NC STATE UNIVERSITY

CSC 405
Introduction to Computer Security

Windows

Alexandros Kapravelos
kapravelos@ncsu.edu

(Derived from slides by Chris Kruegel)

Windows

- ~82 % of all computers run Windows[1]
  - when dealing with security issues, it is important to have (some) knowledge of Windows
  - good example of non-open source system and security issues

- Started in 1985
  - graphical add-on to MS DOS

- Two main families
  - building on DOS legacy

    Windows 1.0, Windows 3.11, Windows 95, Windows ME
  - NT line (true 32 bit, multi-user OS)

    started with NT 3.1, NT 4.0, Windows 2K, XP, Vista, Windows 7, Windows 8, Windows 10

1 *according to StatCounter for July 2016*

Windows 3.11 – Windows ME

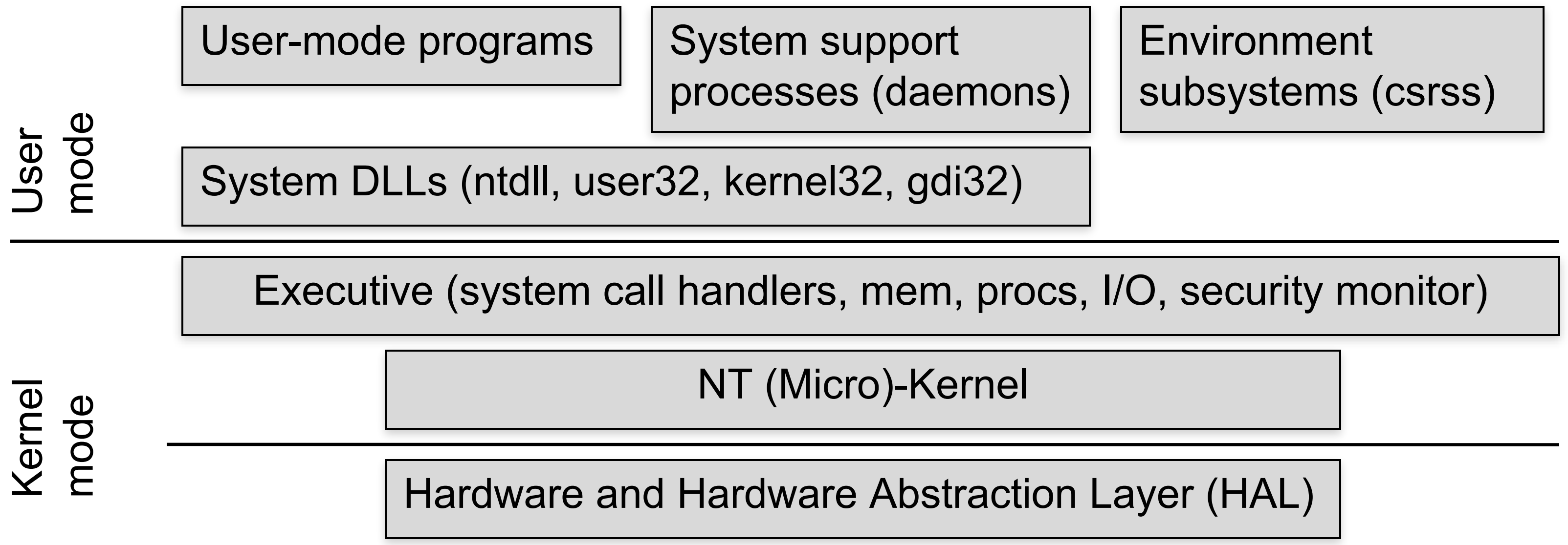
- Conceived as a single user OS
- Basically no security
  - protected mode introduced with Windows 95
  - full access to registry and file system objects
  - authentication process just to select profiles
- Profiles
  - desktop preferences
  - access to saved passwords (stored in a .pwl file)
  - user password unlocks password list file

Windows 3.11 – Windows ME

- Password list files
  - Windows 95 (very easy to break algorithm)
  - Windows 98 (better protection) – but files are world readable
  - guessing attacks possible (also, passwords are converted into uppercase)
- Many vulnerabilities found
  - no patches are issued anymore

