SIEMENS

Corporate Technology

Dealing with Unreliable Software: Exile, Jail, and other Sentences

Bernd Grobauer, Siemens CERT
Heiko Patzlaff, Siemens CERT
Martin Wimmer, Siemens CERT

Trust is the basis of good working relationships – or is it?

Stuff I work with:

☑

☑

☑

☑

Stuff I trust:

□

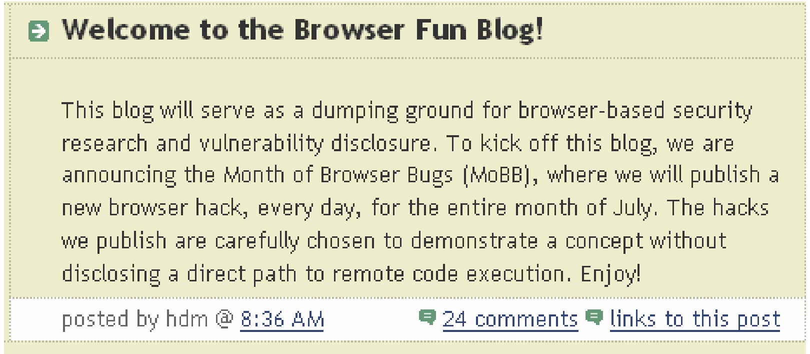
□

□

☑

- Complex and extensible through plugins
- Vulnerability-ridden
- Very much in focus of attacker
- Frequently process content downloaded from Internet

Complexity: Fuzzing Input/Protocol Complexity


Welcome to the Browser Fun Blog!

This blog will serve as a dumping ground for browser-based security research and vulnerability disclosure. To kick off this blog, we are announcing the Month of Browser Bugs (MoBB), where we will publish a new browser hack, every day, for the entire month of July. The hacks we publish are carefully chosen to demonstrate a concept without disclosing a direct path to remote code execution. Enjoy!

posted by hdm @ 8:36 AM 24 comments links to this post

# of Vulns found during MoBB:

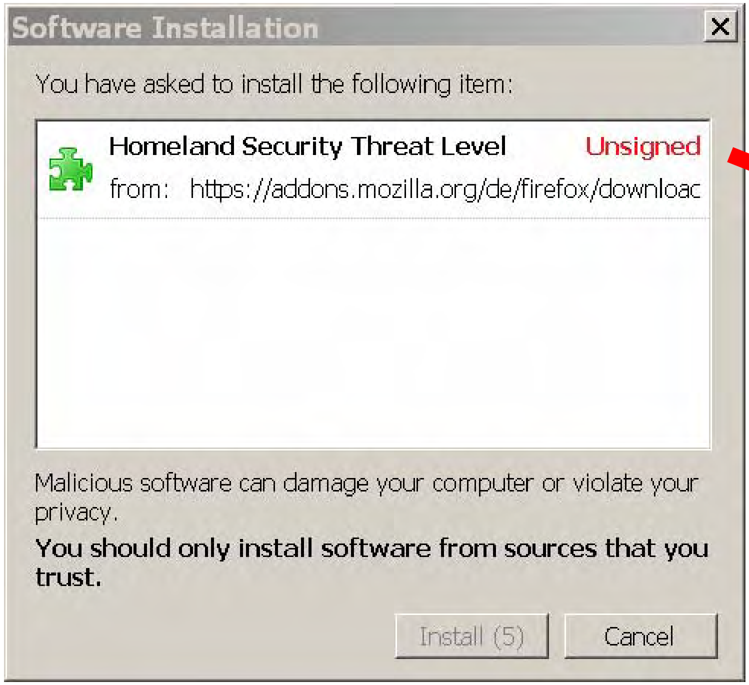
MSIE:	25
Apple Safari:	2
Mozilla:	mrw2
Opera:	1
Konqueror:	1

Slide 3

mrw2 indent?

Martin Wimmer; 11.06.2007

Extensibility: The next big headache ...



Software Installation

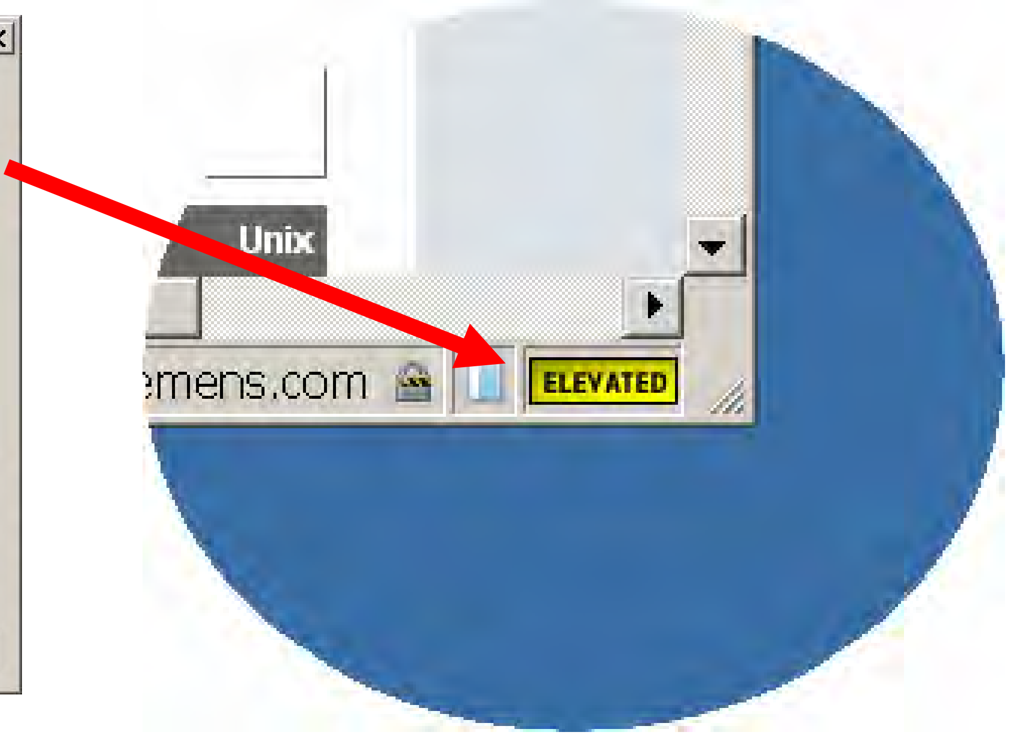
You have asked to install the following item:

Homeland Security Threat Level Unsigned

from: https://addons.mozilla.org/de/firefox/downloac

Malicious software can damage your computer or violate your privacy.

