Intrusion Agent

The Next Generation Of Spy

Frederic Charpentier – fcharpentier@laposte.net

[Abstract]

This document is about a new method to gain access to an internal network, even though they are shielded with firewall systems and proxy.

It discusses in details the design of a so-called Intrusion Agent program and the different ways to implement and diffuse it over computers networks in the purpose to break into private information systems and to steal confidential documents.

This whitepaper is intended to be read by security professionals and pentesters who work to secure their information system and who want a full overview of the risk's level in a secure network designed with the Industrial best practices.

This whitepaper is also intended to be read by program developers interested in viruses and worms' concepts and uses.

I hope you will find the Intrusion Agent concept interesting and eye-opening.

[Introduction]

They are many ways to gain access into a information system. The most used way is to find and exploit some front door's vulnerabilities and then try to gain an authorized access into the internal network .
Whitehats or Blackhats attackers usually try to break into front office systems, like portal web servers, ftps or transactional gateway servers.

To perform their attack, they often use the same process : information gathering, vulnerability seeking and, finally, exploitation of a vulnerability to obtain a remote shell access using techniques like buffer overflow, format string, boggus cgis or sql injection.

Then, the attackers will try to rebound into the internal network.

Many methods exist to secure networks and applications. Networks are secured by packet filters, proxy. Servers and applications are hardened and chrooted.

Nowadays, there still are vulnerable servers and programs, but the exploitation of their threats are more and more difficult for many reasons examined above.

In a penetration test or in a real attack, the final objective is to demonstrate that it is possible to corrupt information. By corrupting information, I mean steal, modify or erase business sensible information.

Hopefully for administrators, confidential information and documents don't reside on front office servers. These information are stocked in back office database, files servers and, sometimes, directly on workstation filesystem.
These confidential information are shielded behind many firewalls, proxy, virtual LAN and all kind of securing network components. Thus, network administrators are busy to keep these materials in a high level security, with regular update and Intrusion detection system.

Here is the problem. Attackers usually don't try to break your firewall and proxy systems.

Like the famous chinese strategist Sun Tzu wrote : "Attack him where he is unprepared, appear where you are not expected."

So, the question is : "Is it possible to get the control of a computer despite of its protection and even if it is not accessible from the outside ?".
This whitepaper will try to give an answer to this question and to prove that wherever a user is able to browse the web, even though he browses through proxy systems, an attacker can gain access to a computer.

Furthermore, we will consider a scenario showing how our intrusion agent would be dangerous for an organisation if used with a distributed architecture and with worm-like proliferation.

Index

[Current hacking methods]

The topic of this part is not to discuss all well-known methods used by hackers, but to show underlying philosophy and the pros and cons of these methods .

Buffer Overflow and format string

An attacker can use many ways to get connected and control a computer or a server . The most used principle is to exploit an unchecked buffer or user's input vulnerability on a remote daemon and to inject executable malicious code.

To discover vulnerable servers on a targeted network, attackers use well-known tools like port scanner, banner grabber, fingerprinter, vulnerability scanner…

Thus, they can find vulnerable listening daemon and launch some buffer overflow attacks against them to inject executable codes named "shellcodes" on the targeted computer. These shellcodes contain executable programs which open a stealthy port with a remote command shell on the remote machine.

Once the attacker gets a shell, he will try to obtain high privileges on the remote machine. Indeed, daemons and open applications often run under a low privileged user. To obtain high privileges, the attacker will use a so-called "local exploit". The principle of local exploits is the same as buffer overflows or format strings used against remote daemon. Here, the exploit is launched against a local program accessible from granted privileges of the attacker.
Using a local exploit, a remote attacker, who already got a remote shell under the "apache" user, can gain the "root" user privileges.

This kind of attacks are really efficient when launched against frontal servers settled into a company's demilitarised zone (dmz) . Usually, even if the attacker can entirely deface or corrupt a server, it will not be easy for him to jump into the company's internal network.

Methods used for these attacks (buffer overflow and format string) are so complex techniques that very few attacker are able to find and create the attack.

So, this philosophy of hacking consists in the fact that attacked machine are vulnerable to known threats and are available on a public network.

SQL Injection and parameters tampering

Nowadays, systems administrators are aware of security threats : they are reading security forums and regularly upgrade their dmz systems with great precautions.

Under these new conditions, the network's warfare is changing : smart attackers invent new methods to accomplish their tricks and attacks.
The targeted systems aren't frontal servers anymore. Attacks are now focused on the critical core business systems, where the valuable information is settled : internal servers and databases.

To break into the internal servers and databases protection, attackers will exploit, in a different way, vulnerabilities in the frontal Extranet servers.

The principle is to bypass web application forms inputs to inject unattended commands, like SQL commands, javascript, path to files....

Tampering eb parameters could lead to dump customer database or documents on backend computers. Indeed, Database servers are usually settled in the backoffice network, not accessible from the internet, and only a web application, like servlets or cgis, could request data on it.
Tampering web paramaters enables the attacker to use a frontal web server as a bouncer to the internal database or file servers.

We will not discuss these methods, the lector is invited to refer to XSS and SQL-Injection literature, but we will see that combined with an intrusion agent these techniques could lead to a global attack against a company information system.

XSS and session hijacking

Another great attacks family is the session hijacking.
Eventhough your network architecture is armored and your application fully patched, an attackers could still steal confidential data.

These attacks known as session hijacking and cross-site-scripting could allow an attacker to log on your system or web portal as an authenticated user and thus to gain access to information and control.

The main principle is to heist an authenticated established session from a user or an administrator. These attacks could seem complex because they usually involve third-part programs or users and some good conditions.

A technique used is the cross-site-scripting. The principle of this kind of attack is based on a url link send to a user. This user must be already connected and authenticated on the targeted portal. The sent link contains a boggus url with an embeded javascript. Clicking on this link, the user opens a new session on the targeted portal and executes the javascript in the authenticated context.
The javascript will grab some informations in the user's cookie and send these informations to the attacker. Receiving this cookie's information, the attacker can forge a request on the targeted portal as the real user.