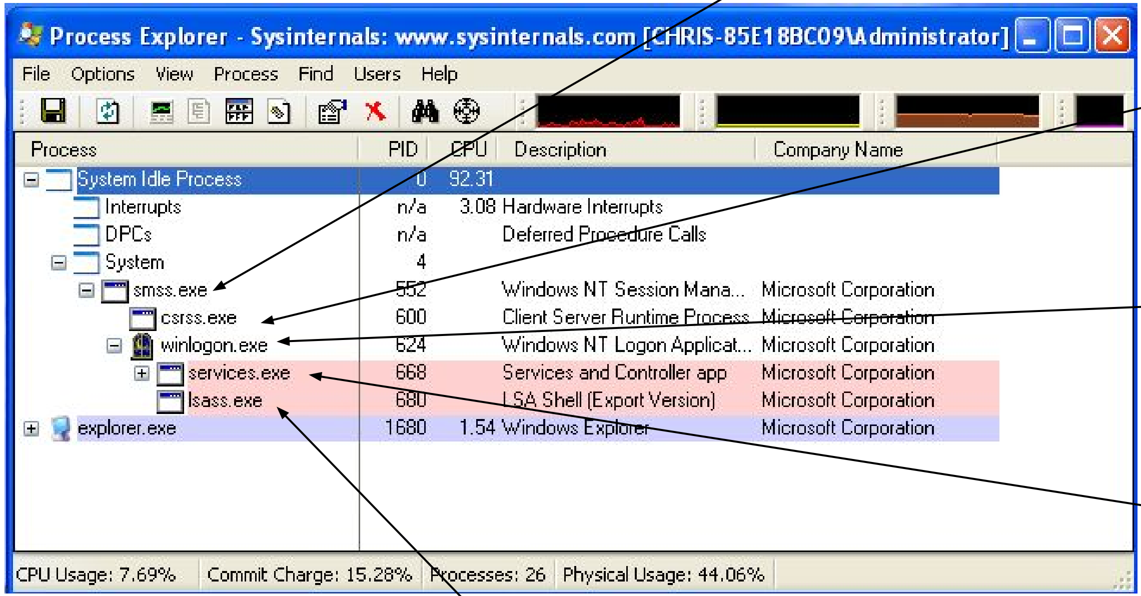
Windows NT

- Competitor to Unix
  - true multi-user
  - emphasis on portability and object-oriented design
  - isolation for applications and resource access control
  - similar to Unix, kernel and user mode


User-mode programs
System support processes (daemons)
Environment subsystems (csrss)
System DLLs (ntdll, user32, kernel32, gdi32)
User mode
Executive (system call handlers, mem, procs, I/O, security monitor)
NT (Micro)-Kernel
Kernel mode
Hardware and Hardware Abstraction Layer (HAL)

Windows NT

Important system processes


Process Explorer - Sysinternals: www.sysinternals.com [CHRIS-85E18BC09\Administrator]
File Options View Process Find Users Help

Process	PID	CPU	Description	Company Name
System Idle Process	0	92.31		
Interrupts	n/a	3.08	Hardware Interrupts	
DPCs	n/a		Deferred Procedure Calls	
System	4			
smss.exe	552		Windows NT Session Mana...	Microsoft Corporation
csrss.exe	600		Client Server Runtime Process	Microsoft Corporation
winlogon.exe	624		Windows NT Logon Applicat...	Microsoft Corporation
services.exe	668		Services and Controller app	Microsoft Corporation
lsass.exe	680		LSA Shell (Export Version)	Microsoft Corporation
explorer.exe	1680	1.54	Windows Explorer	Microsoft Corporation

CPU Usage: 7.69% Commit Charge: 15.28% Processes: 26 Physical Usage: 44.06%

Session Manager (similar to `init`)

Client Server Runtime Process (Win32)

Windows Logon Process (`login`)

Service Control Manager (SCM)

Local security authentication (LSA) process

Windows NT

Security Components

- Security Reference Monitor (SRM)
  - kernel process
  - *performs access control decisions*
  - generates security context
- Local Security Authentication (LSA)
  - user process
  - manages security policies (permission settings)
  - user authentication
- Windows Logon
  - user process
  - gather login information