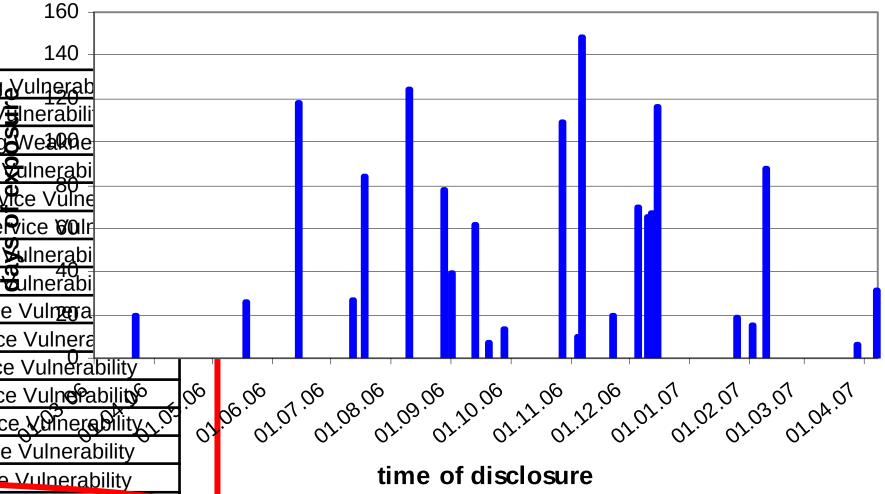
You should only install software from sources that you trust.



Vulnerabilities over Vulnerabilities

Zero-day exploits of MS products 06/07

~~(excerpt, source [eEye07])~~



🟩	15.03.2007	Microsoft Internet Explorer "navcancl.htm" Cross Site Scripting and Phishing Vulnerab
🟩	26.02.2007	Microsoft Internet Explorer UTF-7 Charset Inheritance Cross-Site Scripting Vulnerabili
🟩	23.02.2007	Microsoft Internet Explorer "onunload" Event Handling Address Bar Spoofing Weakne
🟩	13.10.2006	Microsoft PowerPoint Invalid Container Object Client-Side Denial of Service Vulnerabi
🟩	07.08.2006	Microsoft Windows GDI Library WMF Image Handling Remote Denial of Service Vulne
🟩	25.07.2006	Microsoft Internet Explorer Native Function Iteration Client-Side Denial of Service Vulr
🟩	25.07.2006	Microsoft Internet Explorer Forms ListBox and ComboBox Denial of Service Vulnerabi
🟩	25.07.2006	Microsoft Internet Explorer "ASFSourceMediaDescription" Denial of Service Vulnerabi
🟩	25.07.2006	Microsoft Internet Explorer HTML Help Control "HHCtrl.ocx" Denial of Service Vulnera
🟩	21.07.2006	Microsoft Internet Explorer "Content-Type" Header Handling Denial of Service Vulnera
🟩	21.07.2006	Microsoft Internet Explorer CEnroll Object Handling Remote Denial of Service Vulnerability
🟩	21.07.2006	Microsoft Internet Explorer Outlook View Control Client-Side Denial of Service Vulnerability
🟩	19.07.2006	Microsoft Internet Explorer Office Web Components Remote Denial of Service Vulnerability
🟩	17.07.2006	Microsoft Internet Explorer DirectX Image Transform Object Denial of Service Vulnerability
🟩	17.07.2006	Microsoft Internet Explorer "MHTMLFile" Object Client-Side Denial of Service Vulnerability
🟥	15.07.2006	Microsoft PowerPoint Presentation Handling Multiple Memory Corruption and DoS
🟩	15.07.2006	Microsoft Internet Explorer "FolderItem" Object Access Remote Denial of Service Vulnerability
🟩	13.07.2006	Microsoft Internet Explorer DirectX Image Transform Object Denial of Service Vulnerability
🟩	12.07.2006	Microsoft Internet Explorer "TriEditDocument" Object Remote Denial of Service Vulnerability
🟩	11.07.2006	Microsoft Internet Explorer HTML Editing Component Denial of Service Vulnerability
🟩	10.07.2006	Microsoft Internet Explorer DirectX Transform Control Denial of Service Vulnerability
🟩	09.07.2006	Microsoft Office Object Library "LsCreateLine()" Improper Memory Access Vulnerability
🟩	09.07.2006	Microsoft Internet Explorer "DirectAnimation" Control Denial of Service Vulnerability
🟩	09.07.2006	Microsoft Internet Explorer Remote Data Service Object Denial of Service Vulnerability
🟩	07.07.2006	Microsoft Internet Explorer "appendChild()" Client-Side Denial of Service Vulnerability
🟩	06.07.2006	Microsoft Internet Explorer Structured Graphics Control Denial of Service Vulnerability
🟩	03.07.2006	Microsoft Internet Explorer Data Access ActiveX Remote Denial of Service Vulnerability
🟨	06.06.2006	Microsoft Internet Explorer Keystroke Events Handling Arbitrary File Upload Issue
🟧	27.04.2006	Microsoft Internet Explorer ActiveX Control Dialog Box Security Bypass Vulnerability
🟨	27.04.2006	Microsoft Products "mhtml" Cross Domain Information Disclosure Vulnerability
🟩	26.04.2006	Microsoft Office 2003 "mailto:" URI Handler Arbitrary File Attachment Weakness
🟨	13.02.2006	Microsoft Internet Explorer Drag and Drop Events Timing Vulnerability
🟨	06.02.2006	Microsoft HTML Help Workshop Multiple File Handling Buffer Overflow Vulnerabilities

≈54 days of exposure for 25 samples in 2006 / start 2007

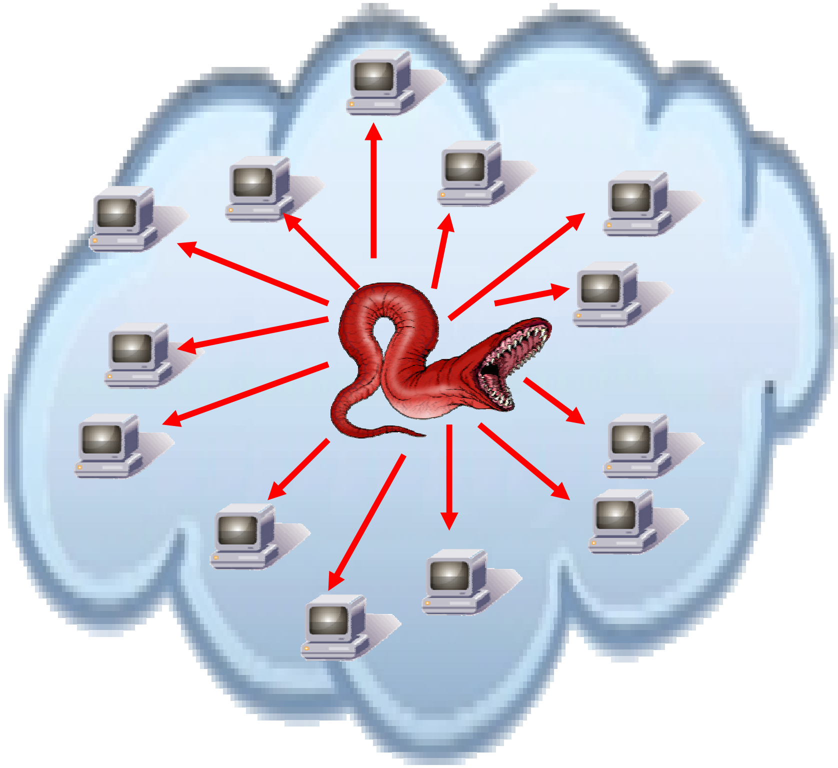
Critical vulnerability, unpatched for 300 days

