BINDER: An Extrusion-based Break-In Detector for Personal Computers

Weidong Cui
wdc@EECS.Berkeley.EDU

OASIS Retreat
January 11, 2005

Joint Work with Prof. Randy Katz (UCB) and Wai-tian Tan (HPL)

Motivation

- Computer worms, spyware and adware have affected both personal and business computing significantly.
- Bot networks are big threats to the Internet
  - Compromised hosts (zombies) can be used for DDoS attacks, spam relay, and worm propagation.
- Misuse-based intrusion detection requires
  - Some central entities must rapidly generate signatures of new threats after they are detected.
  - Distributed computer systems must download and apply these signatures to their local databases in time.
- An attractive, complementary solution
  - Detect break-ins after they occur, but without priori exploit signatures.

Extrusions

- Many threats send malicious outgoing traffic
  - Worms: self-propagation
  - Spyware/Adware: upload/download information
  - Zombies: launch attacks/relay spam
- These network activities are usually unknown to users on the compromised personal computers.
- Extrusions: stealthy malicious outgoing network connections.

BINDER: Break-IN DEtectoR

- Key features of personal computers:
  - Extrusions are not triggered by users.
  - Most normal network traffic is triggered by users.
- Thus we can detect break-ins on personal computers by capturing extrusions.
  - Do not need priori exploit signatures!
- BINDER: An Extrusion-based Break-In Detector for Personal Computers

Design Objectives

- Minimal false positives
  - This is the critical base for any intrusion detection system to be useful in practice.
- Generality
  - Work for a large class of threats
- Security with open design
  - Cannot be bypassed by disclosing the scheme
- Small overhead
  - Must not use intrusive probing and affect the performance of the monitored systems

Extrusion Detection (I)

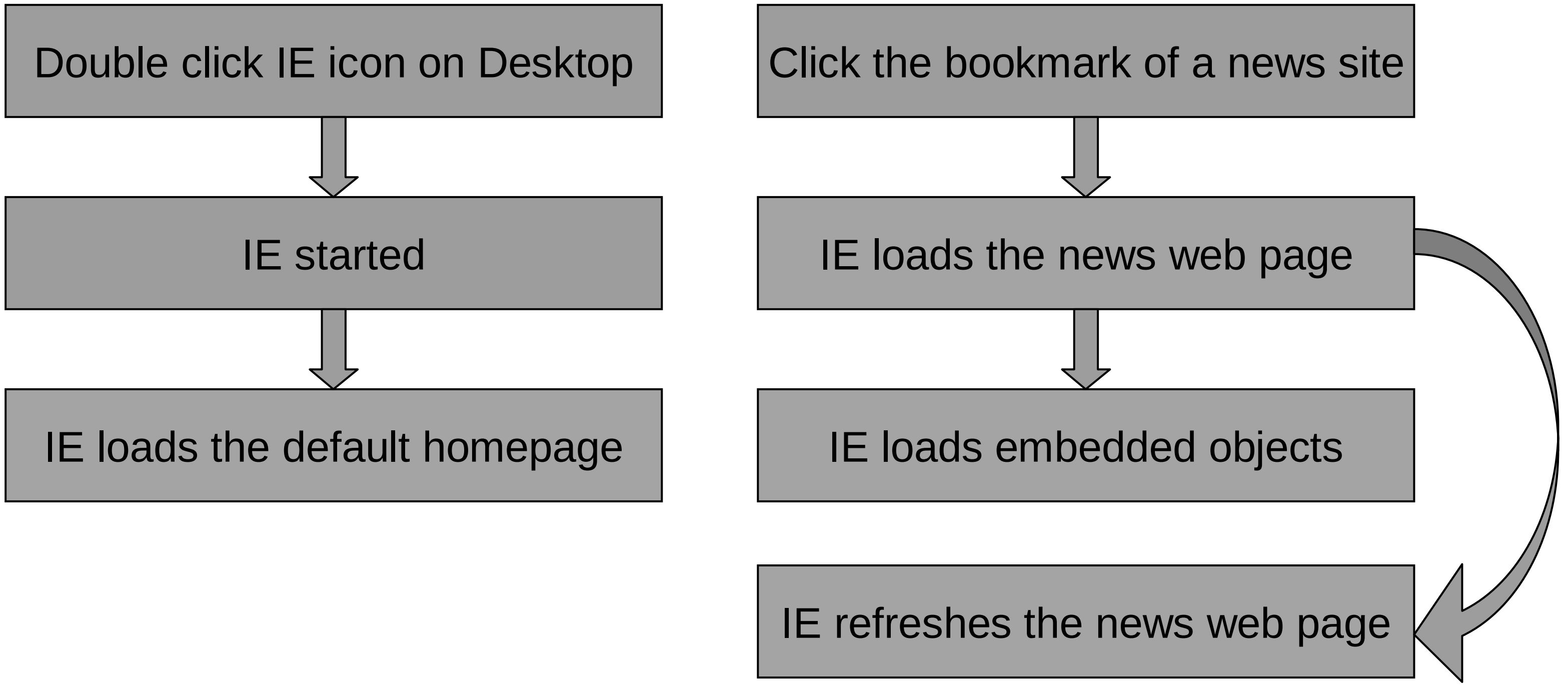
- Observation
  - extrusions are not triggered by users.
- How to determine if a network connection is triggered by a user?
  - Simple way: a network connection is generated shortly after a user input.
  - A smart malcode can bypass it by monitoring user input.
- Our approach
  - Use process information to limit the correlation between user input and network traffic.
  - Only processes that receive user input are allowed to make connections.

