



Shellcode Penetrate Firewall

Author: san@xfocus.org



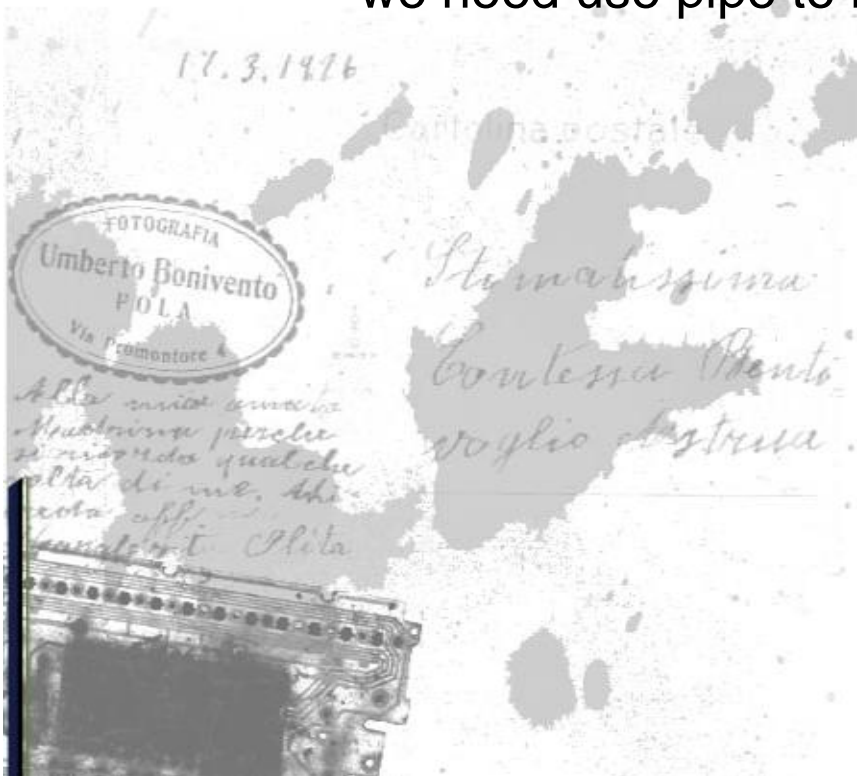
- Overview
 - 1. Implementations of remote shellcode.
 - 2. Questions and advantages of find socket of current connection.
 - 3. Implementations of Win32 platform.
 - 4. Implementations of Linux x86 platform.
 - 5. Implementation of AIX PowerPC platform.
 - 6. Thanks
 - 7. References

- 1. Implementations of remote shellcode
 - 1.1 Bind port
 - 1.1.1 Bind a new port
 - 1.1.2 Reuse the original port
 - 1.1.2.1 Port reuse
 - 1.1.2.2 Port rebind
 - 1.2 Reverse connect
 - 1.3 Reuse the socket of the current connection.
 - 1.3.1 ECB structure of IIS
 - 1.3.2 getpeername
 - 1.3.3 FIONBIO
 - 1.3.4 FIONREAD
 - 1.3.5 OOB Data
 - 1.3.6 Hook the recv function

- 2. Questions and advantages of find socket of current connection
 - 2.1 Bind shell
 - In Unix, the socket can directly duplicate to standard input, output, error handle of `"/bin/sh"`.
 - In Win32, the socket is broken into overlapped socket and non-overlapped socket.
 - The socket that is created by function `socket()` will have the overlapped attribute as default, and it can't directly duplicate to standard input, output, error handle of `cmd.exe`. It must be use pipe to transfer data.

– 2.1 Bind shell

- The socket that is created by function `WSASocket()` will have the non-overlapped attribute as default, so it can directly duplicate to standard input, output, error handle of `cmd.exe`.
- Winsock recommend programmer use overlapped socket, so we need use pipe to bind shell for the best.