33 unpatched
IE and Microsoft Office vulnerabilities

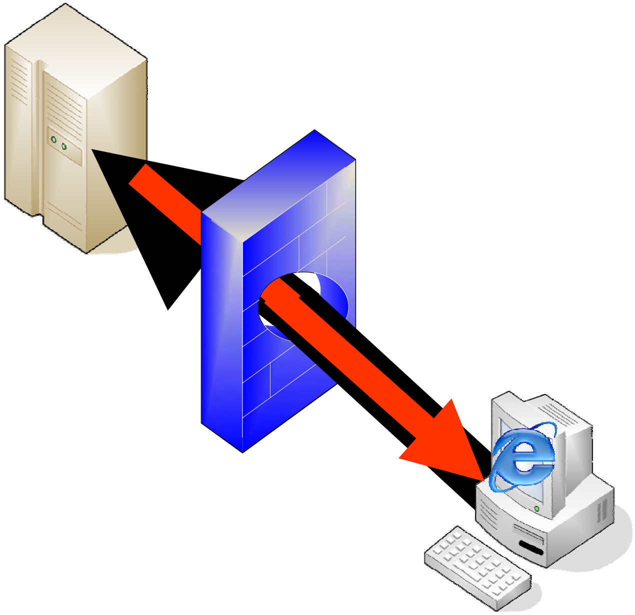
Date: 05/10/2007
Source: [FrSIRT07]

Attacker-Focus

Then: the good old times of scanning worms



Now: Drive-by infections via browsers & tampered documents



What to do

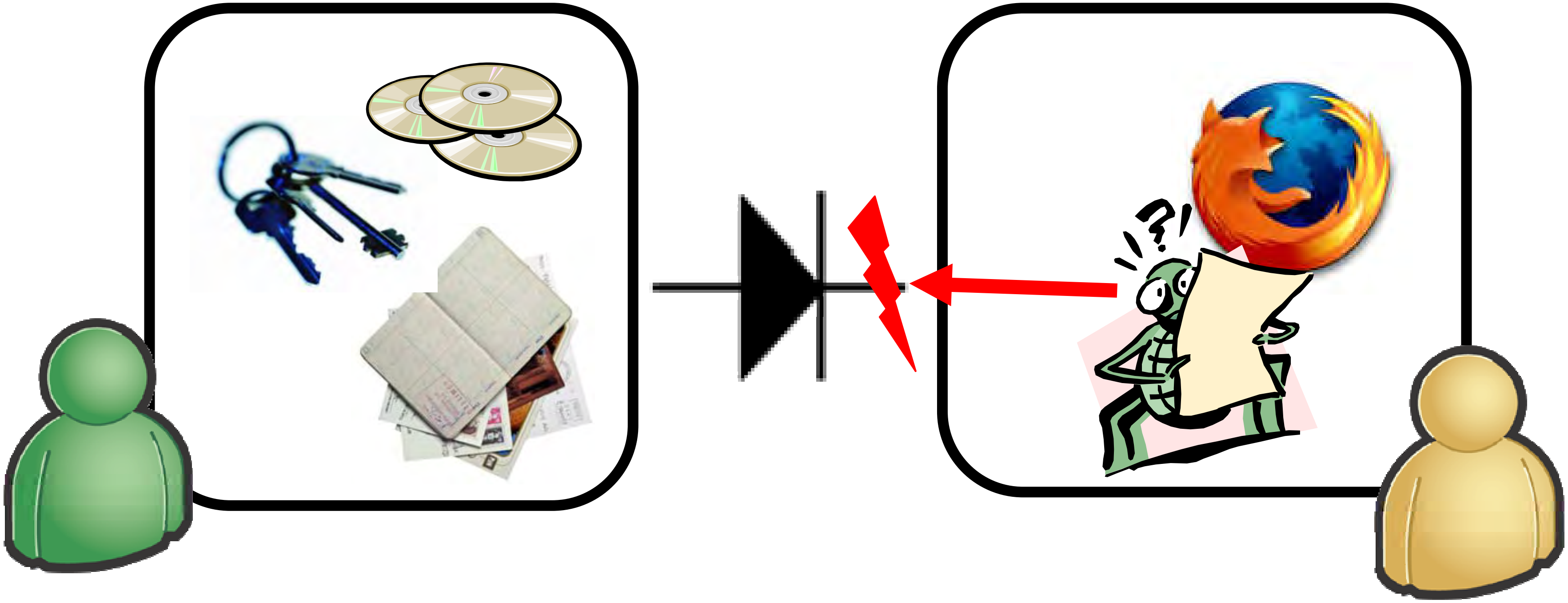
- Traditional security mechanisms
  - Patch systems
  - Virus scanners
  - Firewalls
  - Host intrusion prevention systems

Insufficient w.r.t. zero-day exploits

- Separation / Isolation of critical systems
  - Do not hinder vulnerabilities to be exploited
    … but: **restrict their impact**!