Access Control Decisions

- Object
  - Windows is object-oriented, everything is an object
  - each object has security settings (*security descriptor*)
- Subject
  - threads / processes
  - have a *security context*
- Operation
  - determines desired access (read, write, delete, …)
- Access Control Decision
  - determines whether object permits certain operations for security context
  - implemented by SRM functionality (SeAccessCheck)
  - if access is permitted, typically an object handle is returned

Security Context

- Security Context
  - stored in (access) token
  - associated with every thread / process
- Access token
  - kernel data structure that determines rights of a subject
  - important fields
    - *User SID (Security IDentifiers)*
    - *Group SIDs*
    - *Privileges*
    - Default permissions (used for files that are created)
    - Management information

Security Identifiers (SID)

- Secure Identifiers
  - used to uniquely identify entities (users, groups, …)
  - similar concept to UID/GID in Unix, but unified
  - variable length, numeric values
- Structure
  - SID structure revision number – 48-bit authority value – variable number of 32-bit sub-authority
  - Administrator has S-1-5-21-XXX-XXX-XXX-500
  - Everyone has S-1-1-0
- Administrator
  - account similar to the `root` user in Unix

Impersonation

- Impersonation
  - used to create access tokens with different permissions
  - the Windows equivalent of setuid* calls
  - can be used to elevate or drop access rights

Security Descriptors

- Security descriptor
  - security information associated with objects
  - important fields
    - owner SID
    - primary group SID (only used by POSIX)
    - discretionary access control list (DACL) – relevant for access control
    - system access control list (SACL) – relevant for logging
- Access control list
  - header + list of access control entries (ACE)

Security Descriptors

- Access control entry (ACE)
  - contains a SID (e.g., for user akaprav)
  - corresponding operations (e.g., write, read)
  - type (that specifies either allow or deny)

- ACL assignment
  - complex set of rules:

    either directly set

    or determined via “inheritance” – e.g., from the current directory

    or default taken from access token

Security Descriptors

- Access decision
  - traverse the ACL until

    either all requested permissions are granted, or

    a requested permission is denied
  - this implies that the order of the ACE might matter!
  - typically, deny entries appear first

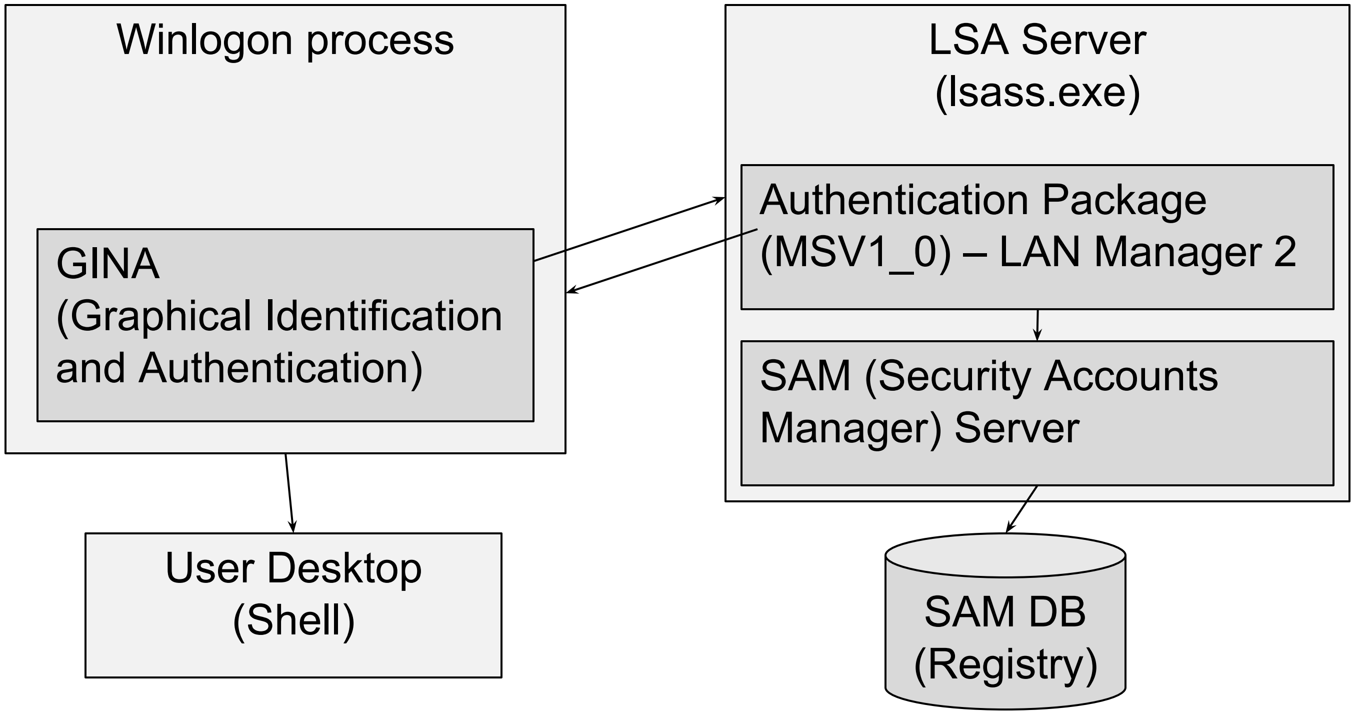
- Owner of resource always gets right to modify the ACL

