WCI 442

Windows Vista System Integrity Technologies

Why?

The bad guys are everywhere!

- They literally want to do you harm
- Threats exist in two interesting places—
  - Online: system started and shows a login screen or a user is logged in
  - Offline: system is powered down or in hibernation
- Policies must address both

Cool stuff!

- Code integrity: protection against online attack
- BitLocker (secure startup): protection against offline attack
- Windows service hardening
- Mandatory integrity control
- Internet Explorer protected mode

Protect the OS When Running

The threats

- Trojan that replaces a system file to install a rootkit and take control of the computer (e.g. Fun Love or others that use root kits)
- Offline attack caused by booting an alternate operating system and attempting to corrupt or modify Windows kernel files
- Third-party kernel drivers that are not secure
- Rogue administrator who changes kernel mode code to hide other acts

Code integrity

- Validates the integrity of certain OS files
  - Implemented as a file system filter driver
  - Hashes stored in system catalog or in X.509 certificate embedded in file
- Also validates the integrity of the boot process
  - Checks the kernel, the HAL, boot-start drivers
- If validation fails, image won't load

What does it check?

- All kernel mode code (**x64 only**)
- All code loaded into a protected process
- Modules implementing cryptographic functions
- Modules loaded into the software licensing service

More on kernel mode code

x64

- All kernel mode code must be signed or it won't load
- Third-party code must be WHQL-certified or contain a certificate from a Microsoft CA
- No exceptions, period
- Applies to drivers, utilities, anything in the kernel

x32

- Signing applies only to drivers shipped with Windows
- Can control by policy what to do with third-party
- Other unsigned kernel mode code will load

More on protected processes

- Only one right now: Media Foundation
- Loaded binaries are codecs
  - Microsoft-supplied: signed by Microsoft
  - Third-party: signed by a Windows Media DRM certificate
- Affects potential playback of next-generation high definition protected content
  - Content and/or playback app control what to do in presence of unsigned kernel mode drivers

Code integrity non-goals

- Protecting from attackers with physical access
- Verifying the integrity of NTLDR
  - Requires secure startup on TPM-enabled machines
  - Requires read-only fixed media otherwise
- Supporting rebinding or hotpatching
  - These change the on-disk image
  - CI will work if patch includes updated hash
- Online checks at boot-time for revocation lists
  - Revocation list updated after boot and stored locally

Protect the OS When Not Running

The threats

- Computer is lost or stolen
  - Theft or compromise of data
  - Attack against corporate network
- Damage to OS if attacker installs alternate OS
- Difficult and time-consuming to truly erase decommissioned disks
- Existing ways to mitigate these threats are too easy for user to circumvent

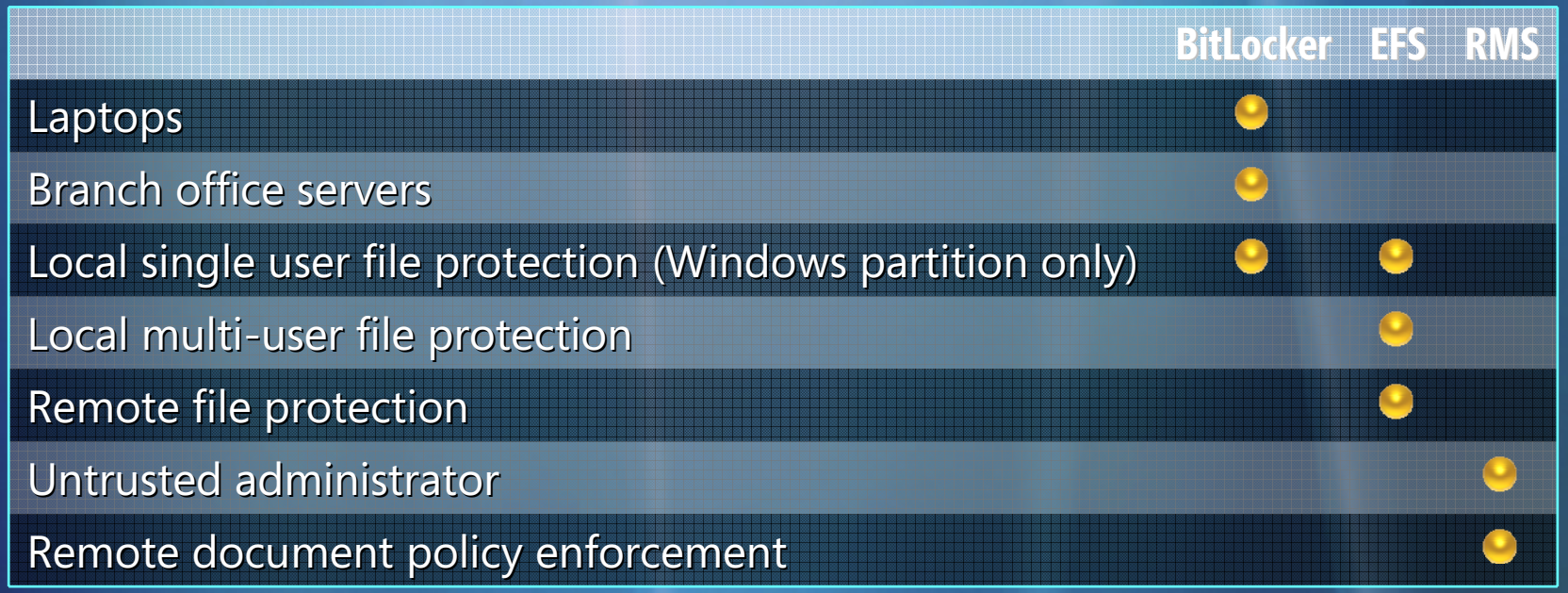
Secure startup ("BitLocker")

