for screen capture
CustomTabCtrl.h

```

```

ghOstView.h
TrayIcon.h
encode.h
SettingsView.h
TrueColorToolBar.h
FileManagerDlg.h      => header for file operations
IniFile.h
SEU_QQwry.h
WebCamDlg.h           => header for camera image capture
FileTransferModeDlg.h
InputDlg.h
ShellDlg.h
AudioDlg.cpp          => for microphone capture
ghOst.cpp
MainFrm.cpp
SystemDlg.cpp
BmpToAvi.cpp          => for format conversion
ghOstDoc.cpp
TrueColorToolBar.cpp
TabSDIFrameWnd.cpp
BuildView.cpp
ghOst.dsp
ThemeUtil.cpp
IniFile.cpp
FileManagerDlg.cpp     => for file ops such as saving, moving, etc.
ScreenSpyDlg.cpp      => for screen capture
CustomTabCtrl.cpp
ghOstView.cpp
TrayIcon.cpp
SettingsView.cpp
WebCamDlg.cpp         => for camera capture
SEU_QQwry.cpp
FileTransferModeDlg.cpp
InputDlg.cpp
ShellDlg.cpp
KeyboardDlg.cpp       => header for capturing keystrokes
StdAfx.cpp            => for including precompiled Windows headers
include/
    Buffer.h
    CpuUsage.h
    IOCPServer.h
    Mapper.h
    Buffer.cpp
    CpuUsage.cpp
    IOCPServer.cpp
control/
    BtnST.h
    HoverEdit.h
    WinXPButtonST.h
    BtnST.cpp
    HoverEdit.cpp
    WinXPButtonST.cpp
res/
    1.cur              => cur is an ico like format for cursors
    2.cur
    3.cur
    4.cur
    dot.cur
    Bitmap_4.bmp       => bitmapped image files
    Bitmap_5.bmp
    toolbar1.bmp
    toolbar2.bmp
    audio.ico
    ghOst.ico

```

```
cmdshell.ico          => ico is a format for icons
keyboard.ico
system.ico
webcam.ico
gh0st.rc2
install.exe
CJ60Lib/
overview.gif
Readme.htm
CJ60Lib/              => graphics extension library, originally by Code Jockey
    readme.txt
    CJ60lib.def
    resource.h
    Globals.h
    stdafx.h
    CJ60Lib.clw
    CJ60Lib.dsw
    CJ60Lib.ncb
    CJ60Lib.opt
    CJ60Lib.positions
    CJ60Lib.rc
    CJ60StaticLib.dsp
    CJCaption.cpp
    CJListCtrl.cpp
    CJToolBar.cpp
    CJ60lib.cpp
    CJControlBar.cpp
    CJListView.cpp
    CoolBar.cpp
    CJDockBar.cpp
    CJMDIFrameWnd.cpp
    CoolMenu.cpp
    CJ60Lib.dsp
    CJDockContext.cpp
    CJMiniDockFrameWnd.cpp
    FixTB.cpp
    ShellPidl.cpp
    CJExplorerBar.cpp
    CJOutlookBar.cpp
    FlatBar.cpp
    ShellTree.cpp
    CJFlatButton.cpp
    CJPagerCtrl.cpp
    Globals.cpp
    SHFileInfo.cpp
    CJFlatComboBox.cpp
    CJSearchEdit.cpp
    stdafx.cpp
    CJFlatHeaderCtrl.cpp
    CJSizeDockBar.cpp
    hyperlink.cpp
    CJFrameInfo.cpp
    CJTabctrlBar.cpp
    MenuBar.cpp
    Subclass.cpp
    CJFrameWnd.cpp
    res/
        btn_arro.bmp
        button_images.bmp
        btn_explorer.bmp
        cj_logo.bmp
        vsplitba.cur
        hsplitba.cur
        cj60lib.rc2
```