So, the attacker has stolen the session of the authenticated user. The main principle here is to trick an already authenticated user to run a javascript on any responding system on the cookie domain (xxx.domain-victim.org). Responding systems are any network services, often web sites, which are able to echo a sent string with control on it.
If a system of the victim domain echoes a javascript code, then the user's browser will execute to javascript. The responding systems usually used are web "search" inputs, web server invalid url handlers or even smtp and echo services.

Other session hijacking attacks are based on the weakness of the session's conservation system. Many servers use systems like session-id to keep the authenticated context with the user. If this session-id is predictable or easy to brute-force, then an attacker can try all the possibilities to steal a running session.
This kind of attack could also be performed on a low level system like statefull firewall using systems as syn-cookies to keep context with a client.

We will not discuss all these methods, but it is important to keep in mind that even though your systems seem well hardened, they can still be vulnerable.

The common end goal

The common end goal of a real attack is to gain control over a system and to be able to come and come back again.
To allow this, the attacker will settle a backdoor program on the hacked system. Then, the attacker himself will patch the vulnerable system to keep other attackers out.

Another way to get rooted on a remote system with a backdoor is the email way. The attacker will use social-engineering tricks to convince a user to execute the attached malicious program.
The problem with such attack methods is that the attacker will not be able to connect directly to the program, because internal networks are filtered with firewalls and proxy servers.

So, this kind of malicious program can just embed viruses or “blind” worms. Their effect can be disastrous, but our objective is not to destroy a company local network, but to get a “stealth access” into it.

Stealth access into local network is the purpose of our Asymmetric Intrusion Agent : How an attacker can use internal backdoor to get connected anonymously from the outside.

[Asymmetric Intrusion Agent]

In this part we will describe our concept agent. First, we will explain the foundation concept of the agent. Then, we will go deeper and deeper into the conception and features of the agent.

The agent concept

The concept consists in a backdoor program running on a victim machine of an internal network which gives the attacker an interactive access from the outside.

The objective of this agent is to maintain, bypassing all security components, an attacker into the internal network and allow the attacker to exploit internal computer vulnerabilities which are more common than external vulnerabilities.

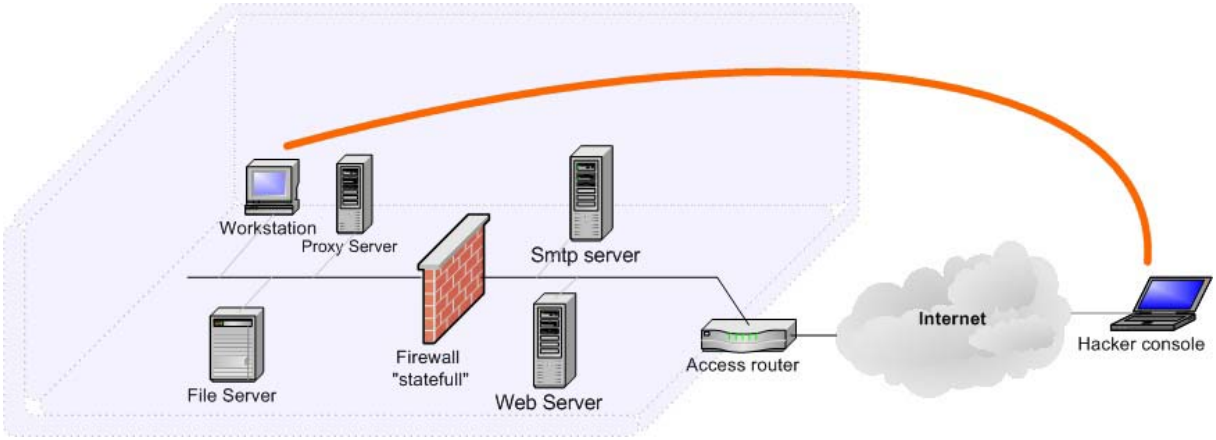
The agent is an executable program, which embeds an HTTP protocol, able to create a link from the inside to the outside. Indeed, the HTTP protocol, which is often used to browse the web, can permit us to encapsulate other kind of traffic than HTML data.

The agent works in pair with an external program called “the console”. The console is used by the attacker to get interactive control with the agent settled in the internal network.

The Asymmetric Intrusion Agent is a kind of reverse backdoor : it is the agent that establishes the connection with the outside console.

Agent and console work together to create a full-duplex tunnel between the victim machine and the attacker.

As depicted in the scheme 1, the console owns a direct access to a internal victim machine.



Scheme 1

A problem remains in the agent conception : the agent doesn’t know by advance in which environment it will be injected. Thus, the agent must be able to adapt to the local configuration. By example, the agent will have to discover the IP addressing plan, the address of proxy servers, etc, by itself.

The goal of the agent is to act like a real living organism : the agent grows up, evolves, mutates and spreads itself on other available networks from his birth location.

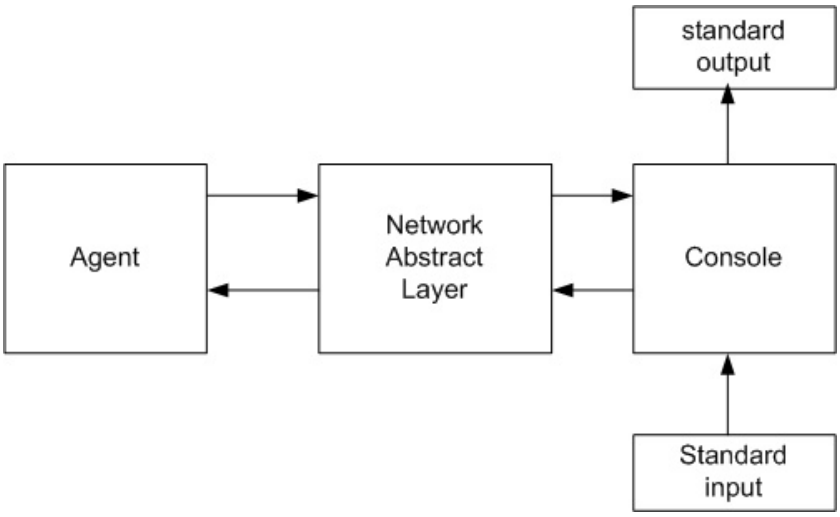
In the following parts , we will discuss the main design of this agent and how it will be able to act like a living organism.