Ensure boot integrity	**Resilient against attack**	Protect system from offline software-based attacks
	Lock tampered systems	Prevent boot if monitored files have been altered
Protect data when offline	**Encrypt user data and system files**	All data on the volume is encrypted: user, system, page, hibernation, temp, crash dump
	Umbrella protection	Third-party apps benefit when installed on encrypted volume
Ease equipment recycling	**Simplify recycling**	Render data useless by deleting TPM key store
	Speed data deletion	Decommissioning takes seconds, not hours

Won't EFS protect me?

- Yes—for those who know what they're doing
- Users often store data on the desktop—is it EFSed?
- EFS doesn't protect the operating system
- EFS is very strong against attacks
  - Four levels of key protection
  - Properly configured, EFS is computationally infeasible to crack

Encryption scenarios



OS co-existence

- BitLocker encrypts *Windows volume only*
- You won't be able to dual-boot another OS on the same volume
- OSes on other volumes will work fine
- Data on protected volume is unavailable outside the OS
- Attempts to modify the protected Windows volume will render it unbootable

Enabling BitLocker

- Create a 1.5GB active partition
  - This becomes your "system" partition—where OS boots
  - The TPM boot manager uses only 50MB
  - Windows runs from on your "boot" partition—where the system lives
- Initialize TPM chip if you're using it
  - In management console or BIOS
- Enable BitLocker in Security Center
  - Update hard disk MBR
  - Encrypt Windows "boot" partition

Recovery options

- Useful in case of some kind of hardware failure
- It's a password; stored in different ways—
  - Removable media
  - Printed
  - Active Directory
- Also, service packs and driver upgrades trigger a loader that recomputes and reseals TPM secrets

Can use TPM 1.2 chip

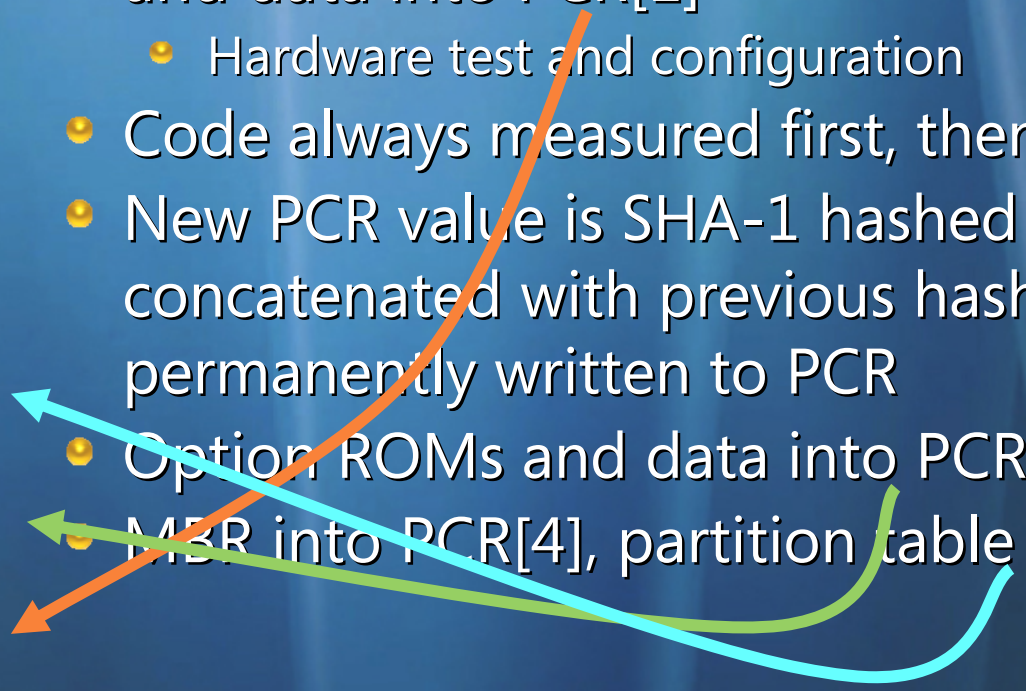
- Microcontroller affixed to motherboard
- Stores keys and digital certificates
- For BitLocker, TPM stores storage root key
  - SRK decrypts volume encryption key *only when system boots normally;* compares each boot process against previously stored measurements
  - No user interaction or visibility (unless you require a PIN or additional start-up key)
  - Recovery key can be archived in Active Directory for the inevitable "omg" moment
  - Prohibits meaningful use of software debuggers during boot

