OLD WIN32 CODE FOR A MODERN, SUPER-STEALTH TROJAN

BLACKHAT EUROPE 2004

Eric DETOISIEN – Team Rstack Eyal DOTAN – Tegam International

Agenda

- Introduction
- Communication model
- Code injection
- API Hooking
- Final demo
- Future evolutions
- Prevention
- Conclusion

Introduction

- Today there are reliable (?) methods for securing public part of the information system from outside threats (hardening, defense in depth, authentication, crypto, ...)
- But the weakest link in information security is still the endpoint a.k.a the workstations. Even if sysadmins can trust their users, how can they trust the programs they are running?

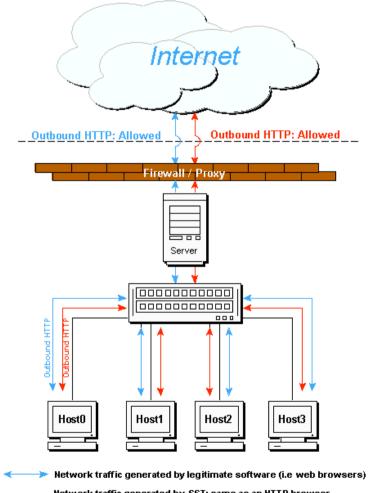
Introduction

- Trojan horses are more and more widespread, they are very interesting to target the end-user.
- But most Trojan horses are easily detectable and thus are inefficient in a corporate environment behind firewalls, proxies, and desktop security software.

Introduction

- In this presentation, we will show that it is perfectly possible to implement a super-stealth Trojan horse using Win32 techniques that have been around for over 10 years now.
- We will also discuss prevention methods and discuss about the existence of such stealth programs.
- We'll assume the following protections : <u>User</u> <u>privileges</u>, <u>Desktop Firewall</u>, <u>Edge Firewall</u>, <u>authentication-enabled Proxy</u>

- Communication capability is the first thing that trojan needs
- The Trojan horse communicates with the outside world via HTTP protocol. It regularly checks for instructions on a Web server maintained by the attacker.
- Edge firewalls see HTTP requests initiated from inside the network, but they cannot tell the difference between this and a regular Web trafic from a browser



Network traffic generated by SST: same as an HTTP browser using HTTP protocol, and no Listening ports

- This communication mechanism is basic and still quite efficient.
- HTTP Covert Channel isn't new in the security world (HTTPTunnel, Setiri, webdownloader, ...) but trojans with this function aren't widely diffused (why?)
- In a Windows environment the simplest way is to use WININET.DLL

DEMO

[Simple trojan based on WININET.DLL for HTTP communication]

HTTPTrojanBasic skills

- HTTP communication
- IE Proxy configuration detection

HTTPTrojanBasic drawbacks

- Detected by Personal Firewall
- Blocked by proxy authentification
- Not stealthy
- Survival instinct very low

