Security - Windows (part 2)

Note Title 4/7/2011

Announcements

- Checkpoints looked good!
- Due date for lab is next Wed.
- Lab 5 will be over malicious code
  - Lost 2 lectures

from MS perspective

System Hardening in Windows

1. Attack Surface Reduction: 80/20 rule
   - If the feature is not used by 80% of users, it should be off by default.

Issue: Many more non-technical users.

(2) Replace anonymous network protocols with authenticated ones!

Ex: Blastr Worm used remote procedure call (RPC), called anonymously.
SP2 required all RPCs to be authenticated.
When Zotob worm came out (exploiting RPC vulnerability in Plug 'n' Play), it was less successful even with the bug present, since the worm wasn't authenticated.

Best part: The user is unaware!

In general, Windows tends to harden servers more than clients.

Why?

- Servers are bigger targets
- clients need versatility
  "easier" to strip services from a server
- servers are run by more experienced tech people

Account Defenses in Windows

- Malicious software running with some SIDs can be very bad.
- Use principle of least priviledge.
- XP defaulted to admin, since older code would not work otherwise
- In Vista (finally), default is user (not admin). If not an admin, dialog box lets you authenticate as one.
  If account is an admin, still gives an "are you sure?" box. (Why?)

Low Priveledge System Accounts

Most Windows services are processes that start at boot-up + run as long as the computer is on.
<u>Ex</u>: File, Print, DNS

Many <u>need</u> elevated priviledges, but not all the time.

Windows has Local Service account + Network service account <u>as</u> alternatives which are not in admin group.

Ex: RPC & Blastr Worm

- Before SP2, RPCSS ran as System account - most priviledged.
  It needed it only to execute Distributed COM objects on a remote computer directly.
  But most RPC traffic didn't need this priviledge!

SP2: RPCSS runs as Network account.
New process DCOM server still runs as System, but only used for a small portion of traffic.

(like Apache on linux or IIS6 on Windows)

Stripping Priveledges

In terms of applications coding, it is possible to strip priviledges.

Ex: Index process in windows.
Checks what has been changed so it can reindex the file.
Only admins can get a volume handle, but as soon as that is done, it downgrades itself
(calls AdjustTokenPriviledges)

Network defenses

While user account priveledges are important, attack from the network may have nothing to do with that!

Services such as email, DNS, web servers, etc, will provide vulnerabilities in the system with no user interaction.

Denial of Service & IPv4

IPv4 is unauthenticated – think UDP. Spoofing is easy, & (therefore) so are distributed denial of service (DDOS) attacks.

Fundamental flaws here, so difficult (if not impossible) to fix!

Windows offers built in IPv6 & IPSec support, both of which fix this issue.

Ex: XBox & IPSec

Firewalls

All Windows OSs (since XP) have a built in firewall.

On XP, limited:
- not turned on by default
- only blocked inbound connections / on specified ports

On SP2, slightly better:
- Allows secure file sharing & printing on a home network (so can open port only to local subnet)
- Firewall enabled by default

Firewall in Vista:

- fully integrated component of TCP/IP / networking stack
- supports blocking outbound ports (optionally)
  ↳ (not really all that great...)

Buffer Overflow Defense

C is the big problem.
Why? not checking buffer sizes in common functions (gets..)

- Designed to be a high-level assembly language

Why not rewrite everything? (pointers)

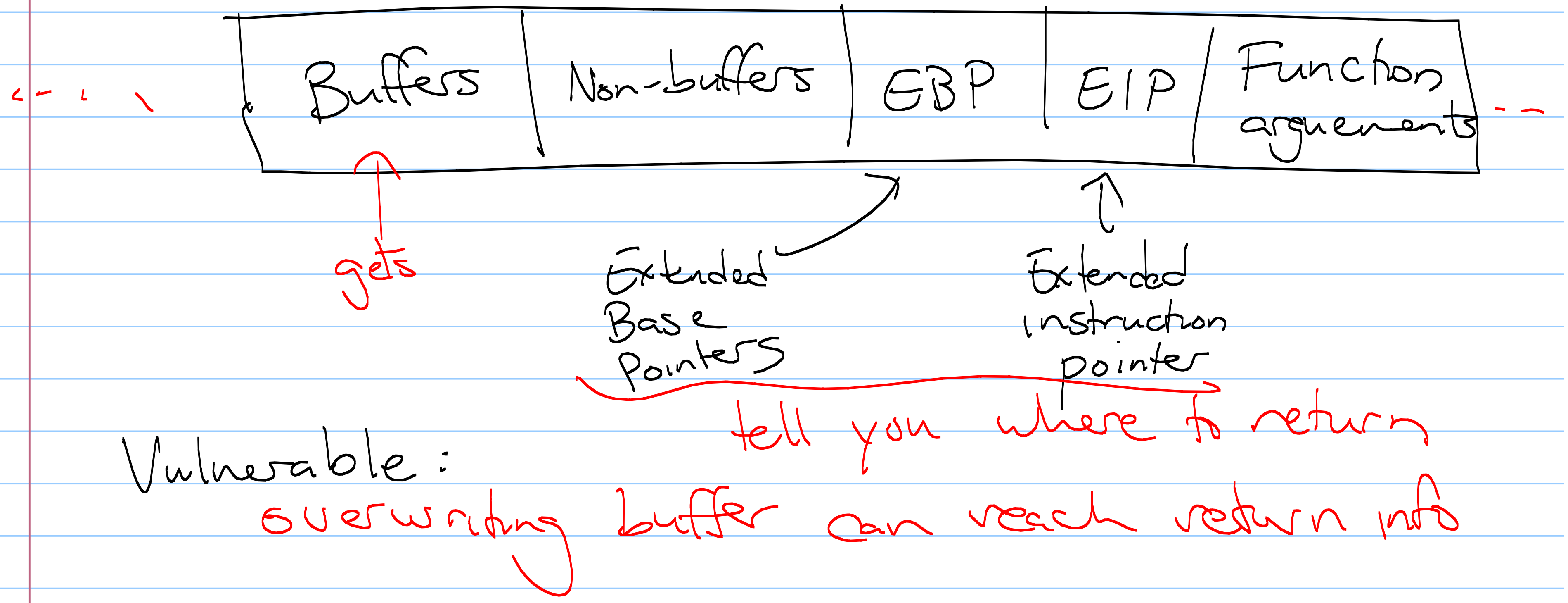
- lots of code
- speed

Underlying problem!