- In principle, concepts are more powerful that Unix
  - permissions for many groups can be defined
  - fine-grain control via allow and deny rules possible

Privileges

- Recall that access token also stores privileges
- Privileges
  - not all (security-relevant) operations are associated with objects
    - examples: shut down computer, set system time, …
  - other privileges might disable or bypass access control checks
    - examples: backup files, debug processes, …
- Super privileges
  - some privileges are so powerful that they basically grant full access
    - “Act as part of the OS,” “Debug Program,” “Restore files” …

Authentication



SAM DB

- Stores hashed passwords
  - similar to /etc/passwd (and /etc/shadow)

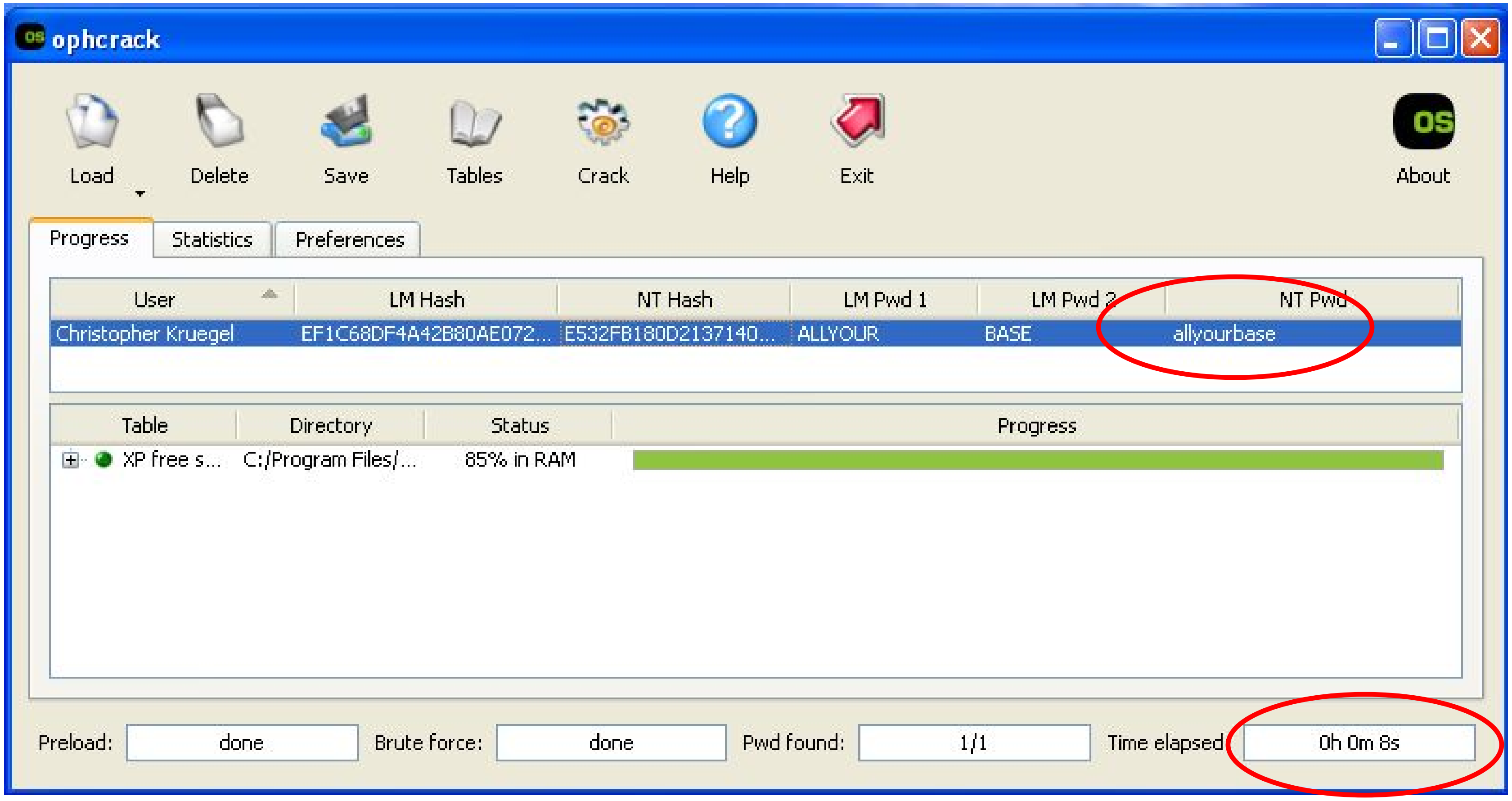
- Two formats
  - LM (LAN Manager) hash
  - NTLM