TPM architecture

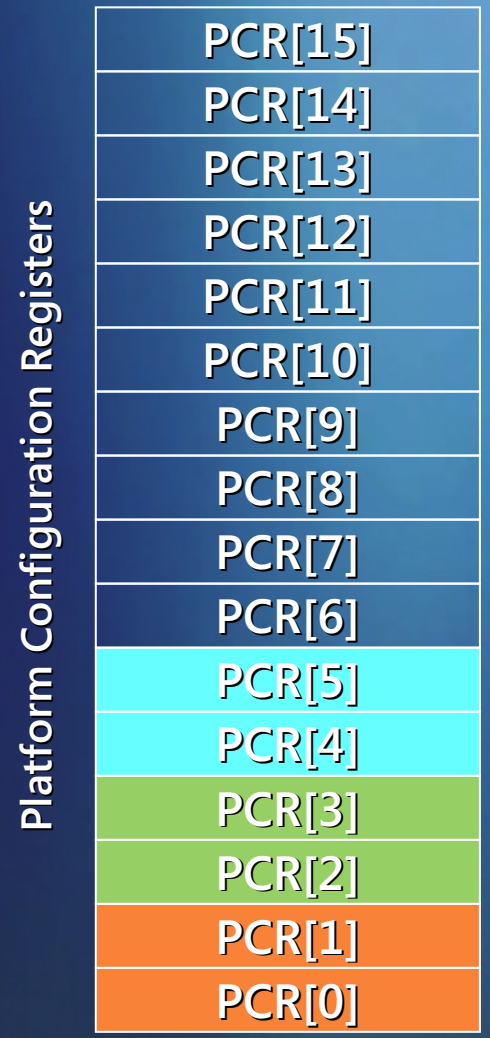
Platform Configuration Registers

PCR[15]
PCR[14]
PCR[13]
PCR[12]
PCR[11]
PCR[10]
PCR[9]
PCR[8]
PCR[7]
PCR[6]
PCR[5]
PCR[4]
PCR[3]
PCR[2]
PCR[1]
PCR[0]

- Reset all registers, transfer execution to Core Root of Trust Measurement
- Measure next stage of firmware into PCR[0] and data into PCR[1]
  - Hardware test and configuration
- Code always measured first, then executed
- New PCR value is SHA-1 hashed then concatenated with previous hash; permanently written to PCR
- Option ROMs and data into PCR[2] and [3]
- MBR into PCR[4], partition table in PCR[5]

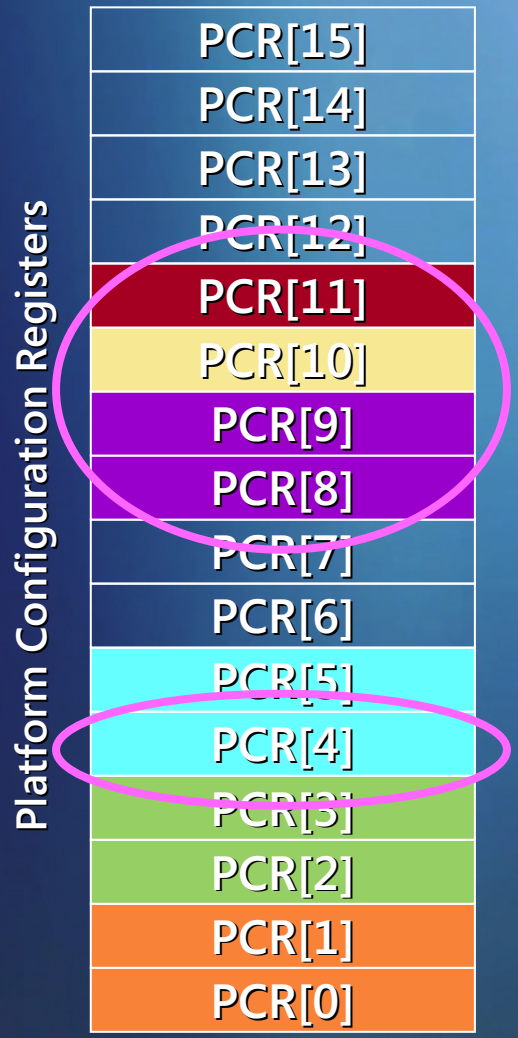


TPM architecture



- MBR takes over; loads first sector of active boot partition into memory; measures first 512 bytes into PCR[8]
- Boot sector loads; measures remainder into PCR[9] and transfers execution
- Boot code measures BOOTMGR into PCR[10] and transfers execution
- Any additional boot applications must load only from BitLocker volume
  - BitLocker keys are in PCR[11]
- Finally, BOOTMGR transfers control to operating system; OS checks integrity of all executables loaded

TPM architecture



- TPM measures all code and reports results
- Default BitLocker consumption: 4,8,9,10,11
- You can add others, with caveats
- Option ROMs in 2,3
  - Any change invalidates the PCRs
  - Includes inserting smartcard reader or USB drive
- BIOS ROMs in 0,1
  - Reflashing BIOS invalidates the PCRs

BitLocker can’t stop everything

- Hardware debuggers
- Online attacks—BitLocker is concerned only with the system’s startup process
- Post logon attacks
- Sabotage by administrators
- Poor security maintenance

Deployment considerations

- Requires hardware and software upgrades
  - Phase in, start with high priority computers
- Mostly a feature for laptops
- Also consider for desktop computers in insecure environments (factory floor, kiosk, …)
- Enterprise key management

Protect Services From Exploit

The threats

- Remember Blaster?
  - Took over RPCSS—made it write msblast.exe to file system and added run keys to the registry
- No software is perfect; someone still might find a vulnerability in a service
- Malware often looks to exploit such vulnerabilities
- Services are attractive
  - Run without user interaction
  - Many services often have free reign over the system—too much access
  - Most services can communicate over any port