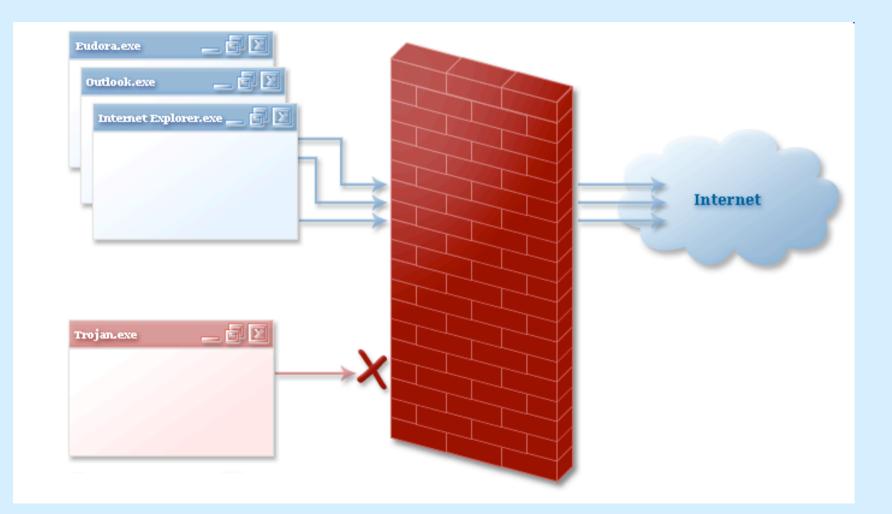
Code injection mechanism

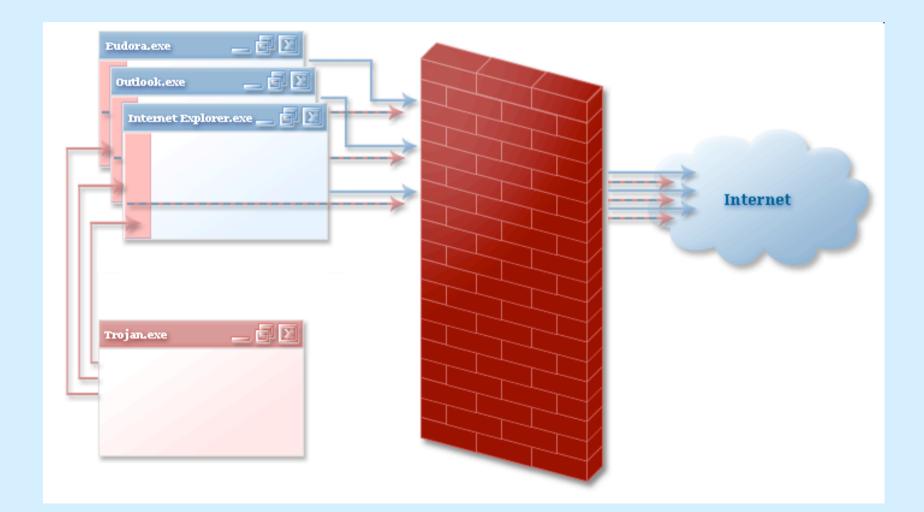
- Code injection is a technique that has been known for about 10 years: "Load Your 32-bit DLL into Another Process's Address Space Using INJLIB" -Jeffrey Richter (May 1994).
- Direct code injection (no DLL, pure thread injection).
 Harder to code but more stealther
- Most importantly: no particular rights are required. All processes that belong to the same user can be injected.

- APIs for code injection:
 - OpenProcess : get handle on target process
 - VirtualAllocEx : memory allocation in target process
 - WriteProcessMemory : inject (write) code into allocated memory
 - CreateRemoteThread : execute code from target process

Benefits for a Trojan horse

- Bypassing desktop firewalls by injecting authorized applications.
- Even behavior monitoring software can be fooled by such manipulation.
- Allows the Trojan to easily hook APIs in injected processes, for other purposes (which we'll see in the next section).





DEMO

[Previous simple trojan but now it's injected in process]

HTTPTrojanInjected skills

- HTTP communication
- IE Proxy configuration detection
- Personal Firewall bypass
- Stealthy

HTTPTrojanInjected drawbacks

- Blocked by proxy authentification
- Survival instinct very low

Use in existing malicious programs

Today there are several malicious programs that use injection techniques (not necessarily for the same goals described above) :

- BackStealth (proof of concept)
- Optix, Beast and other Trojan horses
- Keylogger ...

Taking code injection farther

- « Inject and die »: once the Trojan injected its code, it can terminate and disappear from Windows' task manager and process list.
- Survival of the injected thread: once the Trojan horse dies, it depends on the survival of injected processes.
 Solution: injecting all user processes at a regular interval.

DEMO

[Multi-Injection Proof of Concept]

New skills

- Very good survival instinct
- Control of all processes

« Still here » drawbacks

Blocked by proxy authentification

How does it work?