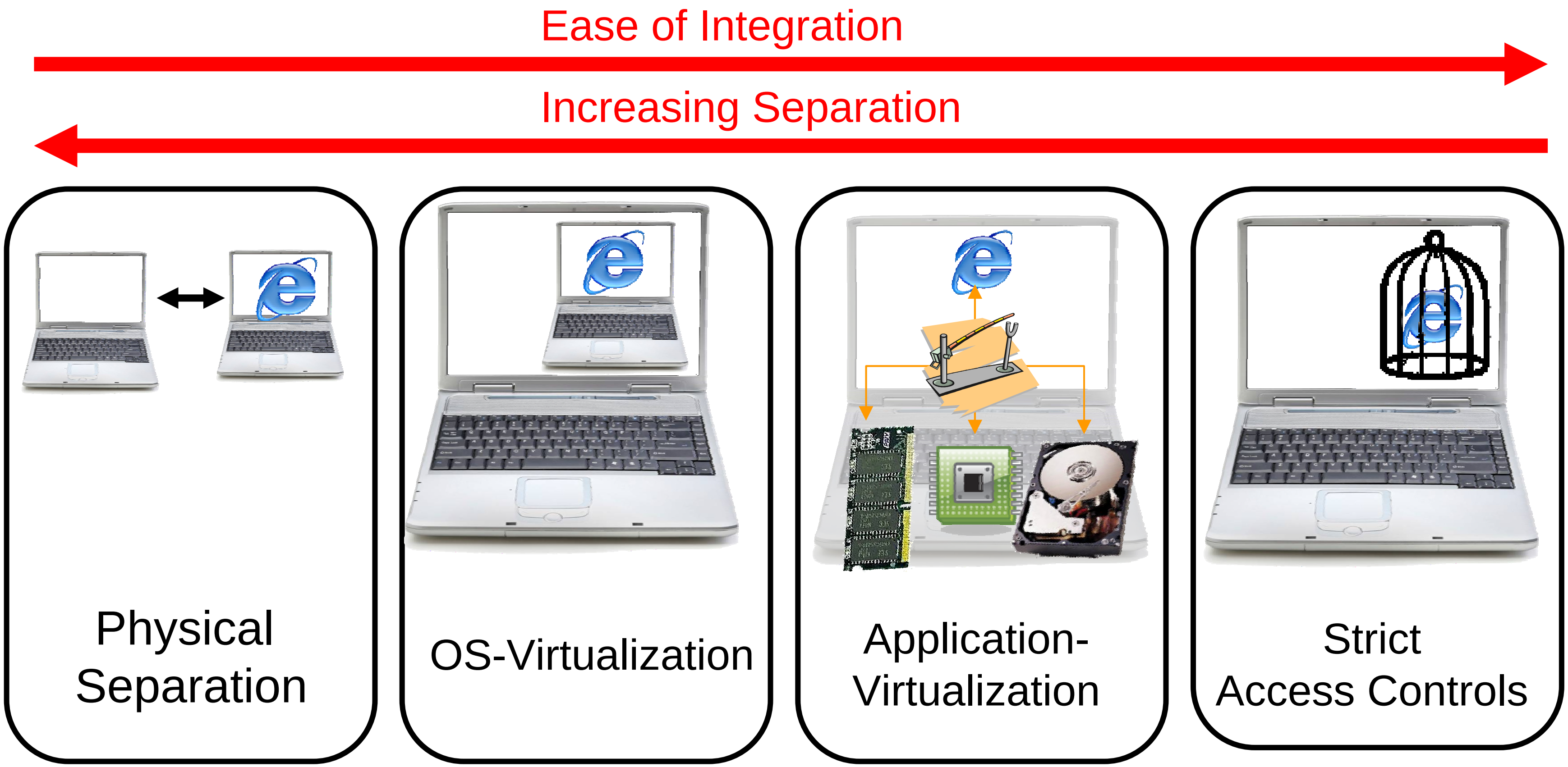
Poor man‘s separation

- Start browser as different user with limited rights



- What other methods of separation are there?
- How do they affect integration/usability?

Degrees of Separation



Ease of Integration

Increasing Separation

Physical Separation

OS-Virtualization

Application-Virtualization

Strict Access Controls

Physical Separation

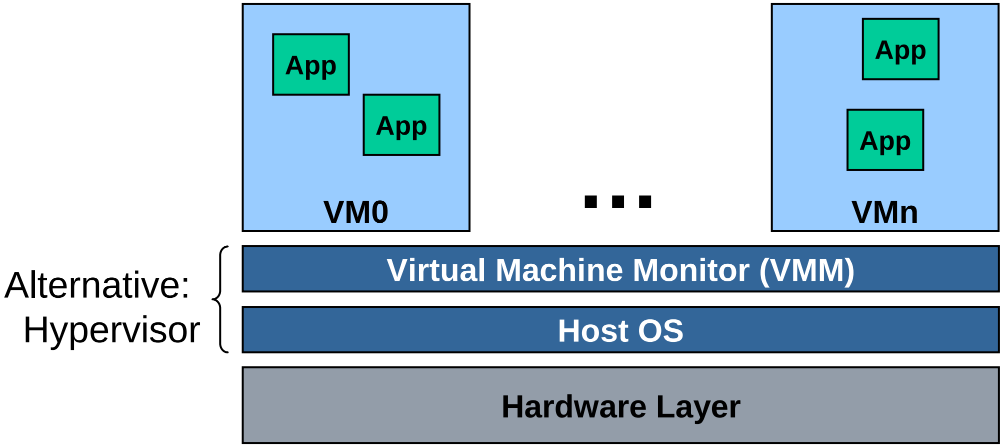
- Basic Idea: Run untrusted client software on dedicated system, enable remote access for users

- Enabling Technologies:
  - Terminal-Server Solutions (RDP, Citrix, ...)
  - Windows-Forwarding (X11)

- Integration/Usability Issues:
  - works only with network access to server
  - how to download/upload data
  - how to view data (where are the viewer applications located?)
  - may be cumbersome to use (cut&paste, ...) (depending on used technology)
- Use Case:
  - Providing tightly controlled Internet-WWW-connectivity in high-security environment

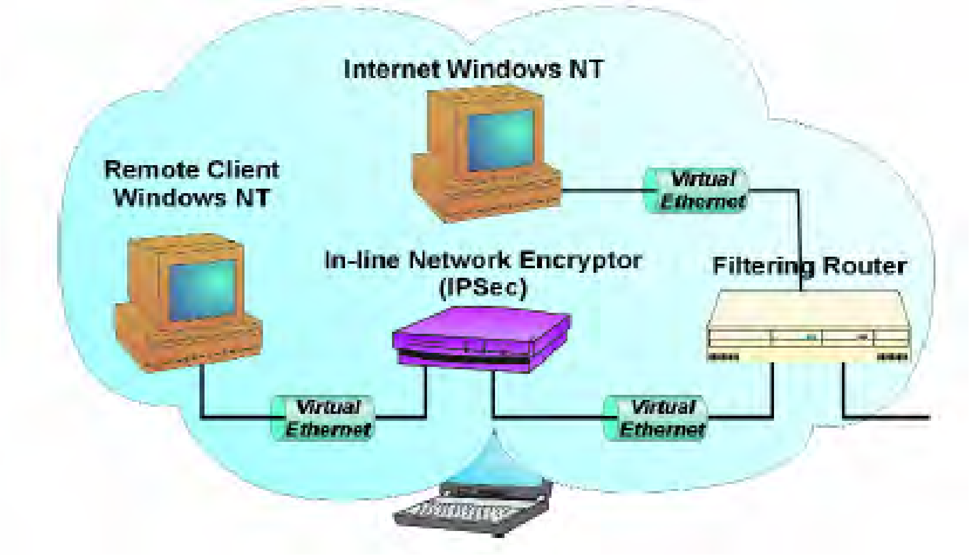
OS Virtualization

- Allows running more than one operating system on the same hardware simultaneously



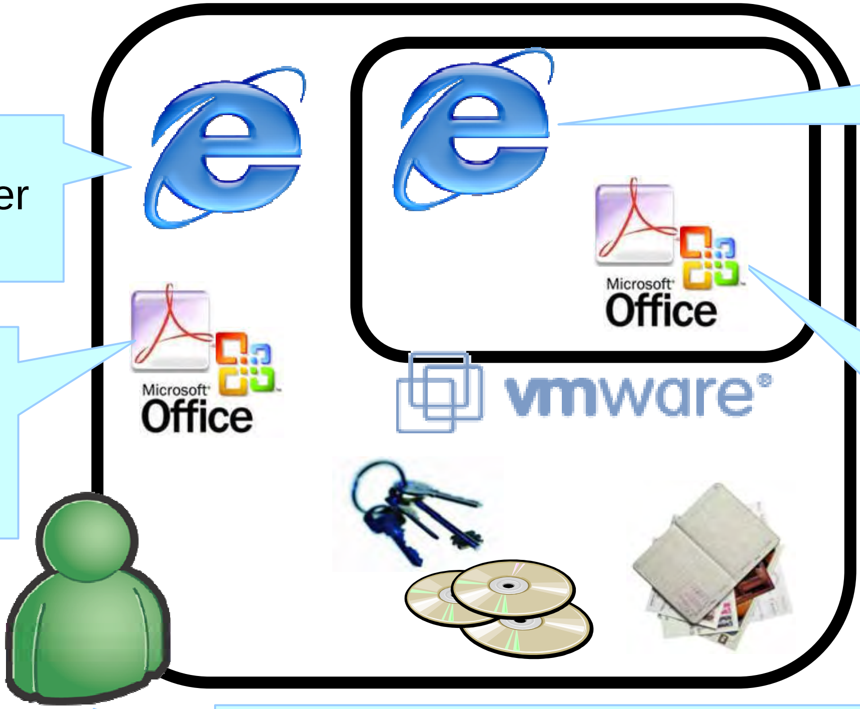
The NSA NetTop Project (1999-2000)