Service hardening

Service refactoring	• Move service from LocalSystem to something less privileged • If necessary, split service so that only the part requiring LocalSystem receives that
Service profiling	• Enables service to restrict its behavior • Resources can have ACLs that allow the service's ID to access only what it needs • Also includes rules for specifying required network behavior

It's about the principle of least privilege—
it's good for people, and it's good for services

Refactoring

- Ideally, remove the service out of LocalSystem
  - If it doesn't perform privileged operations
  - Make ACL changes to registry keys and driver objects
- Otherwise, split into two pieces
  - The main service
  - The bits that perform privileged operations
  - Authenticate the call between them

SVCHOST group refactoring

Windows XP Service Pack 2		
LocalSystem	Wireless Configuration	RemoteAccess
	System Event Notification	DHCP Client
	Network Connections	W32time
	COM+ Event System	Rasman
	NLA	Browser
	Rasauto	6to4
	Shell Hardware Detection	Help and Support
	Themes	Task Scheduler
	Telephony	TrkWks
	Windows Audio	Cryptographic Services
	Error Reporting	Removable Storage
	Workstation	WMI Perf Adapter
	ICS	Automatic updates
	BITS	WMI
		App Management
		Secondary Logon
Network Service	DNS Client	
Local Service	SSDP	
	WebClient	
	TCP/IP NetBIOS helper	
	Remote Registry	

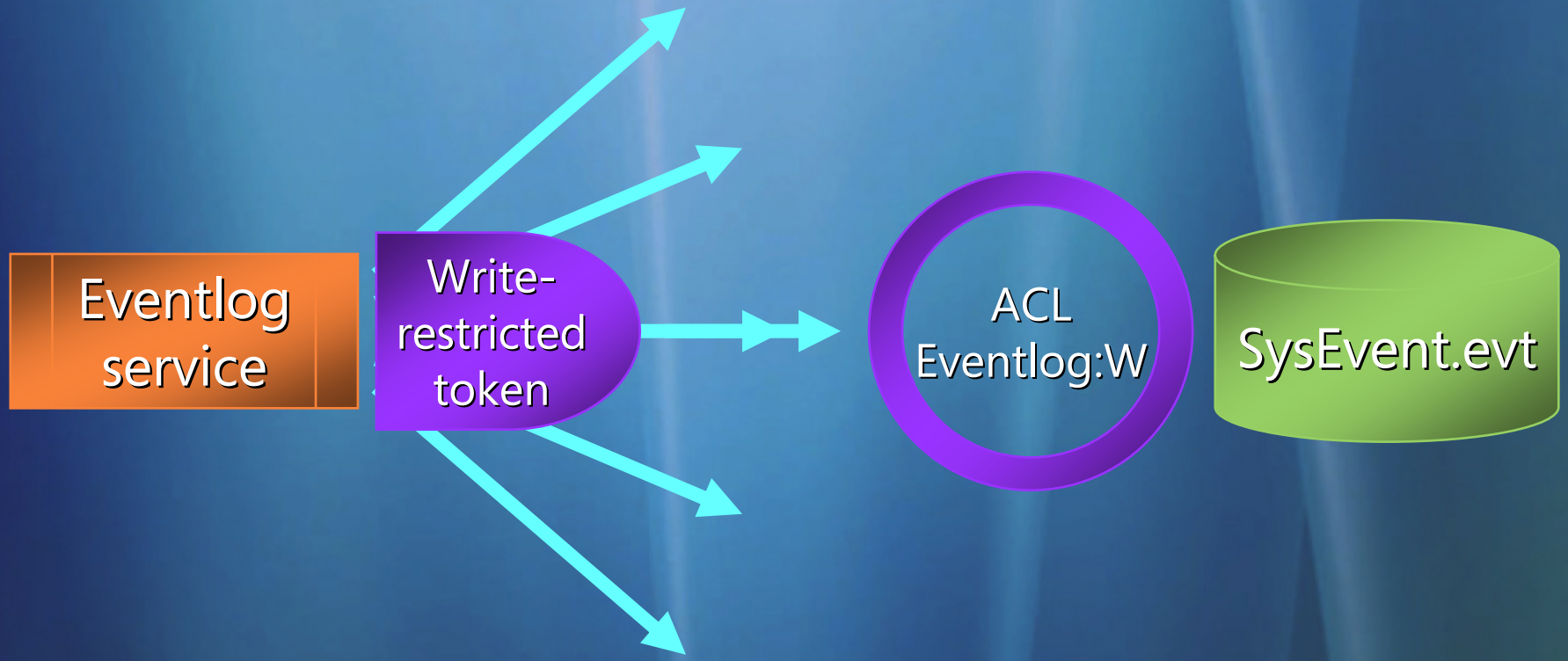
Windows Vista		
LocalSystem *Network restricted*	Removable Storage	WMI
	WMI Perf Adapter	App Management
	Automatic updates	Secondary Logon
	TrkWks	
LocalSystem *Demand started*	BITS	
Network Service *Restricted*	DNS Client	Browser
	ICS	6to4
	RemoteAccess	Task scheduler
	DHCP Client	IPSEC Services
	W32time	Server
	Rasman	Cryptographic Services
	NLA	
Local Service *Restricted* *No network access*	Wireless Configuration	Network Connections
	System Event Notification	Rasauto
	Shell Hardware Detection	Themes
		COM+ Event System
Local Service *Restricted*	Telephony	Error Reporting
	Windows Audio	Event Log
	TCP/IP NetBIOS helper	Workstation
	WebClient	Remote Registry
		SSDP