Inputs and outputs

To describe the design of the agent, we will first explain the inputs and the outputs of the program.
As we said, the agent will chat with the remote console through a HTTP¨tunnel. The creation method of this tunnel will be dissected in the next section : Network Abstract Layer.

On the Scheme 2, we can see an overview of the agent's architecture : the agent, running on the victim machine, converses with the console through the Network Abstract Layer.

On the console, the attacker uses an interactive shell to control the agent.



Scheme 2

The attacker uses a socket output to send command to the agent. The agent executes the received command and sends the return on another socket connected to the console. Thus, the console displays the agent's response on the standard output.

All sockets are handled by an abstraction layer in charge of the HTTP protocol negotiation and exchanges.

Thus, the design of the agent enables the development of complex function over this layer, abstracting all HTTP and socket details.

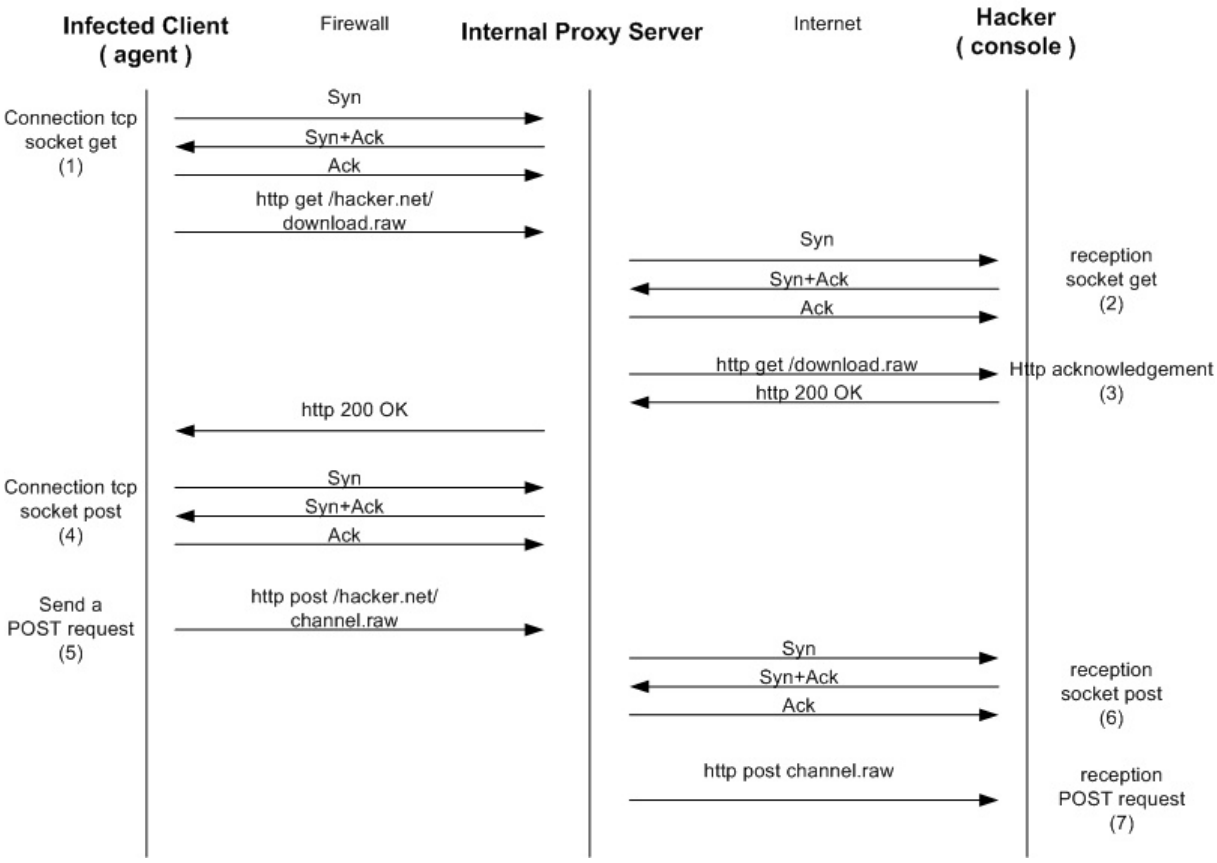
As we will see further, the agent will be able to run a program on the victim machine and connect the input and the output of this program with the input and output of the console through the abstract layer.

Another important point about the design, is the size of the agent : the smaller the agent, the faster its injection.
All complex tasks are transferred onto the console side. When the agent executes a function, each returned information is sent to the console as raw data. Then, the console performs any complex task (presentation, regular expression, checks, ..) for the end user.

Network Abstraction Layer

To describe in depth the design of the abstraction layer, we will now explain the protocol used by this network layer.
As we said, the agent will chat with the remote console through a HTTP¨tunnel.

To dissect the protocol, we will refer to the schema bellow.



The principle is to establish a TCP connection between the infected computer and the proxy server of the lan (1). The way the agent found the IP address of the proxy is described in the next section.
As soon as the TCP connection is ok, the agent sends a HTTP GET command. The GET command requests a file on the external hacker computer (the console). For anonymous reasons, the address of the hacker computer could be defined with a dyn-dns address (because the address is hard written in the agent executable code).