– 2.2 Find socket in multi-thread enviroment

----- start sample code -----

```
s = WSASocket(2,1,...)
```

```
bind(s,..)
```

```
listen(s,...)
```

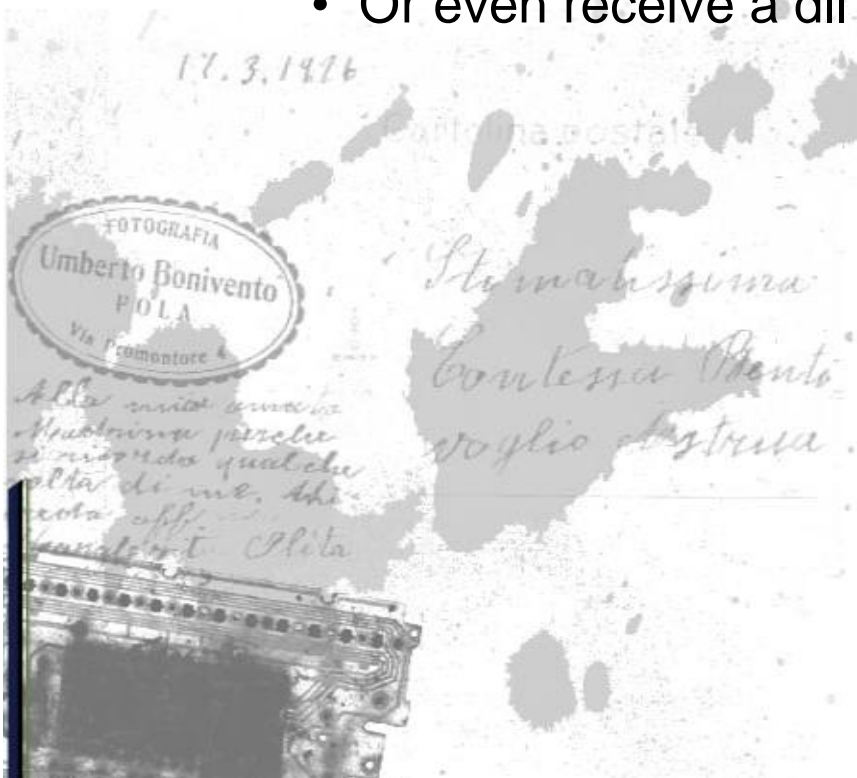
```
s2 = accept(s,...)
```

----- end sample code -----

- When the s is under accept, all of the operations to this socket will be wait until connect up.
- The WaitForSingleObjectEx function determines whether the wait criteria have been met. The s will return WAIT_TIMEOUT under accept, otherwise it will return WAIT_OBJECT_0.

– 2.3 Advantage

- Reuse the current socket is more harder to discovery and it is the best way to through the firewall.
- The shellcode of finding socket is small, so it can receive a more complex shellcode and jump it to execute.
- Or even receive a dll file to implement very complex function.



- 3. Implementations of Win32 platform
 - 3.1 Implementation of port reuse
 - The server must be bind in 0.0.0.0, and it didn't use SO_EXCLUSIVEADDRUSE option.
 - The exploit must write IP address and port of the server to shellcode before send overflow data.
 - The shellcode execute as following:

```
setsockopt(s, 0xFFFF, 4, &d, 4);  
bind(s, &sockaddr, 0x10);
```
 - "netstat -na" display that 0.0.0.0 and actually IP address were both bind the same port at server side. So this method is easy to discovery.

– 3.2 Implementation of rebind

- LSD introduce this method in their Win32 Assembly Components.
- Create new process in suspended state.
CreateProcess(NULL,"cmd",NULL,NULL,0,CREATE_SUSPENDED,NULL,NULL,&si,&pi);
- Get full context of created primary thread containing, among others, the processor register set.
GetThreadContext(pi.hThread,&ctx);

– 3.2 Implementation of rebind

- Use the VirtualAllocEx() function to allocate memory in a remote process.

```
v=VirtualAllocEx(pi.hProcess,NULL,0x5000,MEM_COMMIT,  
PAGE_EXECUTE_READWRITE);
```

- Copy the buffer containing assembly procedure to the allocated memory.

```
WriteProcessMemory(pi.hProcess,v,buf,sizeof(buf),NULL);
```



– 3.2 Implementation of rebind

- Modify the context of primary thread in such a way that the instruction pointer register(EIP) would point at the beginning of allocated memory, where the assembly procedure was copied.

```
SetThreadContext(pi.hThread,&ctx);
```

- Resume primary thread of child process what starts execution of copied assembly procedure.

```
ResumeThread(pi.hThread);
```

- Terminate the current process, so it will release the resource of the socket.

```
TerminateProcess(-1, 0);
```

- Loop bind the same port until success.