- Project envisioned use of virtualization technology to
  - provide additional layer of security to COTs components
  - „unclutter“ desktop by putting several devices (filter component, encryption component, different clients for different security levels) on one box
- Commercialized as HP „NetTop“



(Source: Meushaw, R. & Simard D., „NetTop“, Tech Trend Notes, Volume 9, Edition 4)

Our Experiment: Secure yet User-Friendly Browsing (I)



Open trustworthy webpages in „real“ user environment

Open downloaded trustworthy document types with applications on „real“ user environment

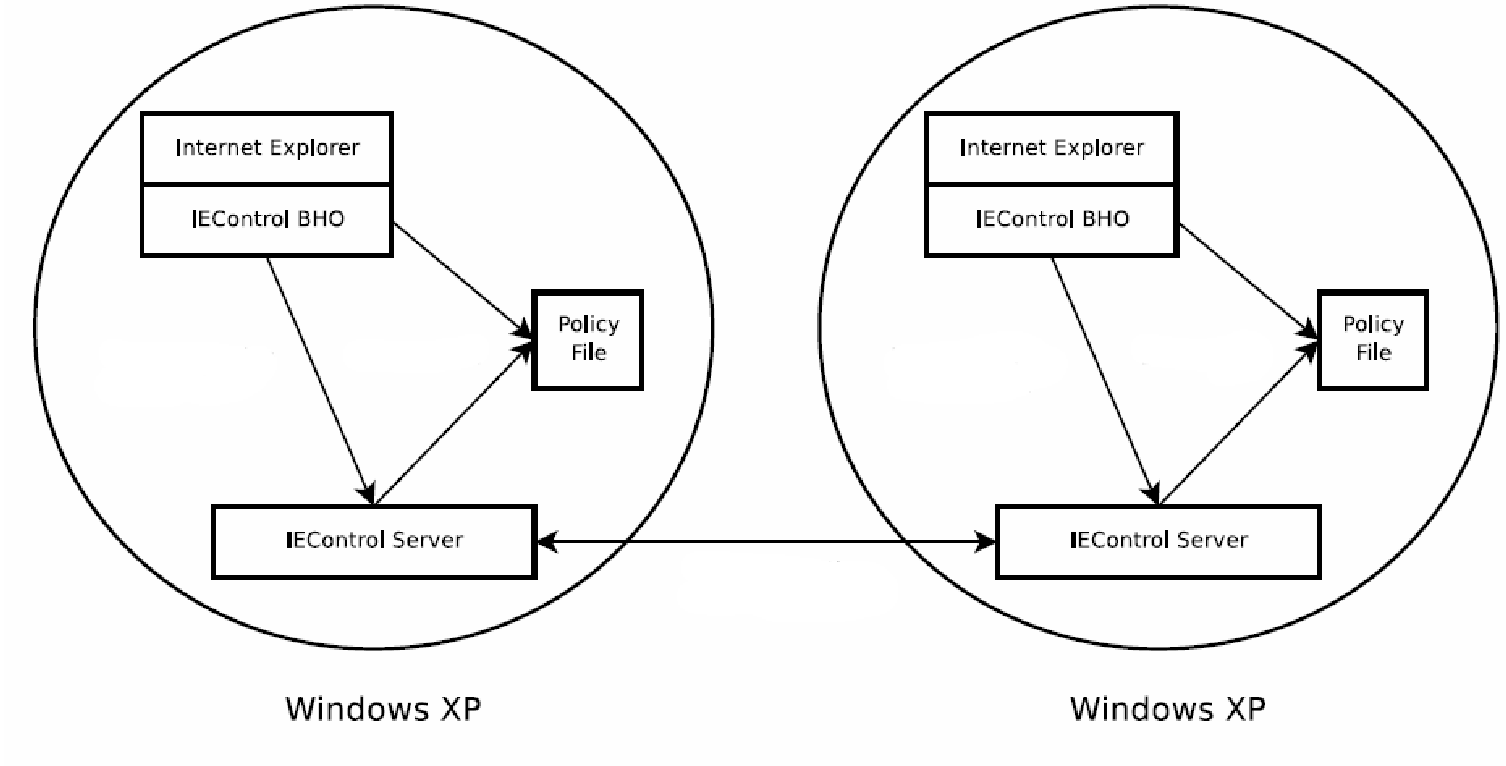
Open untrustworthy webpages in browser running on virtual machine

Open downloaded untrustworthy document types with applications on virtual machine

Give user as much support as possible:
- automatically open webpages/documents in trusted/untrusted environment based on configurable policy
- provide user-guidance

Our Experiment: Secure yet User-Friendly Browsing (II)

Implementation using browser helper object and controlled communication between trusted environment and OS in VMWare



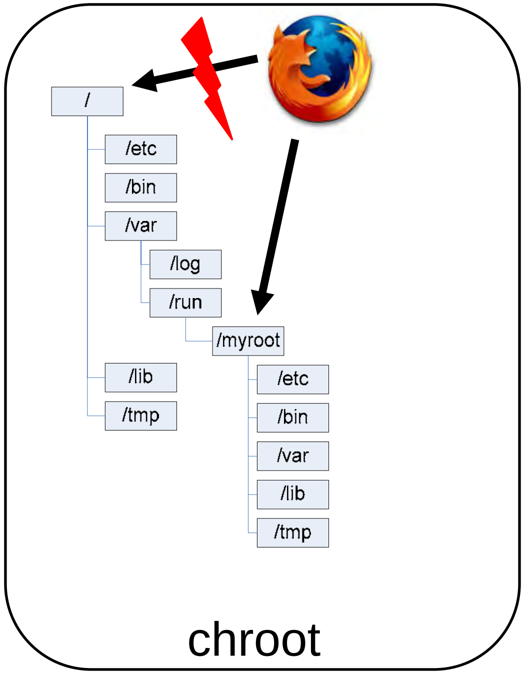
Our Experiment: Secure yet User-Friendly Browsing (III)

- Technical Experiences:
  - URL-based dispatch works rather nicely:
    - BHO examines URL
    - If URLis to be displayed in other browser,
      - request is stopped
      - user is informed via information window
      - request is forwarded to other browser
  - Filetype-based dispatch harder: reliable determination of filetype requires download
- Points to ponder:
  - user experience still clumsy
  - OS in virtual machine requires
    - license
    - maintenance (patching!)
  - also virtual machines may be vulnerable (cf. Ormandy, „An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments”)

Application Virtualization

- Layer between the operating system and applications
- Virtualizing the system environment of programs, providing components such as registry entries, files, environment variables, and global objects
- First steps towards application virtualization: Unix chroot and BSD jail

BSD jail =



chroot

+

- hiding processes not within jail
- restricting access to network ports
- restricting activities such as module loading, mounting files systems, etc.