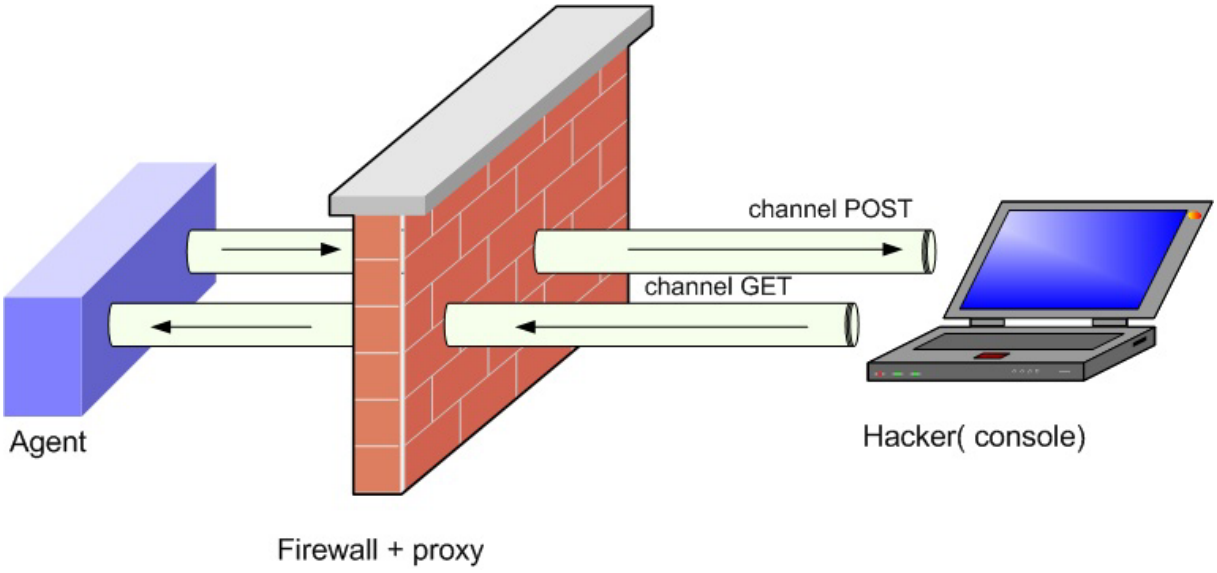
When the proxy receives the get request, it establishes a TCP connection with the console (2) and then transfers the GET command.

On the hacker side, the console gets a listening socket on port 80. When the console receives the GET command from the proxy server, it sends an acknowledgment packet (3).
The proxy receives the acknowledgment and transfers it to the agent.

So we have established an unidirectional tunnel between the console and the agent : the proxy will now transfer all data emanating from the console to the agent. It is our upload channel (GET Channel).
Then, we must create the download channel. Then we use another part of the HTTP protocol : the POST command.
The agent establishes another socket with the proxy server (4) and sends to it a HTTP POST command, requesting to upload a file on the hacker address.

The proxy server will create a socket with the console and forward the POST command. As described in the HTTP 1.1 RFC, the server does not need to acknowledge a POST command.

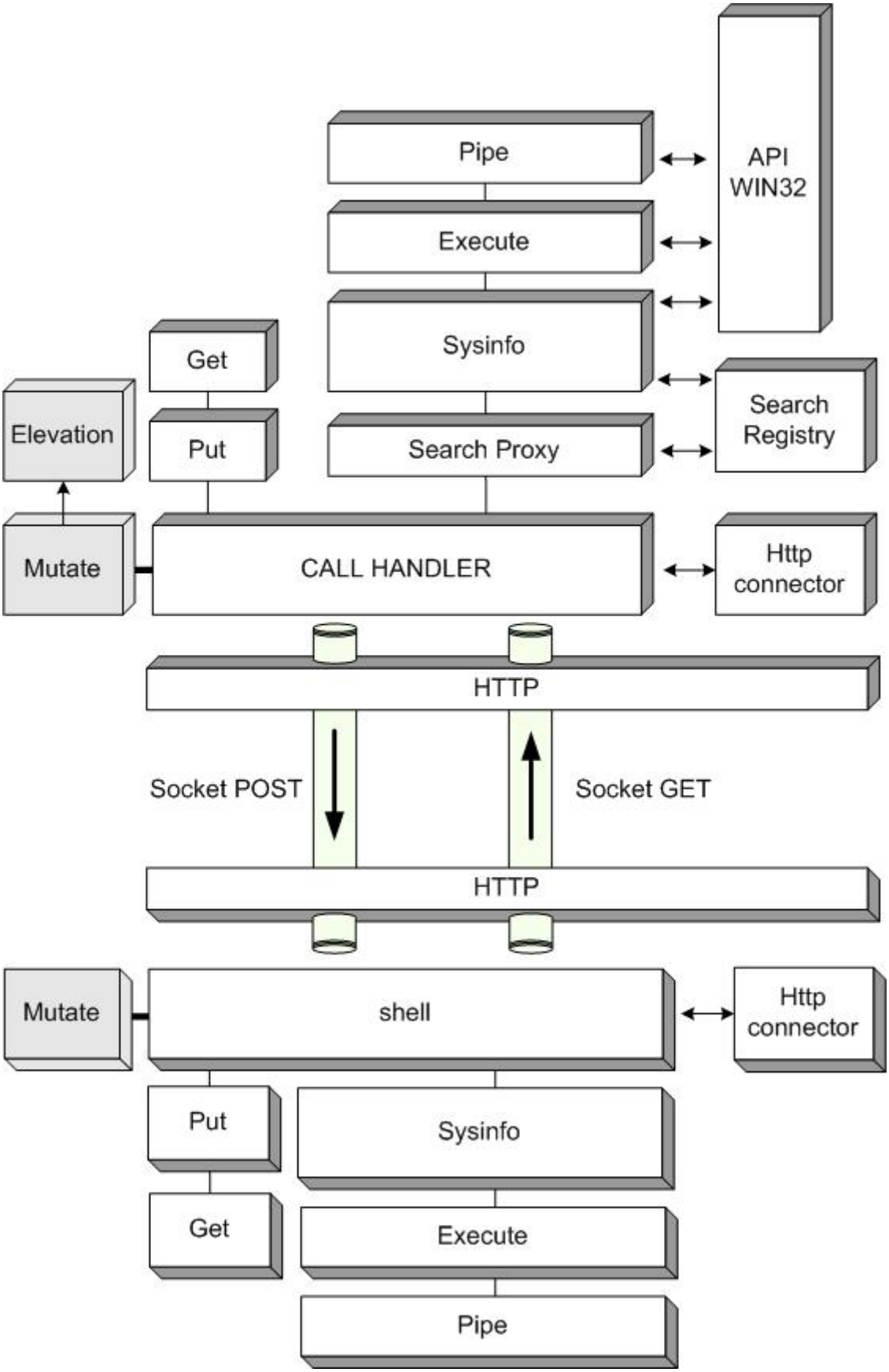
Then we can consider having created a download channel, and so we now have a full duplex channel between the agent and the console.