Extrusion Detection (II)

- Design choices
  - Find conditions to detect extrusions directly,
  - Or find conditions to cover normal connections
- We chose the latter because it matches our design objectives
  - Minimize false positives: control it directly
  - Generality: any abnormal connection is an extrusion
- In what ways can a normal connection be triggered?

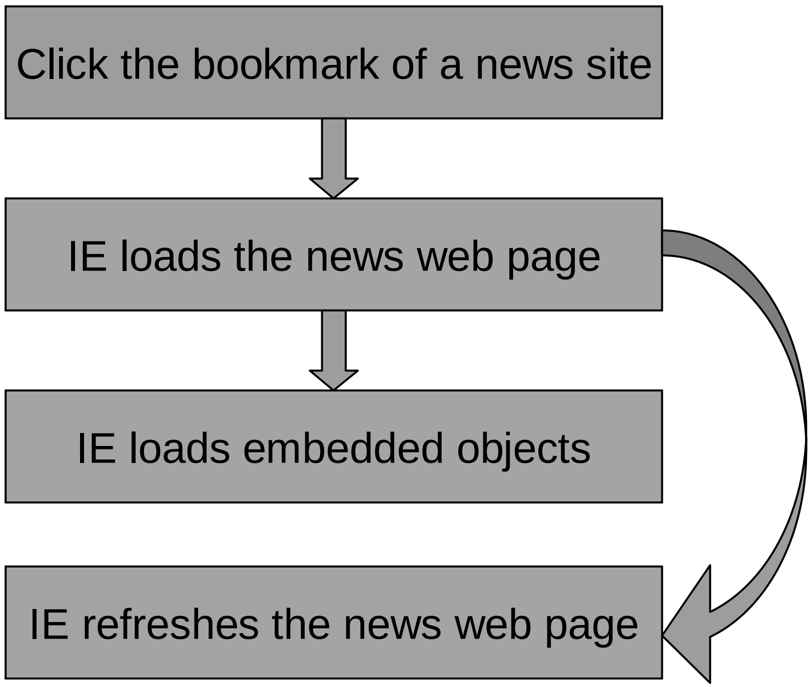
A Motivating Scenario

- A user opens an IE window, goes to a news web site, then leaves the window idle for answering a phone call.
- What may trigger normal connections?