– 3.3 Find socket by getpeername

- The exploit must use the `getsockname()` function to get the client socket information and write it to shellcode before send overflow data.
- The shellcode use the `getpeername()` function to get the attacker side socket information and compare with the information that reserved in the shellcode.
- If the information matched, then break out the loop and bind a shell to the socket handle.
- This method exists a problem that the information sent by attacker in the NAT network environment won't matches with the one got by server.

– 3.4 Find socket by match string.

- The find flow as following:

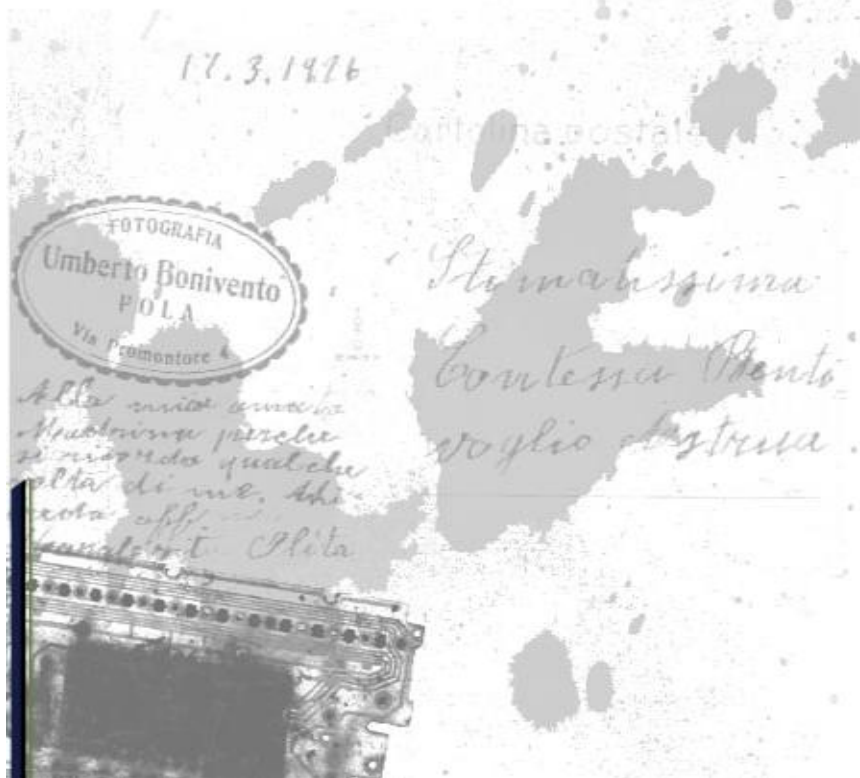
```
while (1)
{
    i++;
    ret = WaitForSingleObjectEx(i, 10, 1);
    if (ret != 0) continue;
    ret = ioctlsocket(i, FIONREAD, &ul);
    if (ul != 4) continue;
    recv(i, buff, 4, 0);
    if( *(DWORD *)buff == 'Xc0n') goto shell;
}
```

- This method avoid the problem of getpeername.

– 3.5 Hook the recv function

- Use the VirtualProtect() function to change page protection at first.
- Then write five bytes that jump to new recv in the start of the recv() function.
- In the new recv function, the first thing is restore the five bytes opcode at start of original recv function.
- Second, call the original recv function to receive the buffer and determine if this socket is the current connection.
- This method can bypass rpc mechanism.

- 3.6 Integrate with file upload and download functions
 - The function of file upload and download is important after exploited success.
 - The client must cooperate with the shellcode to read/send or receive/write files.



- 4. Implementations of Linux x86 platform
 - 4.1 Set socket state by fcntl.

- Idea from NSFOCUS's scz:

```
while (1)
{
    i++;

    oldflags = fcntl(i, F_GETFL, 0);
    fcntl(i, F_SETFL, oldflags | O_NONBLOCK);
    read(i, buf, 4);
    fcntl(i, F_SETFL, oldflags );

    if (buf == 'Xc0n') goto shell;
}
```


– 4.2 Send OOB data

- bkbll, the guy from CNHONKER use this technology first. Out of band data won't be blocked in Berkeley socket implement. The flow of find socket as following:

```
while (1)
{
    i++;

    recv(i, buf, 1, 1);
    if (buf == 'l') goto shell;
}
```

- This is the easiest method in Unix/Linux.