Profiling

- Every service has a unique service identifier called a "service SID"
  - S-1-80-<*SHA-1 hash of logical service name*>
- A "service profile" is a set of ACLs that—
  - Allow a service to use a resource
  - Constrain the service to the resources it needs
  - Define which network ports a service can use
  - Block the service from using other ports
- Now, service can run as LocalService or NetworkService and still receive additional access when necessary

Restricting services

Example: event log



Restricting services: know this

- A restrictable service will set two properties (stored in the registry)—
  - One to indicate that it can be restricted
  - One to show which privileges it requires

Note! This is a voluntary process. The service is choosing to restrict itself. It's good development practice because it reduces the likelihood of a service being abused by malware, but it isn't a full-on system-wide restriction mechanism. Third-party services can still run wild and free…

Network enforcement scenarios

No ports	Services that neither listen nor connect
Fixed ports	Services that listen or send on known fixed ports should be constrained to those ports only
Configurable ports	Administrator configures port in service's administration UI; network rules and firewall automatically update their own configurations
Dynamic ports	Services that listen or send on dynamically-allocated ports

Auditing

- Management events
  - Initial rules configuration
  - Rule changes
  - Rule deletions
- Enforcement events
  - Traffic allowed
  - Traffic denied

Interaction with host firewalls

- Configuration changes implemented immediately
- Rules can't be disabled by WF or third-party
- Rules can't be stopped while services are running
- For dynamic ports, netenf pushes configuration to WF

Example rules

Block any network access for BFE

"V2.0; Action=Block; App=%windir%\System32\svchost.exe; Svc=bfe;
Name=Block any traffic to and from bfe;"

Allow outbound PolicyAgent traffic

"V2.0; Action=Allow; Dir=Out; RPort=389; Protocol=tcp; Protocol=udp;
App=%windir%\System32\svchost.exe; Svc=PolicyAgent;
Name=Allow PolicyAgent tcp/udp LDAP traffic to AD;"

"V2.0; Action=Block; App=%windir%\System32\svchost.exe; Svc=PolicyAgent;
Name=Block any other traffic to and from PolicyAgent;"

Allow inbound/outbound traffic to Rpcss

"V2.0; Action=Allow; Dir=Out; RPort=135; Protocol=tcp; Protocol=udp;
App=%windir%\System32\svchost.exe; Svc=rpcss;
Name=Allow outbound rpcss tcp/udp traffic;"

"V2.0; Action=Allow; Dir=in; LPort=135; Protocol=tcp; Protocol=udp;
App=%windir%\System32\svchost.exe; Svc=rpcss; Name=Allow inbound tcp/udp rpcss;"

"V2.0; Action=Block; App=%windir%\System32\svchost.exe; Svc=rpcss;
Name=Block any other traffic to and from rpcss;"

Protect the OS and Data from Unknown Code

The threats

- A user unknowingly runs code from an unknown source that attempts to modify or delete files
- Code running as LUA attempts a local elevation of privilege by injecting code into a process running as administrator
- Trojans that attempt to execute with full administrator privilege
- System code reads data from the Internet (an untrustworthy source) that contains corrupt data designed to elevate privilege by exploiting a bug

Mandatory integrity control

- Method to prevent low-integrity code from modifying high-integrity code
  - Protect TCB files and data from modification by privileged users
  - Protect user data from modification by unknown malicious code
  - Protect processes running as privileged user from modification by processes running as standard user under the same user SID
- Classical computer security concept known since the 1970s
  - Lots of recent work in various operating systems

Don’t confuse with code integrity

CI
- Verifies code during module loading

MIC
- Implements a type of information flow policy
- Implements an enforcement mechanism
- Integrity level changes trigger a security audit event

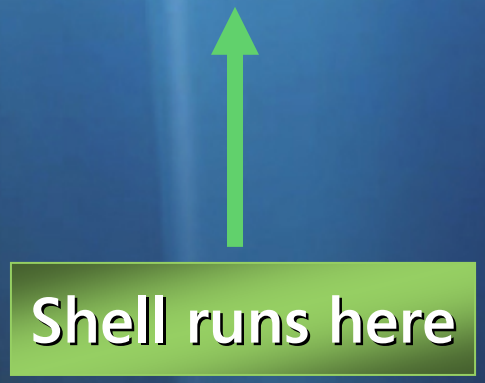
Mandatory integrity control policy is based on **trustworthiness**. Subjects with **low** degrees of trustworthiness can’t change data of a **higher** degrees. Subjects with **high** degrees of trustworthiness can’t be forced to rely on data of **lower** degrees.