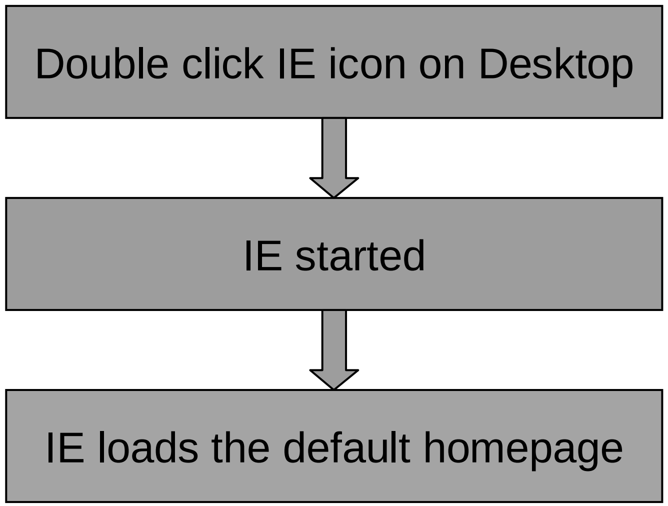
Normal Connection Rules (I)

- Intra-Process Rule
  - User input, data arrivals and previous connections of the same process can trigger new connections



Normal Connection Rules (II)

- Inter-Process Rule
  - User input and data arrivals of a different process can trigger new connections
  - We need to monitor all inter-process communications to apply this rule correctly. But it has high overhead.
  - We approximate this rule using
    - Parent-Process Rule
    - Web-Browser Rule



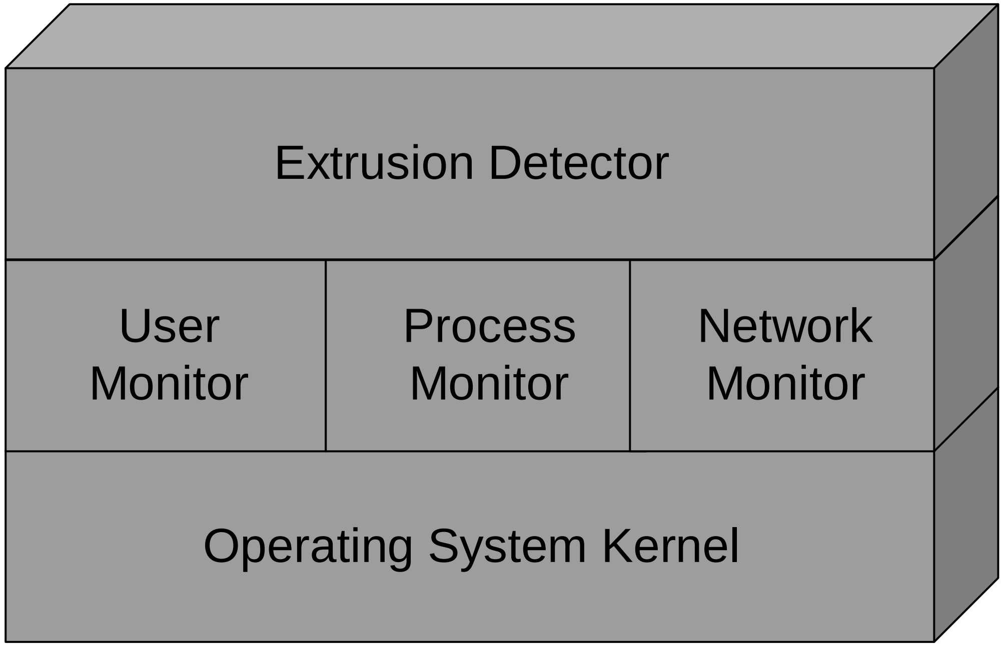
Detection Algorithm

- For a connection request, there are three parameters
  - D_{new}: The delay since the last user input or data arrival received by the parent process before a process is created.
  - D_{old}: The delay since the last user input or data arrival received by the same process
  - D_{prev}: The delay since the last connection request to the same host or IP address made by the same process
- For a normal connection, it must have at least one of the three delays fall into a normal range (less than a pre-defined upper bound).