☆ **Real problem:** Developers trust data they recieve as input.

Windows Protection

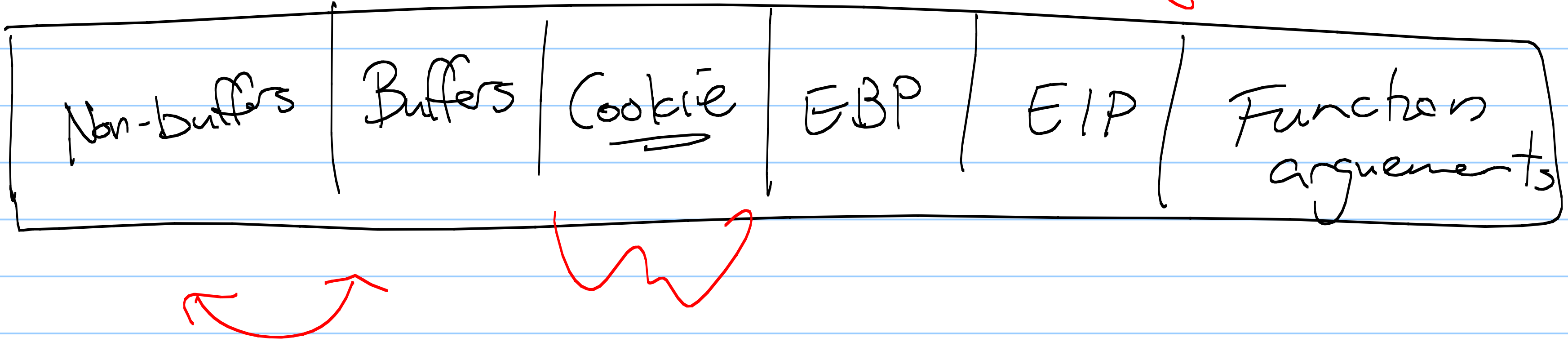
Windows Stack:



Vulnerable:

overwriting buffer can reach return info

In XP SP2:
- Compiled w/ a special option in Visual C++ to add defenses — /GS ← guard stack



This option affects any fcn with ≥4 bytes of contiguous stack data if it takes a pointer or buffer as an arguement.

No eXecute:

(NX for AMD, DEP by MS, & XD by Intel)

Goal: Prevent code from running in data segments.

(We talked about this a while ago.)

Windows XP SP2 & Vista enables this on any hardware that supports it.

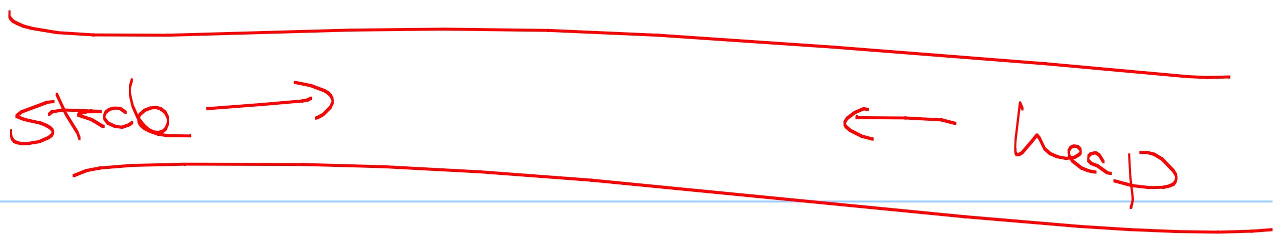
Stack Randomization

When a thread starts in Vista, the OS randomizes the stack base address by 0-31 pages (each 4k bytes in size).

Then it chooses a random offset within that page.

Removes predictability + makes smashing the stack harder.

The "heap"



This is memory which is set aside
for dynamic allocation.
(Think new, delete, memory leaks, etc.)

The heap can also be attacked.

Defenses in Windows:
- a cookie is added to each block & is checked to detect tampering
- integrity checking: when de-allocating, metadata is checked for validity
- heap randomization

Service Restart Policy

In all of the previous items, the program fails if tampering is detected.

But what if we immediatly relaunch?

Machine is still compromised!

In Vista, "critical" services restart only twice to address this.

Web Browser Defense

IE7 attempts to deal with common vulnerabilities.

- Active-X opt-in
  asks user if they are sure

- protected mode - forces IE to run at a low integrity mode
  MAC

Cryptographic Support

- Encrypting File System (EFS)
    - encrypts entire directory rather than individual files.
    - Why?

- Data Protection API
    - built in cryptographic protocol, so keys are handled by OS & not user
    - CryptProtectData & CryptUnprotectData (which use user password + other info to secure)

Crypto (cont.)

- Bit Locker (laptops)
  - encrypts an entire volume with AES
  - key is on USB drive or in chip on the motherboard (more next)
  - Also interfaces with AD to store keys

- Trusted Platform Module (TPM)
  - moves crypto to hardware
  - Vista uses TPM (the chip on motherboard) to verify that OS has not been tampered with
  - known as trusted boot or secure startup