The limitations of DACLs

- No protection of system stability
  - Third-party installers redistribute system binaries
  - Want to stop this, even if run by administrator
- No protection from tricky software
  - Non-savvy users can be convinced to install malware
  - Runs with full capabilities of user
- Weakens power of UAC
  - Can’t distinguish limited version from full (possibly administrator) version of user
  - Both versions have same user SID

Defined integrity levels

System	High	Medium	Low	Untrusted
0x4000	**0x3000**	**0x2000**	**0x1000**	**0**
Local System	Local Service Network Service Elevated (full) user tokens	Standard user tokens Authenticated Users	World (Everyone)	Anonymous



MIC expression

- Add an integrity SID to a user token at logon
  - S-1-16-<*level*>
  - Announces the integrity level of the token
  - Determines level of access the token can achieve
  - Possible second SID used by Secure Desktop to determine protection ring of an application
- Store integrity SID in the SACL of every object's security descriptor (user-created and OS)
  - Specifies the integrity level of the object

Checking MIC level

- During access check, verify the user passes integrity check against an object for write access
  - However, can add ACE to DACL to deny read access to low integrity users *(more on this later)*
- User must *dominate* object to obtain write access
  - User/process level >= object level
  - All users pass integrity check for reading and executing
- MIC trumps DACL
  - If the DACL lets you write, but you don't dominate the object, your write fails

Consider four scenarios

An attachment arrives in mail. While saving, file is written with **low** integrity. When executed, it runs at **low** integrity and can't write to user's data. *MIC prevents process from performing capabilities at user's level.*

IE downloads file from site in Internet zone. IE process that writes file to TIF runs at **low** integrity; thus file is receives **low** integrity. *MIC doesn't trust content or code from the Internet.*

A malicious program is running at **standard** user X and attempts to open process running as **privileged** user X for write, to bypass UAC and execute code will full privileges. *MIC stops this because desired access is write.*

Admin (IL=**high**) runs downloaded program. Process runs as **standard** admin (IL=**medium**). *MIC prevents processes from write-accessing resources ACLed for the administrator.*

Processes also affected

- When user launches .EXE, process receives lower of user's or file's integrity level (if it has one)
  - Process never runs higher than file, regardless of IL of user who started it
  - Protects even administrators from malicious actions of downloaded code
  - Also protects any user data, whose level is typically that of the user—it's higher than the code
- Controlled by AIS (app installer service)
  - Check ILs of user and file
  - Adjust process IL accordingly
  - Impersonate user with correct IL and continue creation

Modifying integrity levels

- Token can lower its own level
  - Not reversible
  - Only a TCB caller can raise
- Secure Input
  - Default: UI ring SID = object integrity SID
  - TCB caller can elevate token UI ring
  - Typically necessary for accessibility utilities—can now control UI but not bypass MIC control of object access

But I want to administer my box!

- Full privilege tokens, including members of the local Administrators group, are controlled by MIC
  - Can't delete files if their level is system
  - Can't lower the level of objects or files
- Built-in "Administrator" account has an additional privilege
  - Grants caller access to object
  - Could grant to other users, but be careful!
  - Granting and use of privilege is audited

Denying read access

- Can use deny ACE to prevent lower level principals from reading or executing higher level objects
- Good for administrator programs
  - Set IL to high
  - Add deny ACE for anything with a lower IL
  - Prevents malware running at lower level from attempting to call admin tools

Unlabeled objects

- System assumes default MIC of medium during access check
- Prevents untrustworthy code running at low from modifying unlabeled objects
  - Regardless of DACL
- OS files are unlabeled
  - Protected from modification with an ACL
- Objects without a SID have no MIC consideration

Non-goals

- Provide for confidentiality of data
  - This is the Bell-LaPadula model
  - Although with no-read-up ACEs, you can use MIC to achieve similar behavior
- Prevent high IL processes from reading data at a lower IL if the policy allows that
- Implement dynamic integrity
- Prevent offline attacks through modifications of ILs on files
  - But BitLocker could help here…

Protect the OS from the Internet

The threats

- Alas, most Windows users still run as admin
  - Meaning: the Internet runs as admin on your PC!
- “Drive-by” installs of spyware and virus code
- Exploits of vulnerabilities give attackers full remote access
- Even non-admins still vulnerable to malicious destruction of personal data