Detecting Break-Ins

- Two phases of a break-in
  - Before the compromised system is restarted
  - After the compromised system is restarted
- In the second phase
  - Malicious processes are started by the OS when the system is boot up
  - Run as background processes that do not receive any user input
  - All connections made by malicious processes will be classified as extrusions.
- In the first phase
  - Some connections made by malicious processes may not be detected as extrusions if they meet some normal connection rules.
  - BINDER need to capture just one extrusion to detect a break-in.
  - In reality, many threats can be detected in this phase.
  - Future work: find more restrictions on normal connection rules to make BINDER more effective in this phase.

BINDER Architecture

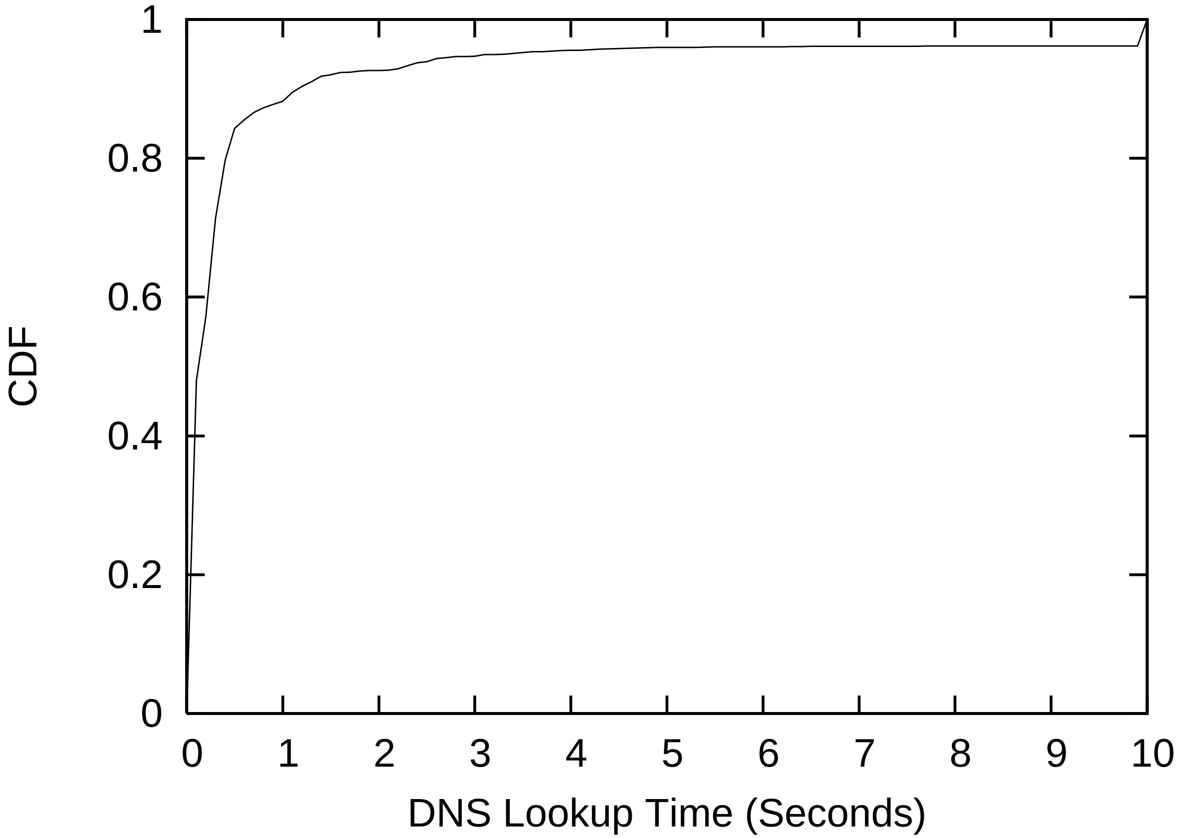


- The User Monitor, Process Monitor, and Network Monitor are OS-dependent for collecting information passively in real time.
- The Extrusion Detector detects extrusions based on information of user input, processes and network traffic.