Application Virtualization (II)

Several products offering „application virtualization“ available for Windows:

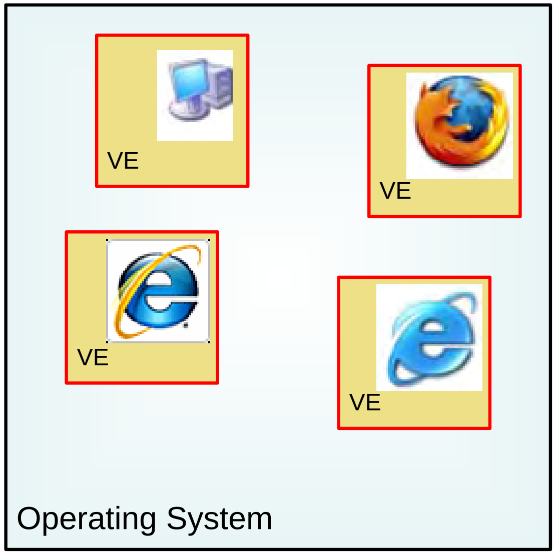
- Application centric
  - Central administration of applications in client-server-environments
  - Support for different program versions

altiris MS Softgrid CiTRIX

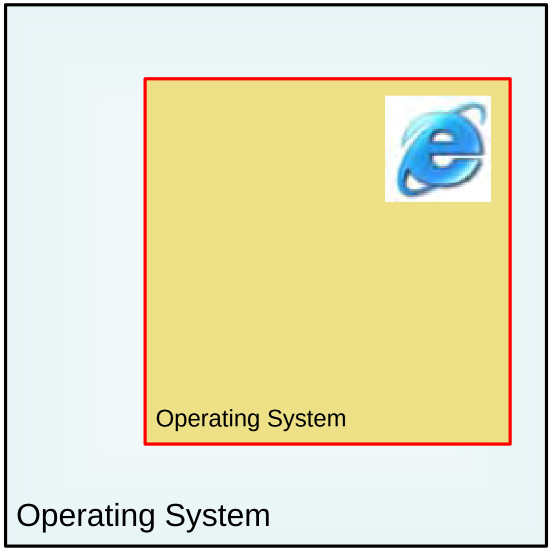
- Security driven
  - Security sandboxes
  - Isolating malware infections

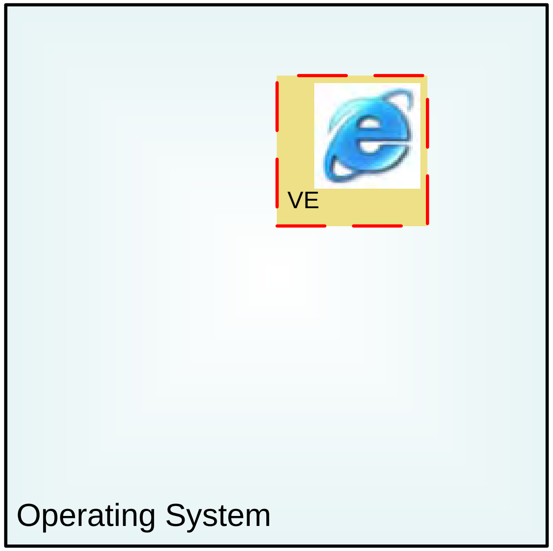
Sandboxie GesWall BufferZone

GreenBorder



OS Virtualization vs. Application Virtualization





- Application Virtualization draws separating border tighter around application; less overhead, easier integration, better usability
- But: what does the exact border look like?
  Does it keep everything inside that it should?

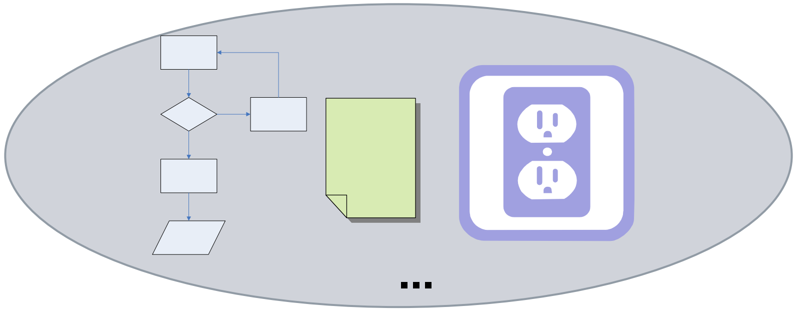
Test-Cases for Application Virtualization Solutions

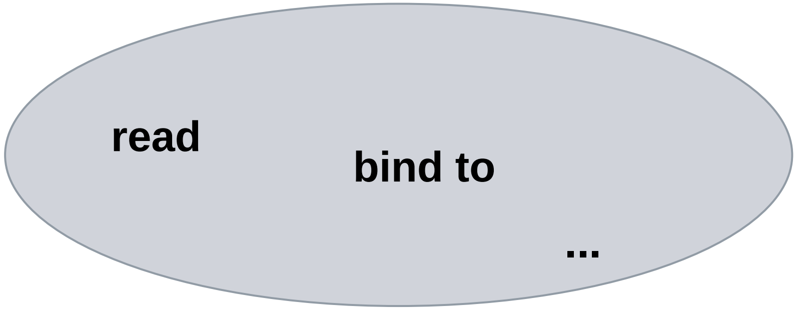
- Does the sandbox provide total isolation from infection by hostile web sites, 0-day threats, spyware, trojans, keyloggers, blended malware attacks and other contemporary malware threats?
- Is personal data on the "real PC" inaccessible to sandboxed programs?
- Does the product prevent sandboxed programs from reading and writing to raw memory?
- Does the product prevent sandboxed programs from accessing key system data such as system configuration and network information?
- Does the product prevent sandboxed programs from deliberately crashing the system
- Can a hostile program escape the sandbox by terminating the application virtualization solution?