Internet Explorer protected mode

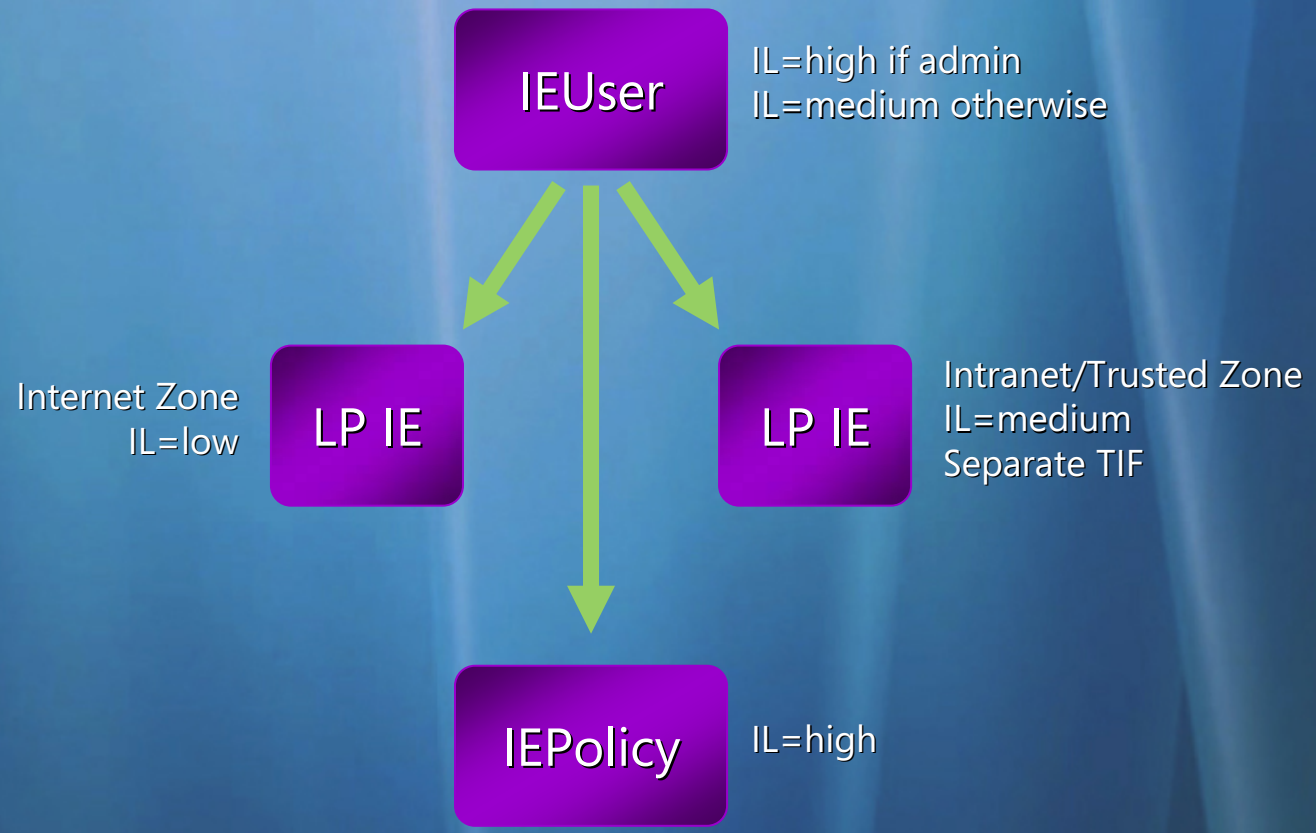
- Built on mandatory integrity control
  - Internet Explorer runs at low integrity level
- Reduce the severity of threats to IE add-ons
- Eliminate the silent install of malicious code through software vulnerabilities
- Preserve compatibility whenever possible
- Provide the capability and guidance for add-ons to restore functionality
- Minimize required user involvement
- Sometimes called “low-rights IE”

Protected mode summary

- Restricts IE from writing outside of the Temporary Internet Files (TIF) folder
  - IE's process has lower write privileges than LUA
  - It builds on the Mandatory Integrity Control (MIC) which restricts writes to higher integrity folders
- Protected mode uses COM to call two new broker processes which allow IE to write outside of the TIF
- A compatibility layer allows add-ons to elevate

This is not a "sandboxing" technology. IE is refactored into a multi-process application, with varying ILs for each process.

Refactoring IE



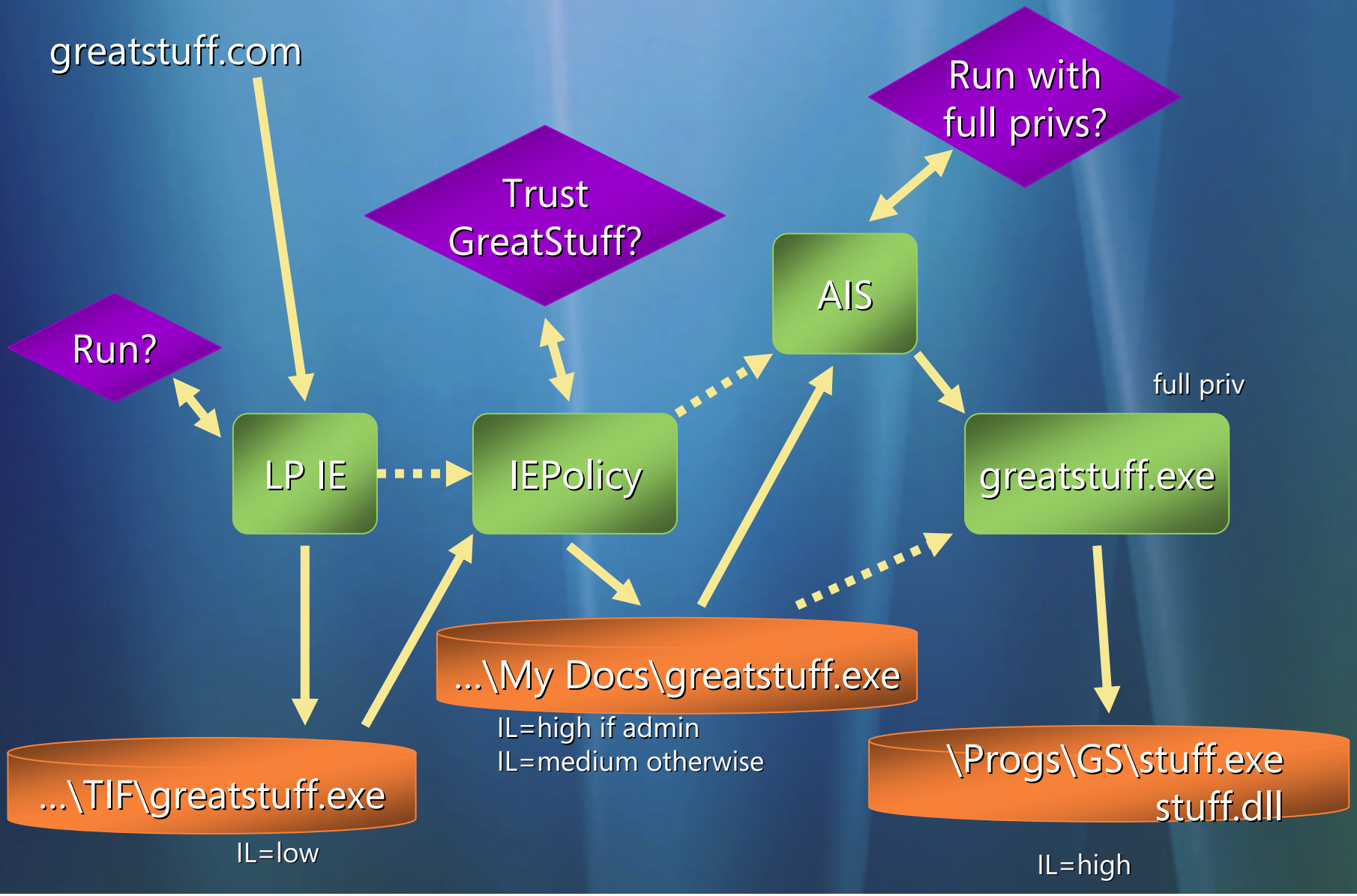
- Again: the principle of least privilege
- Refactoring at the process level—more efficient and less expensive than a virtual machine