The http protocol design and specification

The agent and console design is based on a symmetric module conception. A module named the *call handler* handles the creation of the full duplex channel and gives an interactive shell to the end user.
The *call handler* uses two others modules : the *http connector* and the *watchdog*.

The schema depicted below shows the different modules of the console and the agent.



The *call handler* discusses through the network abstraction layer with the shell module of the console.
When the *call handler* wants to create or recreate the full duplex channel with the console, it calls the *http connector* module.

Here the specs of the *http connector* :

- All proxy servers must accept the HTTP protocol used by the agent.
- All proxy servers must identify the HTTP packet as non-cacheable.

- All request must be initiated from the agent.
- A full duplex connection must be maintained between the agent and the console.

Version

The HTTP protocol used by the agent must follow the protocol HTTP 1.0 specs. Thus, the agent will be able to work with any proxy/firewall architecture.

Identification

Any HTTP header used to create the channel must contains a "x-sessioncookie" field. Thus, the console will be able to handle multiple connection from different agents without ambiguity.

Caching

The console program must always respond with the field "Cache-Control : no-store".
The agent program must always respond with the field "pragma : no-cache".

Thus, the proxy will never cache HTTP traffic between the agent and the console.

Full duplex

When the console receives the POST command from the agent, it must always generate an "echo" command to check whether the agent is able to respond.
If not, the agent must restart the negotiation until it succeeds.

Request Identification

The agent must use the MIME type "application/tunnelled" for the fields "Content-Type" and "Accept".

Encoding

The POST channel traffic must be encoded with the Base64 algorithm. (RFC 2045 "Internet Message Bodies").
Otherwise, some proxy servers could interpret this traffic as mal-formed HTTP requests.

HTTP packet sample

GET Request

From Agent to the Console :

GET /download.raw HTTP/1.0
User-Agent: WORM (version beta)
x-sessioncookie: tD9hKgAAfB8ABCftAAAAAw
Accept: application/x-tunnelled
Pragma: no-cache
Cache-Control: no-cache

From the Console to the agent :

HTTP/1.0 200 OK
Server: CONSOLE/2.0 [v101] MacOSX
Connection: close
Date: Thu, 19 Aug 1982 18:30:00 GMT
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/x-tunnelled

POST Request

From the Agent to the console :

POST /upload.raw HTTP/1.0
User-Agent: WORM (version beta)
x-sessioncookie: tD9hKgAAfB8ABCftAAAAAw
Content-Type: application/x-tunnelled
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 32767
Expires: Sun, 9 Jan 1972 00:00:00 GMT

From the Console to the Agent :
No response is expected from the console ! The agent indefinitely keeps on sending data to the console.

Connectivity details and support

On each side, if a socket error is raised, the *http connector* must restart the negotiation.

Some proxy servers kill the communication after a while. To be sure that the link is still up, the console must periodically send an echo request.

The chosen value for the field “Content-Length” is 32767. Proxy servers accept this value as a maximum. Usually, most of the proxy servers ignore this value, and so they allow us to send an endless datastream through them (the POST channel).

The chosen date value used in the GET Channel acknowledgment pinpoints backwards to a date in the past (*Thu, 19 Aug 1982).* So, the proxy server will not conserve the datastream in a cache buffer.

The field « Expires » must be set with the same date.

The field « Pragma » must be set with the value « No cache » and the field “Cache-Control” with « No cache ».
That way, proxys using the HTTP 1.0 specs will not use cache buffers.

The call handler and intrinsic shell

As we depicted in the module schema, all communications are handled by a module called the *call handler*.

The function of this module is to create and maintain a full duplex communication between the agent and the console.

As we described above, the agent is always the one which starts the communication with the console.

To achieve its mission, the agent will use the services of two others modules : *http connector* and *proxy search*.
The function of the *proxy search* module is to determine whether the infected computer uses a proxy. If this computer seems to use a proxy, then the module must find its IP address and return it to the *call handler*.

To stay hidden in the targeted network, the *call handler* follows a simple connection-tempo algorithm.
So, if the agent cannot join immediatly its console, it will fall into a "sleep mode" waiting for the next try. The algorithm tries to follow the same behaviour as a web browser.

On the other side, the console shows a interactive shell to the user. In fact, the agent's call handler chats with the console's shell module.

To optimize the agent's size, all complex task are performed on the console side, by the "shell" module.

The modules

Here we will see the 'basic' modules implemented in the agent. These simple modules are the fundation of the complex functionality that will allow us to give some complex behaviour to our agent.

Registry search

The agent uses some Windows API functions to look for specific key and value into the registry. Depending on the agent's execution privilege, this function cand read and write the registry.

Proxy search

When the call handler module tries to establish a link with the console, this module is called. The *proxy search* module will seek the registry for proxy information.
Thus, the module will lookup whether the Internet Explorer, Netscape and Opera programs are installed and use proxy settings.

If not, the call handler will try to connect directly to the console.

If the agent still cannot join the the console, the agent will scan the local network, based on the local IP plan, trying to find common proxy (80, 3128 and 8080 ports).

Sysinfo

The agent can lookup the registry for the release version of the system. These informations will be used in the future to exploit specific vulnerabilities of the system.