Events

- User Monitor
  - User Input: Time, Process ID
- Process Monitor
  - Process Start: Time, Process ID, Image File Name, Parent Process ID
  - Process Finish: Time, Process ID
- Network Monitor
  - Domain Name Lookup: Time, Host Name, IP addresses
  - Connection Request: Time, Process ID, Local Port, Remote IP, Remote Port
  - Data Arrival: Time, Process ID, Local Port, Remote IP, Remote Port
- Extrusion Detector
  - Process-based data record: Process ID, Image File Name, Parent Process ID, Last User Input Time, Last Data Arrival Time, All Previous Network Connections

Why consider DNS lookup?



- DNS lookup may take significant time between a user input and the corresponding connection request.

Implementation on Windows

- User Monitor
  - Based on Windows Hooks APIs
- Process Monitor
  - Based on the built-in Security Auditing on Win2K and WinXP
- Network Monitor
  - Based on TDIMon (Transport Drive Interface) and WinDump
- Extrusion Detector
  - OS-independent detection algorithm
  - Whitelisting

Whitelisting on Windows

- System daemons
  - Allowed to make connections at any time
  - System, Spoolsv.exe, svchost.exe, services.exe, lsass.exe
- Software updates
  - Allowed to connect to the update web site at any time
  - Symantec, Sygate, ZoneAlarm, Real Player, MS Office, Mozilla
- Network applications automatically started by Windows
  - Allowed to make connections at any time
  - Messengers of MSN, Yahoo!, ICQ, AOL
- 15 rules in total

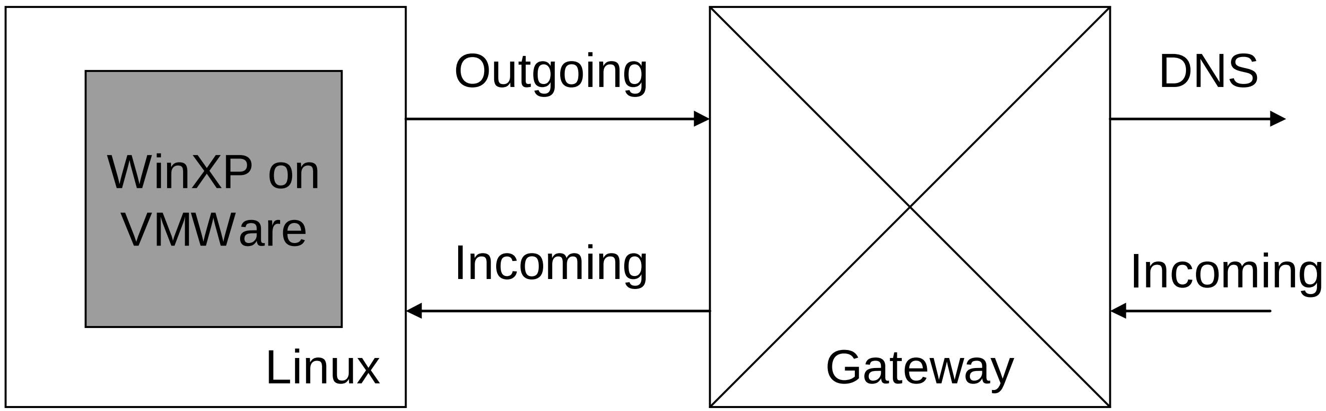
Evaluation Methodology (I)

- Real-world trace-based experiments
  - Installed BINDER on 6 computers used by different people for their daily work over 5 weeks
  - Diversity on hardware, OS, user behavior

User	HW	OS	Days	User Input	Process	Net App	TCPConn
A	Desktop	WinXP	27	35270	5048	33	33480
B	Desktop	WinXP	26	80497	12502	35	15450
C	Desktop	WinXP	23	24781	7487	55	36077
D	Laptop	Win2K	23	99928	8345	28	9784
E	Laptop	WinXP	13	8630	2448	21	10210
F	Laptop	WinXP	12	20490	5402	20	7592