- LM hash
  - uses DES to encrypt static string
  - however, a few flaws
    - no salt
    - splits 14 characters into 2 blocks of 7 characters (hashed separately)
    - all characters converted to uppercase (further reduces key space)

SAM DB

- LM hash
  - can be cracked trivially (ophcrack)
  - disabled by default in Vista (or when password > 14 characters)
- NTLM
  - better security (MD5)
  - still no salt, thus effective rainbow table attacks possible

SAM DB



User	LM Hash	NT Hash	LM Pwd 1	LM Pwd 2	NT Pwd
Christopher Kruegel	EF1C68DF4A42B80AE072...	E532FB180D2137140...	ALLYOUR	BASE	allyourbase

Table	Directory	Status	Progress
XP free s...	C:/Program Files/...	85% in RAM	

Preload: done Brute force: done Pwd found: 1/1 Time elapsed 0h 0m 8s

Rainbow Tables

- Make brute force attacks feasible
  - ideal – store complete mapping between passwords and hashes
  - problem – too much space needed
  - rainbow tables provide space/time tradeoff
  - based on the idea of hash chains

- Hash chain
  - introduce reduction function `r` that maps hash to (some) password
  - now, we can create chains such as

$$\begin{tikzcd}
\boxed{\text{PWD1}} \arrow[r, "h"] & \boxed{\text{Hash1}} \arrow[r, "r"] & \boxed{\text{PWD2}} \arrow[r, "h"] & \boxed{\text{Hash2}} \arrow[r, "r"] & \boxed{\text{PWD3}} \arrow[r, "h"] & \boxed{\text{Hash3}}
\end{tikzcd}$$

… and we only store the start and the end (PWD1 and Hash3)

Rainbow Tables

- Password cracking
  - given `Hash` and rainbow table (assume here that `Hash` == Hash2)

$$\begin{tikzcd}\boxed{\text{PWD1}} \arrow[r, "h"] & \boxed{\text{Hash1}} \arrow[r, "r"] & \boxed{\text{PWD2}} \arrow[r, "h"] & \boxed{\text{Hash2}} \arrow[r, "r"] & \boxed{\text{PWD3}} \arrow[r, "h"] & \boxed{\text{Hash3}}\end{tikzcd}$$