Here are the basic informations that the agent can automatically return to the console :
Product Typpe, platform Id, osversion, major version, minor version, build, netname, ip, sysdir, logged users, windir.

Execute

The Execute module will run a specific local program. It uses the "ShellExecute" function of the WinApi.

Thus, the console can remotely launch any local program, with parameters, on the infected station.
This function could be useful to disable an antivirus programs or to lauchn a server program.

Pipe

The *Pipe* module will enable the console user to remotely run a local program in an interactive mode.

With the *Pipe*, the user will run a program (like ***cmd***, ***telnet*** or ***ftp***) in a shell mode.
To perform this, the agent will connect the input and the output of the piped program with the POST and GET socket.
Thus, the console user will use the piped program as if he was running it locally.

Put and Get

The Put and Get modules allow the console user to download files from the infected station or to upload files on it.
Thus executable programs can be uploaded and ran remotely from the console. This property will be really useful in the next section.

[Elevation and Mutation]

Here the real fun begins. We have seen above the design of a basic intrusion agent. During the introduction, I was explaining the concept of the Intrusion Agent and the possibility to give it the behaviour of a living virus.

Living viruses, like all organisms, can evoluate during their life. A virus can adapt itself to his environment : if the infected organism can resist to viruses, a complex virus can mutate and spread itself on other target.
Thus, from one simple virus and after multiple infections, we find many different aspects of the same virus : the mutants.

Mutation

Our agent tries to follow the same concept : once a computer is infected, the agent can mutate and thus be more useful for the console user.
To perform this, the agent will use a combination of the basic modules described before : the *put* and *execute* modules.

The schema below shows the process of the mutation.

The objective of the mutation is to enable the console user to inject a new agent with more useful functions. For example, a keystroker or a password sniffer can be added to the new agent.

Elevation

A problem appears in this scheme : the execution privileges. Indeed, when the new agent is downloaded and runs on the infected machine, the new agent process is still associated with the user's right.
To go further, the agent will implement a second concept : the elevation.

When the agent is injected, it runs with the user's right. To run other complex programs like servers, sniffers or scanners, the agent must obtain the root system privileges.
To obtain this, the agent will use security threats of the infected system. Indeed, the Windows system owns many security problems that will let the agent running out of the user's privileges.

The schema below shows the process of the mutation and the elevation.

To perform the elevation, the agent will use known local exploits of the operating system. In the actual state of art, the lector can refer to this two local exploits :

- DebPloit (Authentication Flaw in Windows Debugger can lead to Elevated Privileges)
- GetAd (NetDDE Exploit inWM_COPYDATA)

These exploits use flaws of the win32 platforms and allow the user to run a program with the *local system* privileges. Thus, after the download of the mutant agent, the basic agent will also download a pre-compiled release of a system's local exploit.
Then, the *execute* module will run the local exploit with the mutant agent as parameter.

Finally, the first agent will end and let the new agent run with the local system privileges.
Now, we can set the agent to run as a service and so keep it hidden and permanent on the infected system.

With these two functionalities, the intrusion agent could be really dangerous into a company network. It can jump on other computers, set keystoker and sniffer programs to grab passwords and steal confidential documents. This kind of agent is a real threat for companies and governments which have highly condifential informations stored in their computers.

In the next sections, we will see how a hacker would use this kind of agent :

How an attacker injecst a program into the internal lan despite all the security elements ?

What could the administrators do to protect a network if the users do not follow the security policy ?

[Agent Injection]

In this section, we will explain how an attacker could target your lan and inject an agent program.
As we all know, the security of a system is equal to the security of its element the most vulnerable. The attacker will apply this principle and thus target the weakest part of your network : the users and their workstation.

Three ways of injecting are explained here, but many other ways exist and an attacker who really wants to hack your computer will always find a way to inject his agent.

Web injection and fake servers

Every month, a new security flaw about a web browser is discovered and published.
These flaws usually show how to execute arbitrary code on a user's computer browsing a malicious web site.
Indeed, a vulnerable web browser which downloads on an especially crafted web page with executable code in it will run the code.

If the malicious web site is controlled by our attacker, the crafted web page can embed an intrusion agent.
Now, the game consists in tricking the user to browse our malicious web site.

There are many ways to trick a web user, here I suggest three :

Send an email asking to see a genious web page ;
DNS spoofing and redirect.

The second way works fine. The principle is to corrupt a company's DNS server and to inject your IP address instead of Google's one.
Then, your fake server handles a copy of the real Google site. When the user asks the DNS for address resolution of Google, the DNS will resolve it with your IP address.
So, the user thinks he's using the real Google and then does normal search.

The user's request is sent to your web site which will ask the real Google. So, the real response is sent to the user, but with some modifications : the fake server will add a specific security exploit with the agent's code embeded in it.

The lector can refer to the DNS Spoofing package released by the ADM Crew and to the Internet Explorer security flaws.

Files Sharing Networks

Another way of injecting our agent is the Peer to Peer networks. In a company, many users install a Peer to Peer program to downlaod mp3 or divx files.

The principle of this kind of attack is to create a boggus program (or a mp3) that you share on your hard drive.
You just have to wait for a user to download and execute it. If the attacker knows the hobbies and tastes of a target, it is easy to rename the crafted file to trick the targeted user : a lot of buffer overflow are disclosed about Media Player, Winamp, etc.

With the Peer to Peer programs' instant messaging facilities, the attacker can send a memo to the target to propose a file exchange or run a setup program.

Social engineering

If the user does not use Internet and does not execute programs from the outside, it seems impossible to inject our agent.
But, they is still a way to force him to execute your agent : the social engineering.

The social engineering will allow us to con the user and trick him into executing the agent.

They are many ways to con a user, but the purpose of this document is not the explain what social engineering is and its different methods. The reader can refer to Kevin Mitnick's book *Art of deception.*

Social engineering works really fine, it's a scale problem : if you phone 150 people from the same targeted company pretending you're the "security administrator" asking them to download "the latest patch for windows media player", there will always be someone who will trust you and will follow your instructions.