Evaluation Methodology (II)

- Experiments with real-world threats in a controlled testbed using VMWare
- Obtain real malcode.
  - We get virus emails from three channels.
    - Set up a mail server and publish an email address in Usenet
    - From colleagues
    - From local system administrators
- Reinstall operating system
  - By using VMWare, we just need to copy several files
- Contain malcode
  - Open a door for DNS, otherwise no connections at all
- Can only check if the first connection is extrusion



Parameter Selection

- The upper bound of the three delays
  - D_{new} ~ 30 seconds (The delay since the last user input or data arrival event received by the parent process before a process is created)
  - D_{old} ~ 30 seconds (The delay since the last user input or data arrival event received by the same process)
  - D_{prev} ~ 800 seconds (The delay since the last connection request to the same host or IP address made by the same process)
- The 95 percentile is good choice for the upper bound regarding false alarms.
- It can be obtained by training BINDER over a period of virus-free time.

False Alarms

- Incomplete information of inter-process communications
- Incomplete whitelisting
- Incomplete trace collection

User	Inter-Process	Whitelist	Collection	Total
A	2	1	0	3
B	4	1	0	5
C	1	0	0	1
D	0	1	1	2
E	1	1	1	3
F	0	1	1	2

Detecting Break-Ins (I)

- Real-world experiments
  - One computer is infected by Adware Gator and CNSMIN
  - Another computer is infected by Adware Gator and Spydeleter
- Controlled experiments
  - Four email worms: Bagle, NetSky, MyDoom, Swen
- All the break-ins can be detected by BINDER after the compromised host is restarted (in the second phase).
- BINDER detected Spydeleter, Bagle, NetSky, Swen in the first phase.

Detecting Break-Ins (II)

- Adware Spydeleter
  - BINDER Detected it right after it compromised the computer
  - IE => svchost.exe => mshta.exe => ntvdm.exe => ftp.exe
- Bagle
  - The first connection is detected as an extrusion
  - Email client => joker.com => bawindo.exe
- Swen
  - The first connection is detected as an extrusion
  - Similar to Bagle
- NetSky
  - The first connection is generated 90 seconds (>30 seconds) after the attachment is executed, so it's detected as an extrusion
- MyDoom
  - The first connection is not detected as an extrusion

Potential Countermeasures (I)

- Direct attacks
  - General concern for host-based schemes
  - BINDER runs in the kernel space
  - Active research on verifying integrity of files
- Fake user input
  - Use APIs provided by the OS to generate "soft" user input
  - BINDER can monitor these APIs
- Trick user to input
  - Pop-up a window to trick user to input and then make connections
  - Pop-up windows can be detected because they are created before any user input

Potential Countermeasures (II)

- Hide under processes
  - A break-in installs itself as a DLL library file and loads as a thread in a process
  - BINDER relies on the OS to guarantee the process boundary
- Covert channels
  - Break-ins may use IE to download a specific link to disclose private information
  - Active research [Web Tap] on this problem
- Hide under user input
  - Like MyDoom, when a user executes a program, BINDER will treat its connections as normal.

Future Work

- BINDER cannot handle the "hide-under-user-input" case.
- We need to learn more about normal patterns of network traffic, process and user input.
- Study the tradeoff between host-based monitoring and network-based monitoring
  - Host-based: more information, less reliable
  - Network-based: less information, more reliable
  - Virtual Machine Monitor-based?

Conclusions

- Motivation
  - It's important to detect break-ins after they occur.
- Observation
  - Break-ins make outgoing connections unknown to users on personal computers
- Solution
  - BINDER: detect break-ins on personal computers by capturing extrusions
- A prototype of BINDER is implemented on Windows
- Performance
  - Very few false alarms
  - Guarantee to detect break-ins after the victim computers are restarted
  - Can detect many real-world malware right after they break in
- Limitation
  - "hide-under-user-input" cannot be detected