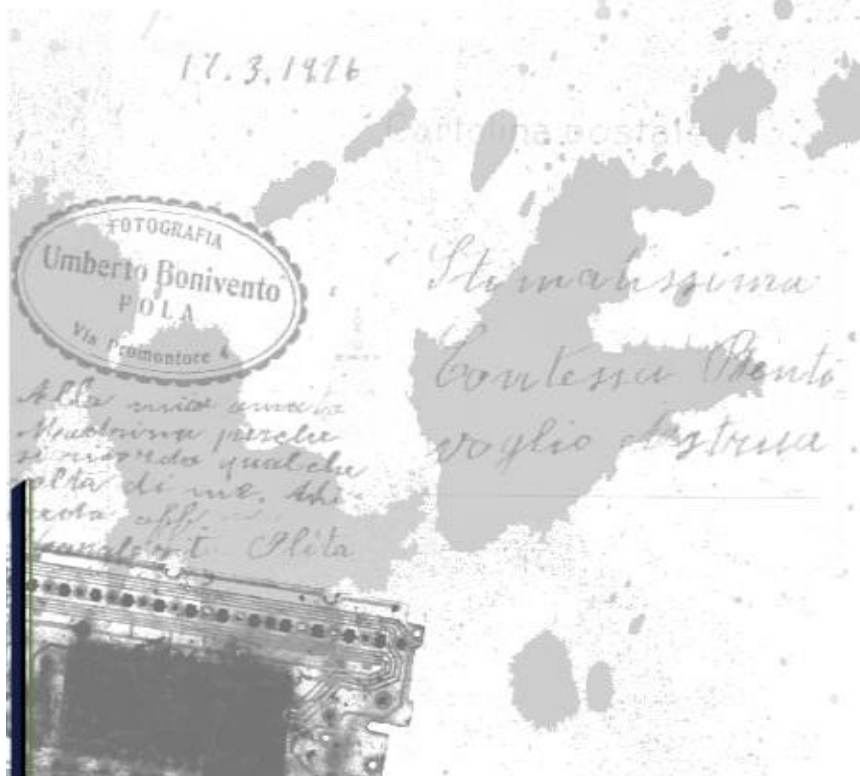
– 4.3 Use character of the ioctl() function

- Option FIONREAD of the ioctl() function determine the amount of data pending in the network's input buffer that can be read from socket. It's flow as following:

```
while (1)
{
    i++;

    ioctl(i, FIONREAD, &ul);
    if (ul != 4) continue;
    read(i, buf, 4);
    if (buf == 'Xc0n') goto shell;
}
```

- 4.4 Integrate with file upload and download functions.
 - Just like Win32 implement, but it can't encode the data of interactive.



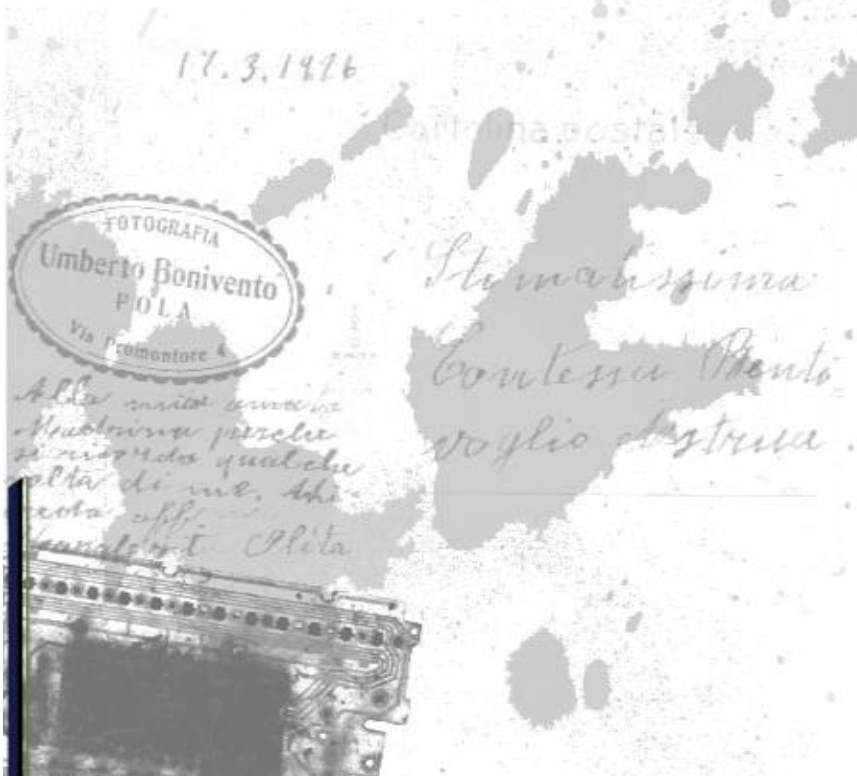
- 5. Implementations of AIX PowerPC platform
 - Cache Mechanism
 - instruction cache
 - data cache
 - Encode shellcode
 - It have problem write in normal way.