```
Include/
    CJ60Lib.h
    CJFlatComboBox.h
    CJMiniDockFrameWnd.h
    CJToolBar.h
    ModulVer.h
    CJCaption.h
    CJFlatHeaderCtrl.h
    CJOutlookBar.h
    CoolBar.h
    ShellPidl.h
    CJControlBar.h
    CJFrameInfo.h
    CJPagerCtrl.h
    CoolMenu.h
    ShellTree.h
    CJDockBar.h
    CJFrameWnd.h
    CJSearchEdit.h
    FixTB.h
    SHFileInfo.h
    CJDockContext.h
    CJListCtrl.h
    CJSizeDockBar.h
    FlatBar.h
    Subclass.h
    CJExplorerBar.h
    CJListView.h
    CJTabCtrlBar.h
    hyperlink.h
    CJFlatButton.h
    CJMDIFrameWnd.h
    CJTabView.h
    MenuBar.h
Lib/
    CJ60StaticLib.lib

common/
    Audio.h
    CursorInfo.h
    macros.h
    VideoCodec.h
    Audio.cpp
    zlib/
        zconf.h
        zlib.h
        zlib.lib
```

- You will find several other RATs at the following URL: <http://www.opensc.ws/trojan-malware-samples/>.

30.4: CYBER ESPIONAGE

- Suddenly everyone seems to be talking about cyber espionage. Much of the current attention on this subject is a result of the seminal work that has come out of a collaboration between the Citizens Lab, Munk Center for International Studies, University of Toronto, and the SecDev Group, a Canada-based consultancy house. This collaboration has produced the first two reports listed below. These reports make for a remarkable reading of the spy-thriller sort:
 - “Tracking GhostNet: Investigating a Cyber Espionage Network,” <http://www.scribd.com/doc/13731776/Tracking-GhostNet-Investigating-a-Cyber-Espionage-Network>
 - “Shadows in the Cloud: Investigating Cyber Espionage 2.0,” <http://www.infowar-monitor.net/2010/04/shadows-in-the-cloud-an-investigation-into-cyber-espionage-2-0>
 - “The Snooping Dragon: Social-Malware Surveillance of the Tibetan movement,” <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-746.html>

The third report mentioned above is from the University of Cambridge by two researchers, Shishir Nagaraja and Ross Anderson,

who also collaborated with the folks at the University of Toronto and the SecDev Group.

- If you do look through the reports listed above, seek answers to the following questions:
 1. At the lowest levels of data gathering, what information did the investigators collect and what tool(s) did they use for that purpose?
 2. How did they identify the malware (the trojan) present in the infected computers?
 3. How did the investigators track down the control and the command computers that the infected machines sent their information to?
 4. What was the capability of the specific trojan that played a large role in stealing information from the infected computers? How did this trojan allow the humans to control in real-time the infected machines?
 5. How did the investigators manage to spy on the spies?
 6. What can you infer from the source code for the trojan?
- The “Tracking Ghostnet” report, which came out in March 2009,

describes an espionage network that had infected at least 1295 computers in 103 countries, mostly for the purpose of spying on the various Tibetan organizations, especially the offices of the Dalai Lama in Dharamsala, India. The espionage network unearthed through the investigative work presented in this report is referred to as the “Ghostnet.”

- The “Shadows in the Cloud” report, released in April 2010, documents an extensive espionage network that successfully stole documents marked “SECRET,” “RESTRICTED,” and “CONFIDENTIAL” from various high offices of the Government of India, the Office of the Dalai Lama, the United Nations, etc. The espionage network unearthed through the investigative work presented in this report is referred to as the “Shadow.” The Shadow network is considered to be more sinister than the older Ghostnet network.
- The “Snooping Dragon” report is about the same attacks that are described in the “Tracking Ghostnet” report, but its overall conclusions are somewhat different. The “Snooping Dragon” report is more categorical about the origin of the attacks and who sponsored them.
- The primary mechanism for spreading malware in both Ghostnet and Shadow was targeted and socially-engineered email containing infected Word or PDF attachments.

- The attackers designated some of their own machines that were used to facilitate their exploits as “Control Servers” and some others as “Command Servers.” The trojan server we talked about in the previous section ran on the Control Servers. Such servers provided the attackers with GUI-based facilities — an example of which was shown earlier on page 15 — to watch and control the infected machines. The Command Servers, on the other hand, served mostly as repositories of malicious code. A human monitoring the trojan-server GUI on a Control Server could ask the trojan client on an infected machine to download a newer version of the malware from one of the Command Servers.