[Diffusion and scenarii]

In this section we will describe two different methods to inject and use an intrusion agent.

scenario 1

The first scenario is based on a web injection and a fake server as described above.

The attacker wants to robe the financial budget of the Aqme company. He already knows the name of the financial responsible : Bill Griton.
He looks up on Google anything about this person. He finds out that Bill is a member of a sailing club.
So, the attacker prepares an e-mail advertising a new web site specialized in second hand boats.

When Bill receives the mail, he just has a look and finds a error page saying "The service is currently unavailable, sorry.".

But Bill doesn't worry and doesn't notice that there's a small running download.

At this moment, the attacker receives the agent's call. He can now begin seeking Bill's computer.
Like in every company, important business documents are stored on a file server.

Our attacker knows that and decides to lookup active shares on the computer. He quickly finds that the drive H: is in fact a network share.

The attacker is now able to download all documents from the company fileserver, with Bill's Id !

scenario 2

Another scenario : an attacker wants to get into a bank's core system (where customers accounts are stored).

The attacker was working in that bank as clerk, but was fired last days.

Normally, to connect to the Citrix server, the employees use a web page with a secure-ID authentication. Since a user is authenticated on the web page, the remote web server sends a java applet of the Citrix client. The connection between the java client and the Citrix server passes through the web server with a SSL 128bits crypto connection.

The fired employee wants to damage his old employer. For that, he needs a permanent access to the Citrix server to elaborate a chirurgical attack on the company database.

He knows that from the Citrix server, a user can surf the web through a proxy. He knows a login/password on that proxy, so he is able to hard code it in an agent.

But how to inject the agent on the Citrix server ?

In that company, all users' credential are erased when a employee is fired or dismissed. And, anyway, no file transfer is allowed on the Citrix server.
It is possible to surf the web, but files downloads are forbidden by the proxy.

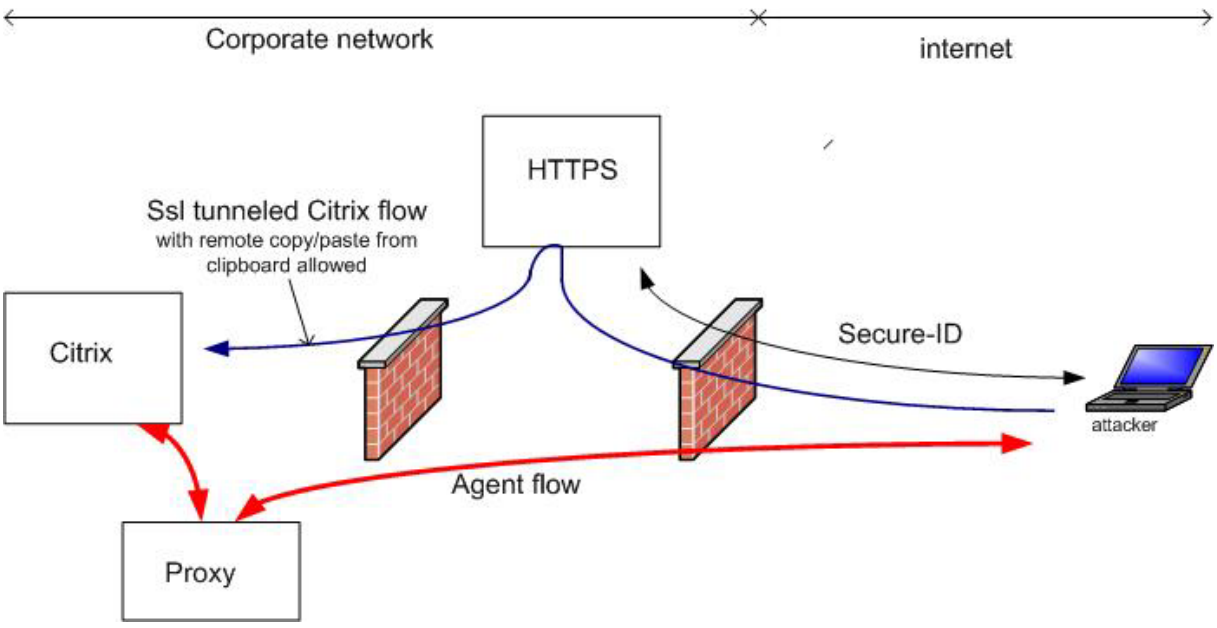
First, the attacker needs to connect to the Citrix server with a valid token. To perform that, the attacker phones a friend from the company and tells him that he needs help and just wants to borrow his ID to backup his personal data. What more normal ? His friend gives him his ID by phone and tells him to be quick.

Then, the attacker is logged on the Citrix with the ID of a friend.

How to put the agent on the Citrix ?

Easy. The attacker knows that Winzip is installed by default on each account. Winzip, like many other programs, is able to encode/decode base64 files.

So, the attacker encodes the agent executable in a base64 file and copies it to his clipboard.



Then, he creates a new file on the Citrix and pastes his clipboard in the remote file. Finally, with Winzip, he decodes the file and obtains the executable agent.

The attacker can now launch the agent and quit the Citrix connection. He just needs to quietly find a privilege escalation exploit (like the **shellExecute Exploit**) and create a secret administrator account with an auto-launch agent.

[Network of Spies]

The main difficulties in this kind of intrusion, is that the agent contains the address of the attacker.

The attacker can use a dynamic dns resolution, but there are still logs somewhere.

A future evolution of the agent concept, because there already is a reverse http backdoor on the Internet, is the concept of network of agent.

Thus, a agent will not discuss directly with the attacker console, but it will use a network of special agents settled over the internet.

These special agents are called “bouncers”. These agents are muted agents, transformed into bouncer agents.