– Self-modifying code

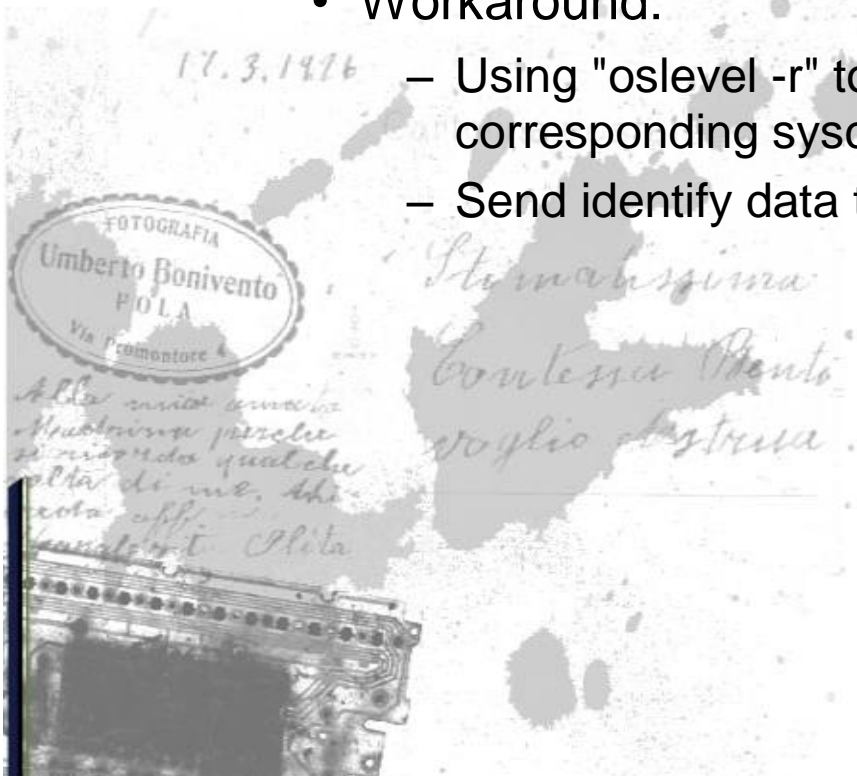
- Store the modified instruction.
- Issue the dcbst instruction to force the cache line containing the modified instruction to storage.
- Issue the sync instruction to ensure dcbst is completed.
- Issue the icbi instruction to invalidate the instruction cache line that will contain the modified instruction.
- Issue the isync instruction to clear the instruction pipeline of any instruction that may have already been fetched from the cache line prior to the cache line being invalidated.
- It is now okay to execute the modified instruction. An instruction cache miss will occur when fetching this instruction, resulting in the fetching of the modified instruction from storage.

- AIX have no cache management instructions.
 - There is a simple method to resolve this problem is execution a syscall after code modified.
- Encode shellcode is ok.