Components and zones

Operation	Requirements	Process
URL navigation and HTML rendering	Least privilege Low integrity	LP IE
Managing user-controlled settings	Least privilege Medium integrity	IEUser
Enforcing policy in downloaded code Initiating execution	Full privilege High integrity	IEPolicy (service)

Operation	LP IE low	LP IE medium
Files downloaded in zone	Low IL	Medium IL
Modify outside TIF	No	Yes
Interact with other apps on desktop	No	Yes
Inject DLL and create remote thread	No	Yes
Renders HTML files in local zone	Yes	Yes

Installing from the Web



In-proc compatibility layer

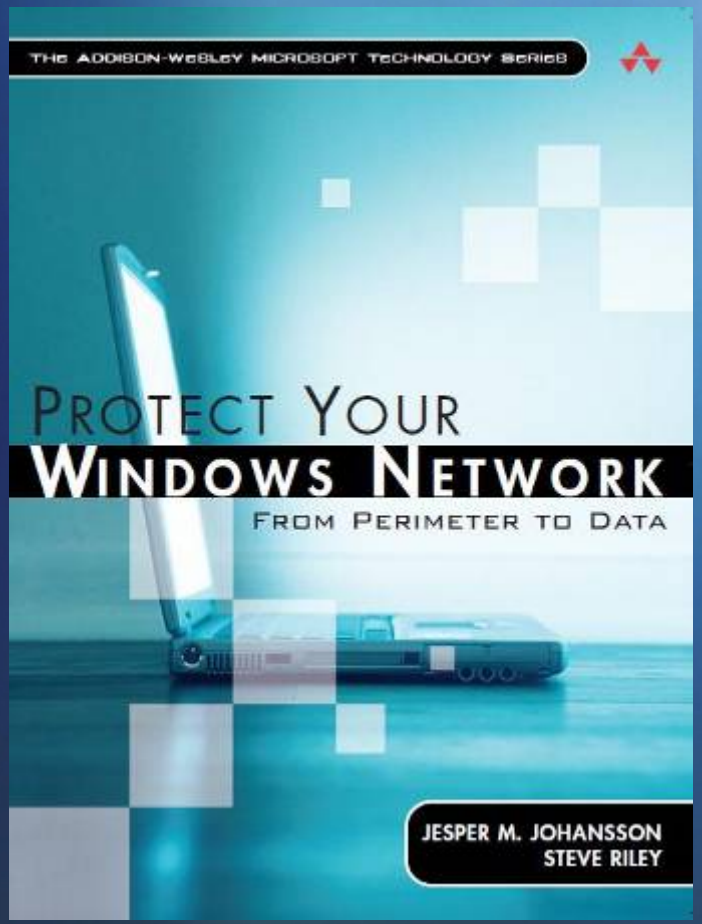
- Redirects file and registry key writes to new low integrity locations—
  - `HKCU\Software\Microsoft\Internet Explorer\Low Rights\Virtual`
  - `Documents and Settings\%user profile%\Local Settings\Temporary Internet Files\Virtual`
- Added to the location IE is trying

If IE tries to write here…	*…it gets redirected here*
`HKCU\Software\FooBar`	`HKCU\Software\MS\IE\Low Rights\Virtual\Software\FooBar`
`C:\Documents and Settings\%user profile%\FooBar`	`C:\Documents and Settings\%user profile%\Local Settings\Temporary Internet Files\Virtual\FooBar`

Steve Riley

steve.riley@microsoft.com

http://blogs.technet.com/steriley



www.protectyourwindowsnetwork.com

Thanks very much!

Microsoft®

Your potential. Our passion.™