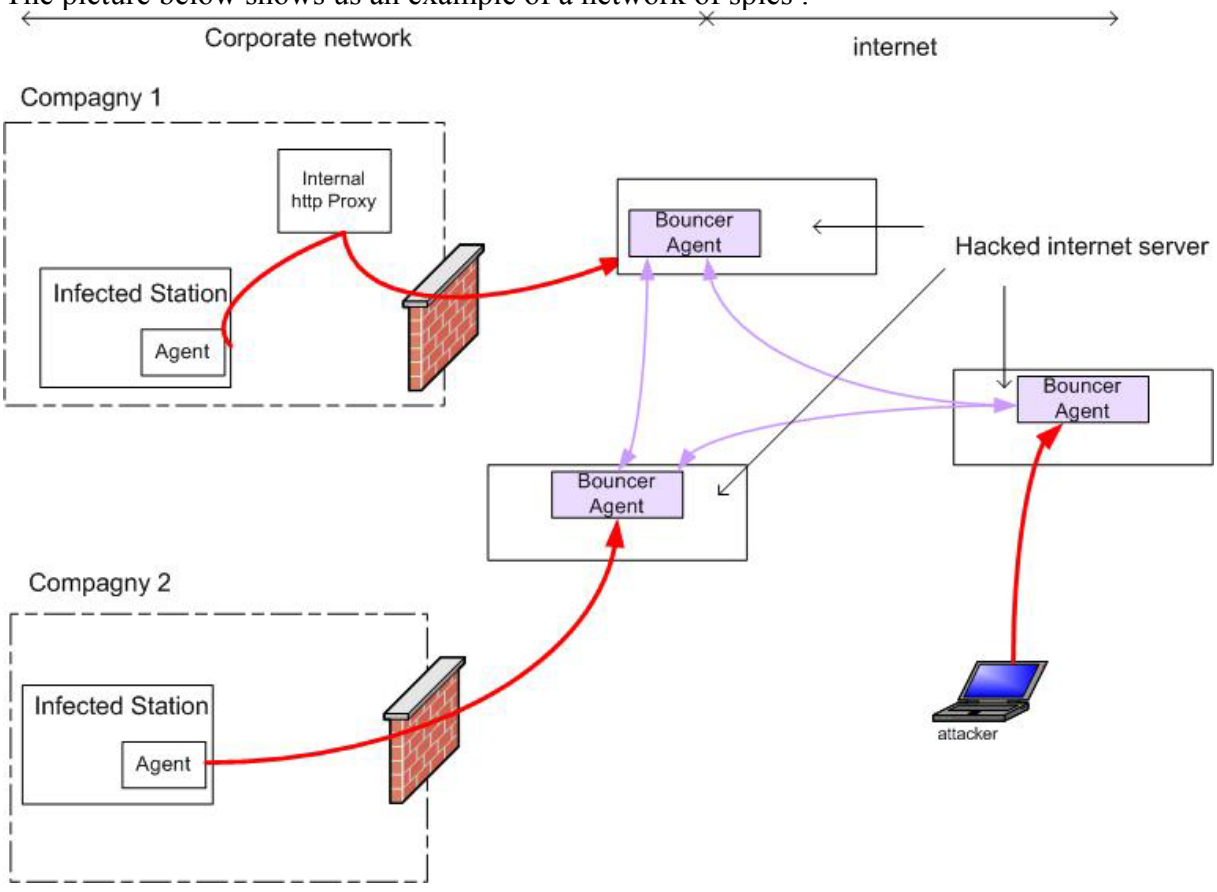
The attacker can use multiple Internet hacked servers to settle bouncer agents. Each bouncer is connected with others to form a web of spies.

Then the attacker just connects to a bouncer and then can contact any of the other agents.

The agent regularly contacts his bouncer to know a list of other bouncers (the map). Thus, if a hacked Internet server is lost, the agents (those which are settled into companies’ LAN) can still connect to the web of bouncers.

This way, the attacker could be multiple and undetectable. Even if an expert dissects a agent, he can not find the real attacker.

The picture below shows us an example of a network of spies :



The attacker can now connect his computer to any of the bouncers with a ssh client and then connect to all the agents belonging to the network of bouncers.

These kinds of network are born and could be shared by many hackers.

[Defenses]

The first idea which comes is that the antivirus will detect the agent. But this idea is wrong : the code of the agent must be self coded. Furthermore, the executable is encoded.

So, an antivirus program can not find a specific pattern for the agent.

A good defense is to settle a proxy server for all outgoing flow. But, we have seen in the scenario 2 that, in some cases, the agent could work.

The only hardening possibility is to use personal firewall programs. In this kind of firewall, the administrator lists all the applications of the system which are authorized to perform outbound connections.

But, there are other solutions than technical ones : a good security policy understandable by anyone in the company is the best solution.

Here are some important rules to be understood by everyone inside the company :

- Do not give any phone numbers to suspicious persons.
- Never run programs if a suspicious person ask you to do so.
- Do not download files from the internet .
- Do not use IE/outlook programs.
- Do not give any of your credentials, even to a friend.
- Do not shut down your Antivirus/Personal firewall programs.

There are no “magic solutions”. Addwares, viruses, worms, agent spies will proliferate more and more over the internet and in your company network.

[References]

About the author

I am a french computer science engineer. I work as a security consultant for a consulting company.
I am in charge of penetration tests, security audits and new techniques development.
I hold a graduate degree in engineering from the Polytechnic School of the University of Nantes (FRANCE).
Use *fcharpentier@laposte.net* for contact.

About the document

The first purpose of this intrusion agent was to demonstrate the ability to create a complex and intelligent shellcode for pentesting.

Then, during the development, I have pinpointed that this agent could be used as a spy (or a network of spies) for intelligence purpose.

I have decided to publish this paper to explain the risk and the countermeasure currently available on the security market.
This kind of technique will not be used for mass-hacking, but it could be really efficient for Intelligence and spy purposes.

And don't forget that the agent technique is not the difficult part here , there are many other ways to do so.
On the attacker side, the difficult part is to elaborate a chirurgical attack from the agent.
On the administrator side, the difficult part is to prevent and harden systems against these kinds of attack.

Special thanks to the security consulting company *XMCO Partners* for their help and labs.
Greetings to *Ghosted* for his help during the coding of my first agent.