  - start computing from `Hash` and check for hits in table

$$\begin{tikzcd}\boxed{\texttt{Hash}} \arrow[r, "r"] & \boxed{\text{PWD3}} \arrow[r, "h"] & \boxed{\text{Hash3}}\end{tikzcd}$$

  - if hit is found, use stored start and follow chain until element before Hash is reached

$$\begin{tikzcd}\boxed{\text{PWD1}} \arrow[r, "h"] & \boxed{\text{Hash1}} \arrow[r, "r"] & \boxed{\texttt{PWD}}\end{tikzcd}$$

Rainbow Tables

- Hash chains
  - problem – can have many collisions that remain unnoticed
    - typically, two hashes map to same password
    - happens very frequently
  - solution – rainbow tables

- Rainbow tables
  - use different reduction functions for each of the *k* steps
    this also makes collisions detectable (last entries in chain match)
  - requires to follow multiple (*k*) chains for cracking

File System

- NT File System (NTFS)
  - successor of FAT (file allocation table) file system
  - better performance, journaling support, quotas
  - supports Windows security features (in particular, access control features)
- Interesting features
  - links (since Vista, even symbolic links :-))
  - alternate data streams (ADS)
- ADS
  - adds additional streams to a file
  - original file size is not modified, and ADS are difficult to identify
  - accessed in the form of filename:streamname (e.g., text.txt:secret)
  - planned to hold meta-data
  - used by malware to hide presence

Advanced Security Features

- Generic exploit mitigation
  - address space layout randomization
  - data execution prevention (DEP)
  - secure heap manager
  - Safe SEH (structured exception handling)
- Kernel integrity
  - code integrity and driver signing
- System integrity and user-mode defenses
  - UAC – User account control
  - MIC – Mandatory integrity control (UIPI – User Interface Privilege Isolation)
  - Malware Removal Tool (MRT), updates, firewall

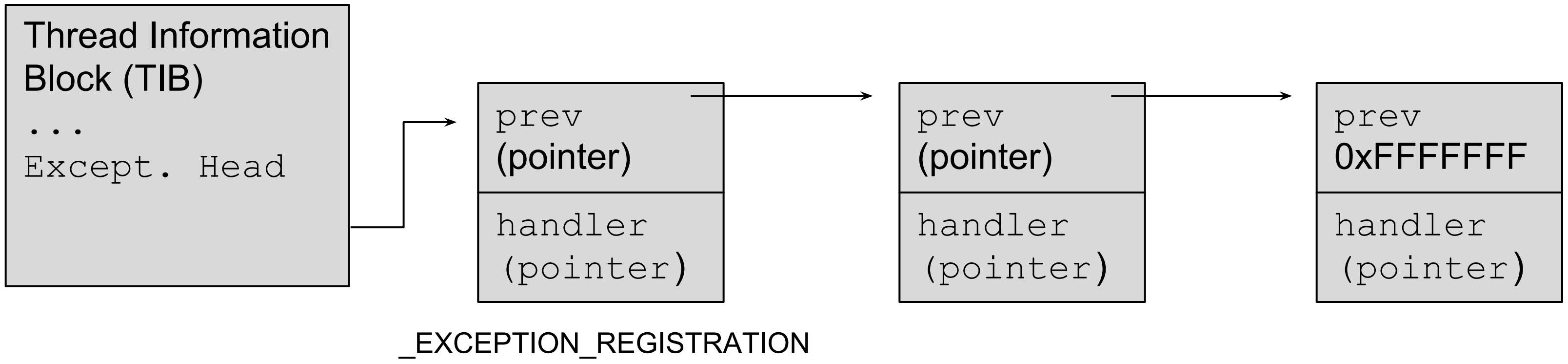
Exploit Mitigation Features

- Data execution prevention (DEP)
  - non-executable stack / heap
  - makes it more difficult for an attacker to inject shellcode
- Address space layout randomization
  - used to make it more difficult to guess addresses when performing memory corruption exploits
  - makes it harder to re-use existing (library) code
- Secure heap manager
  - block metadata randomization
  - entry integrity checking
  - randomization (rebasing)

Exploit Mitigation Features

- Safe SEH (structured exception handling)

- Structured exception handling
  - sort of signals under Windows
  - OS mechanism to react to software / hardware exceptions
  - exception handlers can be nested