[Source: http://www.techsupportalert.com/security_virtualization.htm]

Strict Access Controls: SELinux, for example (I)

- Concept of subjects/objects and associated access vectors





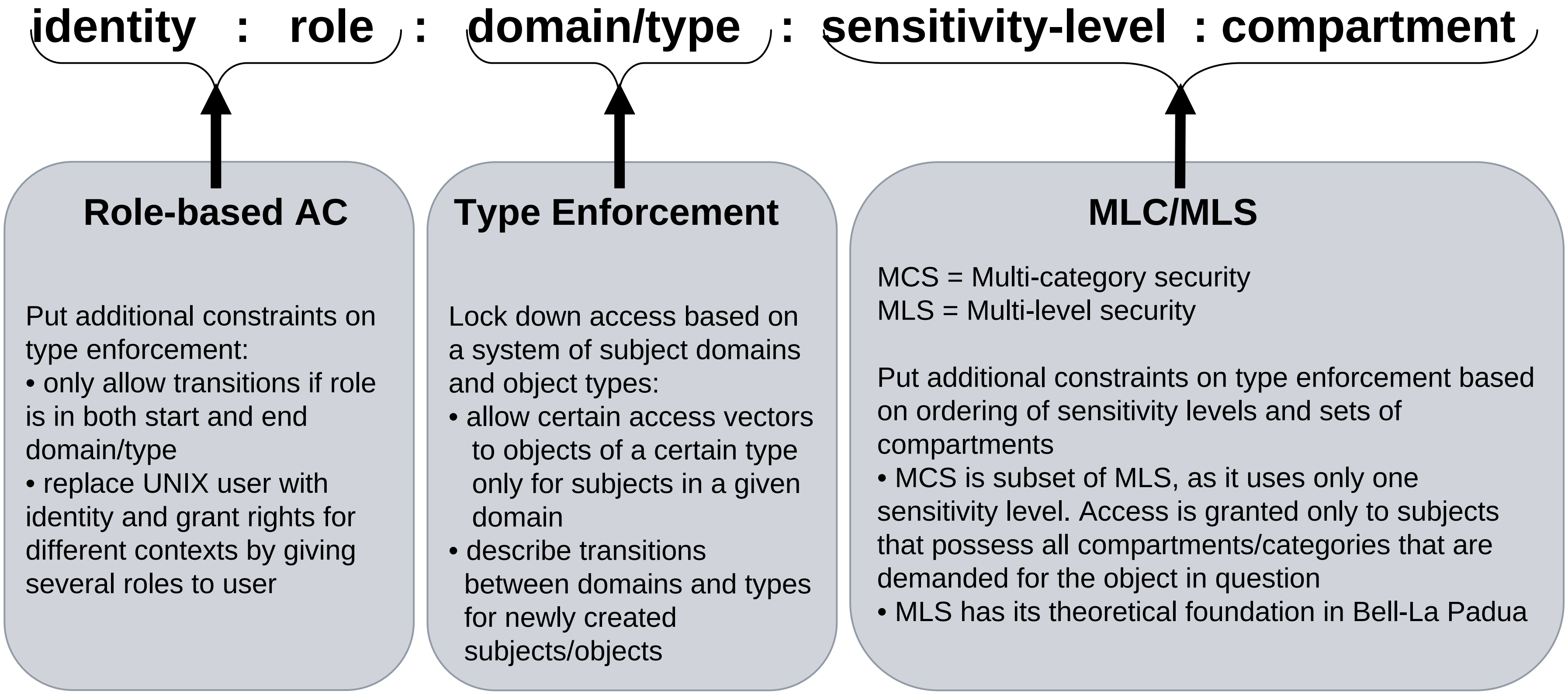
- Rules that allow/deny access based on some system. SELinux associates each subject/object with a security context:

identity : role : domain/type : sensitivity-level : compartment

- The security context
  - is used to control whether a subject is allowed to access an object with a certain access vector
  - is not static but goes through transitions – and getting the transitions right is actually at least as tricky as defining access restrictions for each context

Strict Access Controls: SELinux, for example (II)

SELinux offers several access control systems:



identity : role : domain/type : sensitivity-level : compartment

Role-based AC

Put additional constraints on type enforcement:

- only allow transitions if role is in both start and end domain/type
- replace UNIX user with identity and grant rights for different contexts by giving several roles to user

Type Enforcement

Lock down access based on a system of subject domains and object types:

- allow certain access vectors to objects of a certain type only for subjects in a given domain
- describe transitions between domains and types for newly created subjects/objects

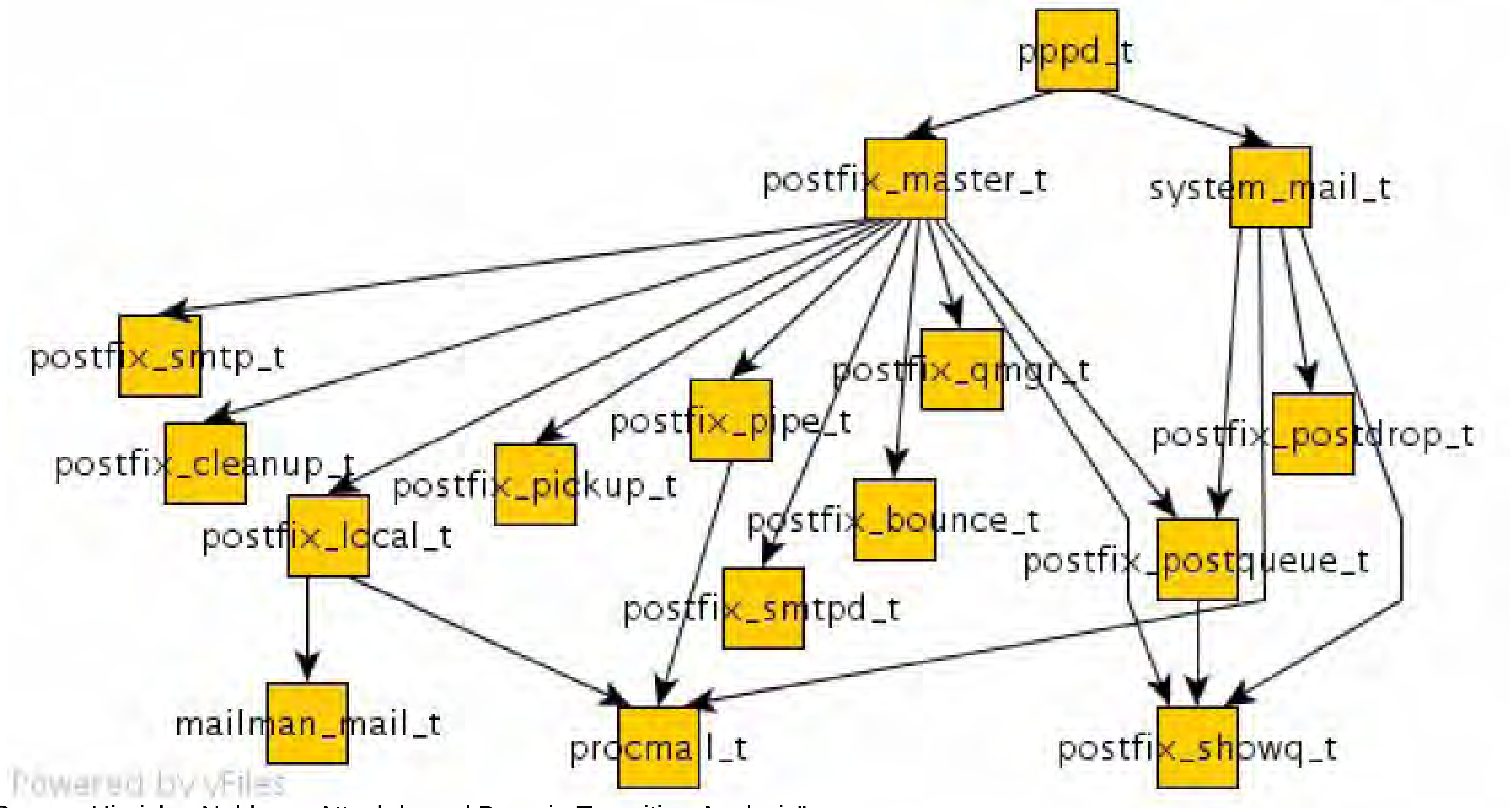
MLC/MLS

MCS = Multi-category security
MLS = Multi-level security

Put additional constraints on type enforcement based on ordering of sensitivity levels and sets of compartments