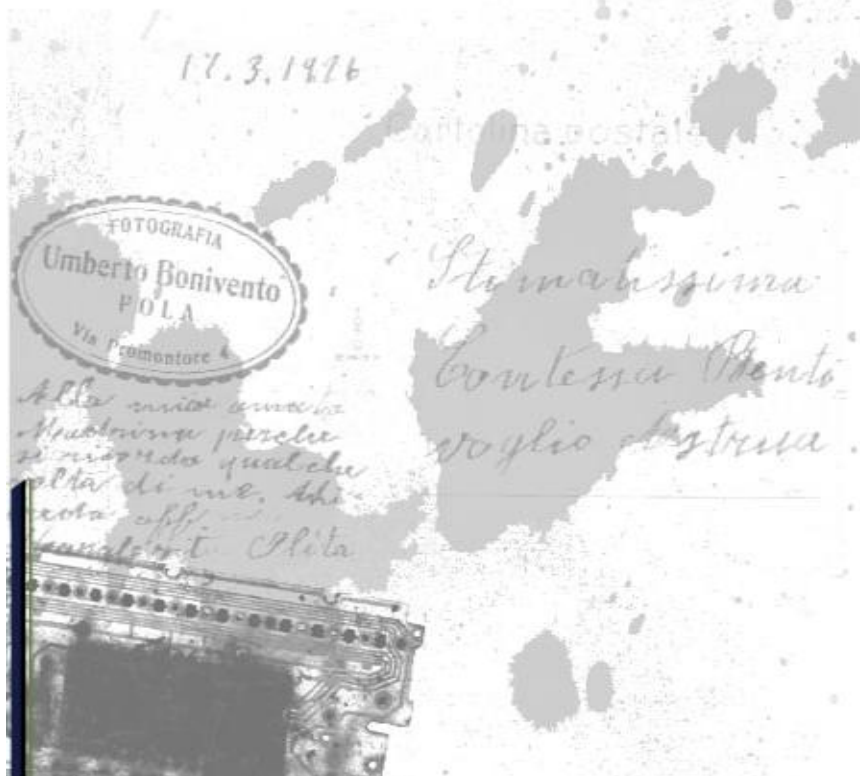


– 5.1 Send OOB data

- Just like Linux implement.
- Problem:
 - There are different syscall numbers in various AIX editions. So the exploit is not universal.
- Workaround:
 - Using "oslevel -r" to determine AIX version, and then write in the corresponding syscall number.
 - Send identify data to dtscpd service to determine AIX version.



- 6. Special thanks
 - eyas, scz, flier, tombkeeper, watercloud, Emmanuel, H D Moore, KF...



- 7. References

- [1] Win32 Assembly Components
 - <http://www.lsd-pl.net/documents/winasm-1.0.1.pdf>
 - <http://www.lsd-pl.net/documents/winasm.ppt>
 - <http://lsd-pl.net/projects/winasm-1.1.tar.gz>
- [2] Win32 One-Way Shellcode
 - <http://www.blackhat.com/presentations/bh-asia-03/bh-asia-03-chong.pdf>
- [3] Understanding Windows Shellcode
 - <http://www.hick.org/code/skape/papers/win32-shellcode.pdf>
- [4] http://www.0x557.org/release/ex_servu.c
- [5] Serv-U FTPD 2.x/3.x/4.x/5.x "MDTM" Command Remote Exploit
 - <http://www.cnhonker.com/index.php?module=releases&act=view&type=3&id=54>
- [6] 一段远程shellcode
 - <http://bbs.nsfocus.net/index.php?act=ST&f=2&t=144419>
- [7] 利用OOB查找socket(更正by bkbll, sorry to scz)
 - <http://www.cnhonker.com/index.php?module=articles&act=view&type=6&id=28>
- [8] The GNU assembler
 - <ftp://ftp.gnu.org/gnu/Manuals/gas/text/as.txt>
- [9] Linux System Call Table
 - http://www.system-calls.com/sys_call_table.php
- [10] UNIX Assembly Codes Development for Vulnerabilities Illustration Purposes
 - <http://www.lsd-pl.net/documents/asmcodes-1.0.2.pdf>
 - <http://www.lsd-pl.net/documents/asmcodes.ppt>
 - <http://www.lsd-pl.net/projects/asmcodes-1.0.2.tar.gz>
- [11] A developer's guide to the PowerPC architecture
 - http://www-900.ibm.com/developerWorks/cn/linux/l-powarch/index_eng.shtml
- [12] UNIX Network Programming W.Richard Stevens

- Thanks for coming and enjoy the XFOCUS conference!