Structured Exception Handling

- _EXCEPTION_REGISTRATION structures
  - where are they stored?

    on the stack!
  - nice target for attackers (since they contain function pointers)
  - overwrite the handler address and cause exception (access violation)

- Arms race
  - MS checked that exception handler is not on stack or heap
  - attackers responded by using trampolines

  - SafeSEH
    - check that exception handler target is in list of known handlers
    - requires compiler support (since handlers must be known in advance)

Kernel Integrity

- Code integrity
  - check that system binaries are not tampered with
  - allow only signed drivers to be loaded into kernel space
- Possible attack vectors
  - page file attack (since fixed)

    Windows allowed raw disk access, so attackers could tamper with memory while it was swapped out
  - buggy drivers

    exploit a bug in a signed driver

    any driver will do, since we can load it ourselves

System Integrity

- Mandatory integrity control
  - picks up on the idea of mandatory access control
  - assigns one of four integrity levels to processes
  - restricts interaction between processes of different levels
  - Internet Explorer in protected mode
- One kind of interaction is message passing
  - feature is called User Interface Privilege Isolation
    limits those processes that can send messages
    to (higher privileged) system processes
  - defends against *shatter attacks*

Shatter Attacks

- Local privilege escalation
  - abuses fact that processes can send messages to other processes' window message loop
  - impossible for destination process to authenticate sender
  - many "interesting" handler functions available
  - requires attacker to control process of interactive user (Session 0)

- Steps
  1. use message to "inject" payload into target process
  2. use message that allows to set callback (originally, WM_TIMER)
  3. callback calls your code

System Integrity

- Malware Removal Tool
  - integrated anti-malware program
- Windows firewall
  - now, also with checks on outbound connections ;-)
- Microsoft Security Development Lifecycle (SDL)
  - started in 2002 as Trustworthy Computing
  - Microsoft-wide initiative to improve software quality
  - code audits, static analysis, fuzz testing, …

Trustworthy Computing

From: Bill Gates

Sent: Tuesday, January 15, 2002 5:22 PM

To: Microsoft and Subsidiaries: All FTE

Subject: Trustworthy computing

Every few years I have sent out a memo talking about the highest priority for Microsoft. Two years ago, it was the kickoff of our .NET strategy. Before that, it was several memos about the importance of the Internet to our future and the ways we could make the Internet truly useful for people. Over the last year it has become clear that ensuring .NET is a platform for Trustworthy Computing is more important than any other part of our work. […]

Bill

Virtualization

Note on Virtualization

- Virtualization
  - has become increasingly popular
  - adds an additional layer under the operating system
  - allows to run multiple guest operating systems in user mode
  - popular solutions are VMware and Xen
  - recently, VM-based rootkits (SubVirt or Blue Pill)

- x86 virtualization problems
  - platform contains "privileged" instructions that do not trap
  - simply different behavior between user and kernel mode
  - interpret each instruction (very slow)
  - restrict / adapt operating system (Xen uses this approach)
  - dynamic recompilation (VMware)

Note on Secure Systems

- Secure system …
  … means here – only run trusted / certified applications
  - chain of trusted needed
- Secure hardware platform
  - hashed (and possibly encrypted) BIOS
  - public key stored in ROM
- Secure boot process
  - chain of verification + loading
    BIOS – root file system – OS loader – OS
- Secure OS
  - enforces limitations and verifies applications

Note on Secure Systems

- Attack vectors
  - hardware modifications
    - mod chips
  - cryptographic attacks (hashes and keys)
  - exploits in trusted code (OS code)

Your Security Zen

- iOS 0-day exploit used in a targeted attack

- CVE-2016-4657: An exploit for WebKit, which allows execution of the initial shellcode
- CVE-2016-4655: A Kernel Address Space Layout Randomization (KASLR) bypass exploit to find the base address of the kernel
- CVE-2016-4656: 32 and 64 bit iOS kernel exploits that allow execution of code in the kernel, used to jailbreak the phone and allow software installation

- Apple issued iOS 9.3.5 patch to fix these

Source: https://citizenlab.org/2016/08/million-dollar-dissident-iphone-zero-day-nso-group-uae/