- MCS is subset of MLS, as it uses only one sensitivity level. Access is granted only to subjects that possess all compartments/categories that are demanded for the object in question
- MLS has its theoretical foundation in Bell-La Padua

Strict Access Controls: SELinux, for example (III)
Type Enforcement applied to Postfix



Source: Hinrichs, Naldurg: „Attack-based Domain Transition Analysis"

Strict Access Controls: SELinux, for example (IV)

Type Enforcement offers best possibility for application separation:

- With an appropriate set of rules, the type enforcement mechanism can separate subjects from one another on the same system
- In the default case,
  - a process is given a particular domain
  - all new processes created from that process are labeled with one of a set of domains created specifically for that application
- Thus, if a subject is compromised, the type enforcement rules constrain the actions an attacker can take with help of the compromised subject.

So far, mostly used for server rather than client applications, e.g., hardening IBM Websphere with SELinux (pilot project for British government)

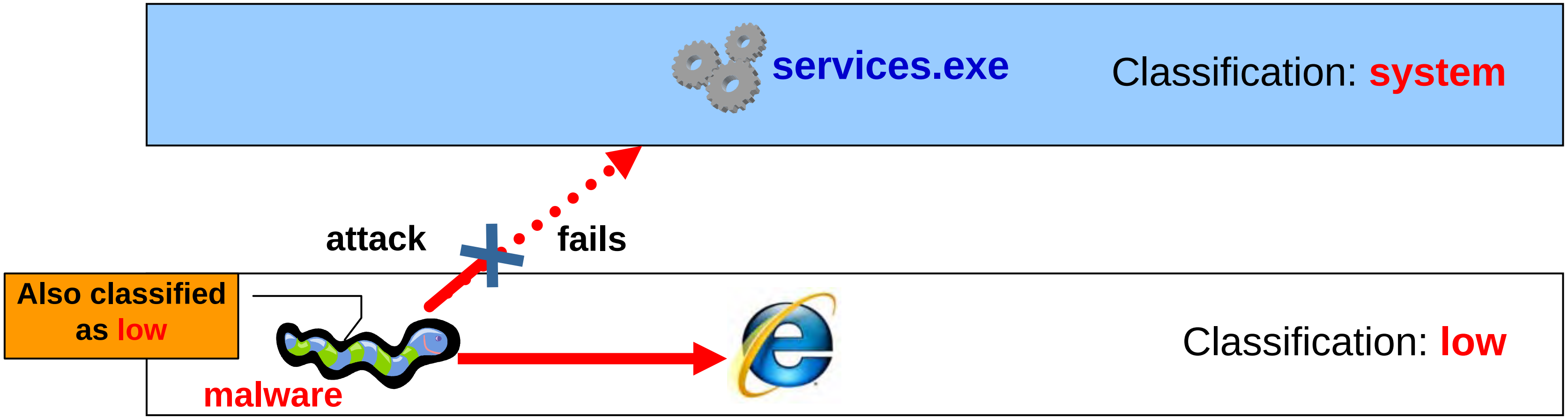
Windows Vista: Improved Separation Mechanisms built in?

- Windows Resource Protection (WRP)
    - Prevent system registry keys and system files from being replaced
- User Account Control (UAC)
    - Basic idea:
        - Use administrator account only if absolutely necessary
        - In all other cases use standard user profiles

        ⇒Avoid silent installations of malware
    - How realized:
        - Administrative user tokens are split into
            - a full administrator access token and
            - a standard user access token
        - Desktop and explorer are launched with standard user access token
        - Applications inherit their access control data
            - hence, they all run as a standard user as well
        - Users are prompted if administrative rights are required
- Mandatory Integrity Control (MIC)
- File System and Registry Virtualization

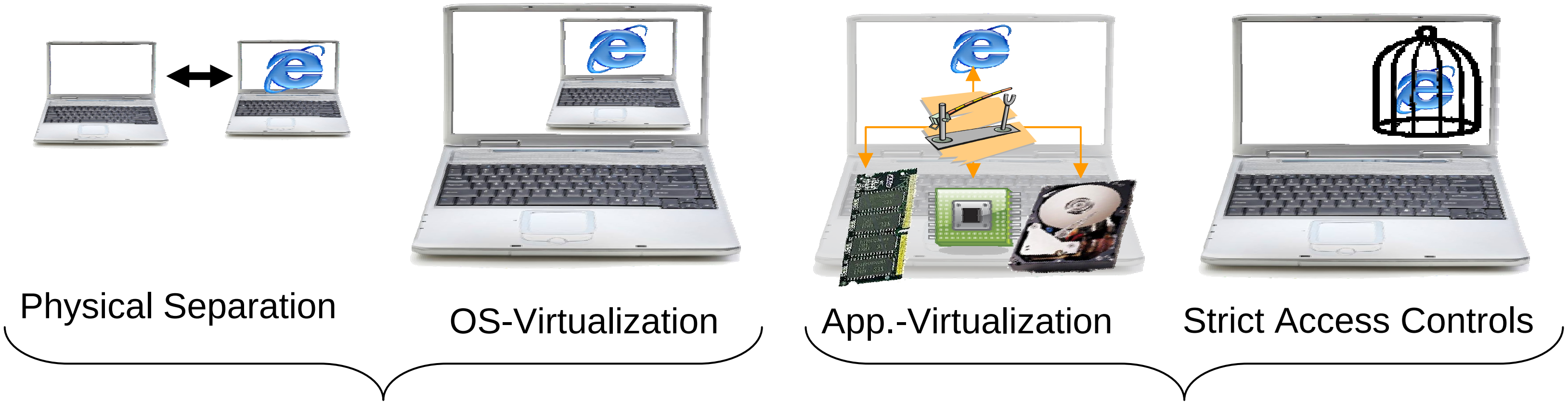
Mandatory Integrity Control

- Based on Biba model
- Four integrity levels
  - low (e.g., MS Internet Explorer)
  - medium (default)
  - high (elevated, administrative privileges)
  - system (only for system objects/processes)
- Securable objects:
  files, folders, pipes, processes, threads, registry keys, services, …
- Hinders low integrity code from modifying processes of higher integrity levels



Conclusion

- Risk of compromise of untrusted/highly vulnerable applications can be mitigated by separating applications from productive environment
- Several possibilities exist:



Physical Separation

OS-Virtualization

App.-Virtualization

Strict Access Controls

- For client applications, user usability not sufficient for most use cases
- Possible solution for high-security environment

- Allow better integration
- Will increasingly become a part of modern operating systems as standard features