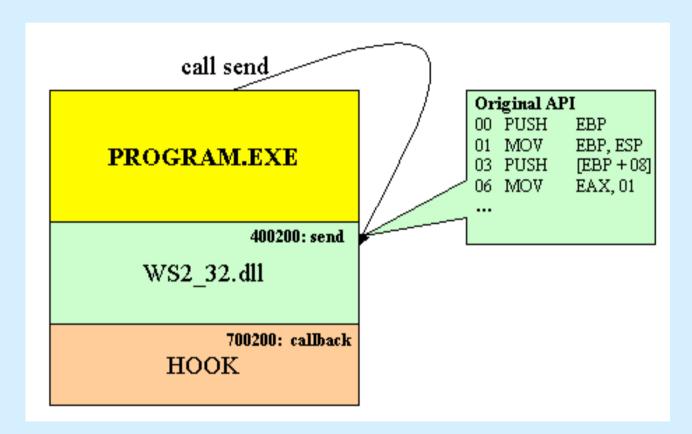
- API hooking has also been known for 10 years: "Peering Inside the PE: A Tour of the Win32 Portable Executable File Format" - Matt Pietrek (March 1994)
- Most popular method: IAT (Import Address Table) hooking.

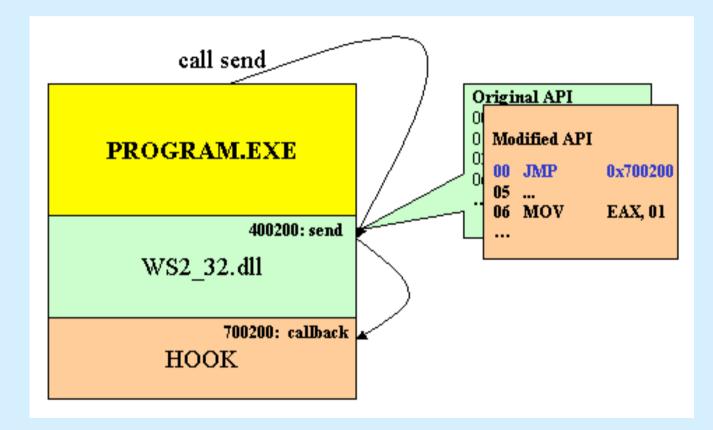
Doesn't work well for Trojan horses:

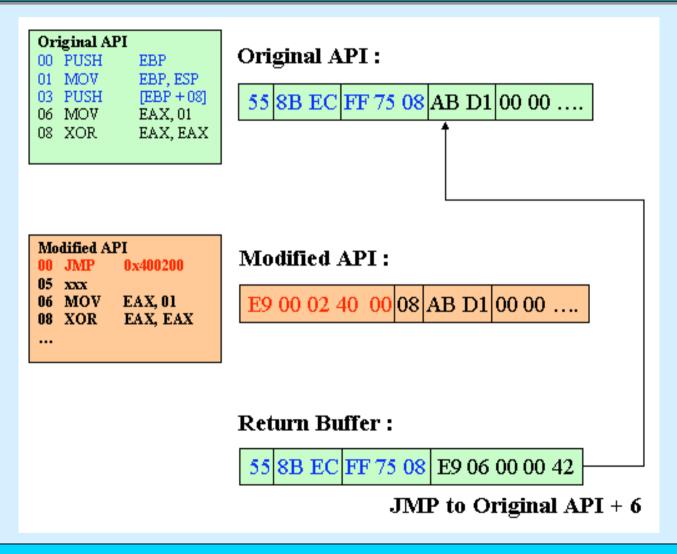
1. Not all APIs are IAT-hook friendly! Some APIs call other APIs directly, without using the IAT (LoadLibrary & GetProcAddress)

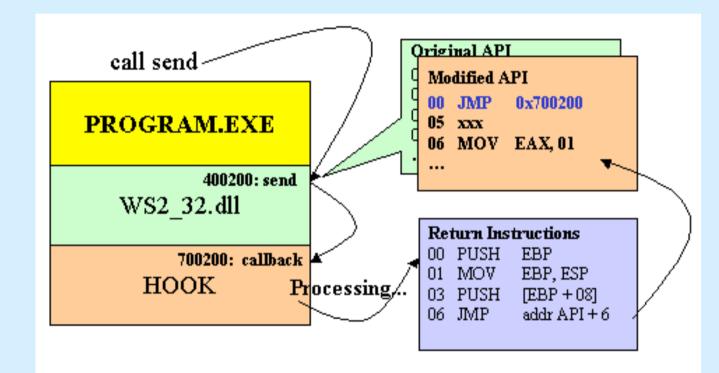
2. When the Trojan horse is injected, the program may have already obtained the API 's address, before the Trojan could hook GetProcAddress.