- The espionage attacks in both Ghostnet and Shadow used the **gh0stRAT** trojan as the main malware for spying. The trojan client in the Shadow network appears to have greater communication capabilities. In addition to communicating with the trojan servers running on the Control Servers, the Shadow trojan client could also receive commands directly through email and through certain social media.
- The trojan clients running on the infected machines communicated with their server counterparts running on the Control Servers using the HTTP protocol and using the standard HTTP port. This was done to disguise the trojan communications as ordinary HTTP web traffic. When a trojan client on an infected machine wanted to upload a document to a Control Server, it used the HTTP POST command. [Your web browser typically makes

an HTTP GET request when it wants to download a page from a web server. On the other hand, when your browser wants to upload to the web server a web form you may have filled out with, say, your credit card information, it sends to the server an HTTP POST ‘request’ that contains the information you entered in the form.]

- The HTTP requests sent by the trojan clients running on infected machines were typically for what seemed like JPEG image files. In actuality, these files contained further instructions for the trojans. That is, the trojan on an infected machine would send an HTTP GET request to a Control Server for a certain JPEG image file; in return, the Control Server would send back to the trojan the instructions regarding which Command Server to contact for possibly additional or newer malware.
- For the investigation reported in “Shadows in the Cloud,” the University of Toronto investigators used **DNS sinkholes** to good effect. A sinkhole is formed by re-registering a now-expired domain name that was programmed into an earlier version of a trojan as the destination to which the trojan should send its communications. Since the older versions of the trojans still lodged in the infected machines are likely to continue communicating with these now expired domain names, by re-registering such domains with new IP addresses, the investigators could pull to their own sites the HTTP traffic emanating from the older trojans. **What a cool trick!** If my understanding is correct, this is how the U. of Toronto folks got hold of the highly-classified documents that were exfiltrated by some of the trojans during the course of

the investigation reported in “Shadows in the Cloud.”

30.5: CYBER ESPIONAGE THROUGH BROWSER VULNERABILITIES

- The beginning of 2010 witnessed Google announcing that its computers had been compromised. Some news reports mentioned that Google's password/login system Gaia was targeted in these attacks. Supposedly, some or all of the source code was stolen. Again according to news accounts, some Gmail accounts were also compromised.
- It is believed that social engineering played a large role in how this attack was carried out. According to a report by John Markoff in the New York Times (April 19, 2010), the attack started with an instant message sent to a Google employee in China who was using Microsoft's Messenger program. By clicking on a link in the message, the Google employee's browser (Internet Explorer) connected with a malicious web site. This connection caused the Google employee's browser to download the Hydraq trojan (also referred to as the Aurora trojan) from the web site. That gave the intruders complete control over the Google employee's computer. The rest is history, as they say. [\[The backdoor to the attacked computer created by Hydraq is similar to what is achieved by the gh0stRAT trojan. However, the former is probably not as powerful with regard to its remote administration capabilities as the latter. An interesting difference](#)

between Hydraq and gh0stRAT is that the former uses port 443 to make connections with its command and control computers. As you may recall from Lecture 19, this port is used for the secure SSL-based HTTPS service for the delivery of web pages. However, the encryption algorithms used by Hydraq are not based on the SSL protocol; they are custom designed. We will not go any further into the Hydraq (or Aurora) trojan.]

- Context-relevant messages and email as lures to get users to click on malware-bearing attachments and URLs are probably the most common attack vectors used today that cause computers to download viruses, worms, and trojans. In addition to those attack vectors for delivering the Hydraq trojan, it is believed that the attack on Google also utilized a more specialized attack vector — a vulnerability in the older and unpatched versions of the Microsoft's Internet Explorer web browser. This vulnerability, presented earlier in Section 28.4 of Lecture 28, has to do with the allocation and deallocation of memory for HTML objects by JavaScript and the fact that JavaScript, like scripting languages in general, is not a strongly typed language.