Best method we know of: JMP redirection









```
int WINAPI SendHook (SOCKET s, const char FAR * buf, int
  len, int flags)
{
  DWORD
                           ZoneTampon;
  int
                           Result;
  // Pre-processing...
  // Call original API
  asm {
      push flags
      push len
      push buf
      push s
call ZoneTampon
      mov Result, eax
  // Post-processing...
  return (Result);
```

API hooking: What for?

- Identifying communication applications: hooking socket APIs such as « connect ».
- Intercepting the CreateProcess API -- making thread survival more efficient, more aggressive. Every time a process is created, the Trojan horse injects it.

- Rootkit-like features: hiding files and registry keys to hide the Trojan's most visible items: registry keys for execution at startup, as well as the Trojan's binary itself. All in user mode, and... without local administration privileges!
- API hooking allows the Trojan to log local TCP/IP trafic in user mode and without Admin privileges.

Here are some actions a Trojan horse can perform :

- Spy e-mail, proxy and socks passwords (send hook)
- Spy incoming and outgoing e-mail messages (recv & send hooks)
- Spy Web authentication data & forms (<u>recv & send</u> hooks)
- Intercept and deny / simulate anti-virus processes connecting to their signature update servers (specific, depends on which anti-virus is to be aimed).

Malware and API hooking today

 Some malicious programs have implemented API hooking as of today (rootkit like Hacker Defender or Vanquish)

Taking API hooking farther

- No DLL injection (same as for thread injection)
- Hooking essential APIs: Winsock APIs, but also LoadLibraryW.

Final Demo

CASPER THE FRIENDLY TROJAN He really wants to be your friend

Injection & API Hooking

- Try to inject code without CreateRemoteThread
- Include a length-disassembler engine for API Hooking

Bypassing anti-viruses

- This kind of Trojan horses is made for aimed attacks, not for mass distribution.
- Still, the Trojan horse can escape detection in case it becomes known, by implementing an auto-update feature that keeps changing the Trojan horse's binary.

Bypassing anti-viruses

This kind of auto-update feature is very common for legitimate software. Why not expect to see autoupdating malware?

Different communication protocols

For our proof-of-concept Trojan horse, we've used HTTP protocol. Other protocols may be used as well: DNS, FTP or SMTP for example.

Sniffing encrypted traffic

When the navigator communicates via HTTPS, data sent via the « send » function is encrypted. By intercepting higher-level APIs, we can see the data before it is encrypted. This depends on the Web browser used. Internet Explorer uses high-level WININET APIs, which allows the Trojan to intercept data before it is encrypted.

Remote control

This kind of Trojan horse would be even more efficient if it had real-time remote control (just like VNC).

The problem left to resolve is the inversed client-server communication we're using -- too heavy for this kind of operations and the HTTP tunneling.

Prevention

- Hence, prevention is better than cure.
- Protection can be achieved by these steps :
 - 1. Don't let come unknow code to the user
 - 2. Educate user to avoid click everywhere symptom
 - 3. Securing the users' workstations from untrusted code.

Prevention

- HTTP, FTP and SMTP filters can help, to keep users away from unauthorized executable code.
- Educating and training users to keep away from untrusted code.
- Specific anti-injection techniques (i.e hooking CreateRemoteThread). But this is not the end of the story... There are other more indirect ways of injecting code into other processes.

Conclusion

- In Windows, whenever a malicious program is executed, its possibilites are almost unlimited.
- Question: since most of the techniques shown in this presentation already exist, how come we don't see many Trojan horses using them?
- Possible answer: The fact that the Win32 API has only recently been unified (i.e. thread injection didn't exist in Windows 9x, and API hooking was quite different from Windows 9x to NT systems).

Conclusion

Q&A THANK YOU

Eric DETOISIEN

http://www.rstack.org http://valgasu.rstack.org valgasu@rstack.org

Eyal DOTAN

Tegam International http://www.viguard.com edotan@viguard.info

http://valgasu.rstack.org/casper/