Challenges in OS Security

Daniela Oliveira
University of Florida

GREPSEC II Workshop, May 16-17, 2015
Challenges in OS Security

1. Safe co-existence with extensions
2. Collaboration with hardware
3. Overcoming monoculture
Challenges in OS Security

1. Safe co-existence with extensions
2. Collaboration with hardware
3. Overcoming monoculture

Immune System
Cyber Security x The Mammalian Immune System

- bacteria
- viruses
- fungi
- parasites
- toxins
Mammalian Immune System

Most successful defense system ever deployed
Though it fails sometimes (cancer, auto-immune diseases, allergies)

Perfected by Nature over millions of years of evolution!
Mammalian Immune System

Employs high level of cooperation and communication among players

Maintains a symbiotic relationship with our microbiota
Properties Lacking in Computer Security Approaches

Maintains a symbiotic relationship with our microbiota

Employs high level of cooperation and communication among players
Why don’t we leverage the immune system mechanisms in security approaches?
Safe Co-existence with Extensions

Kernel extensions represent at least 70% of kernel

Most benign and needed:
Kernel Extensions: Trusting the Untrustworthy

Small fraction is malicious
Untrustworthy Dependence - A Paradox?

OS must co-live with untrustworthy but *needed* extensions!
Untrustworthy Dependence

Immune system faces the same challenge:

Body made of more bacteria than human cells
Most benign and helpful:

- Digestion, obesity control, eczema, auto-immune diseases and allergy prevention

Small fraction cause pathologies
Untrustworthy Dependence

Immune evolved to maintain homeostatic relationship with microbiota:

Controlling microbial interactions with tissues

- Lessen potential for pathological outcomes
Immune System Approach

1. Confinement of bacteria to certain sites
2. Minimization of direct contact between bacteria and cell surfaces
3. Killing violating bacteria
Challenge 3: Overcoming the Problems of Computer System Monoculture?
Predictability poses security problems...

Vulnerabilities exploitable on all systems of same type

- Code Red 2001: 359,000 hosts infected
- $2 billion in losses
Predictability Makes Attacker’s Life Easier

What If Operating Systems Were Trustworthy Unpredictable?

“NO ONE IS SO BRAVE THAT HE IS NOT DISTURBED BY SOMETHING UNEXPECTED.”

JULIUS CAESAR

© Lifehack Quotes
Unpredictability in Warfare – Battle of Salamis (480 B.C.)
Unpredictability “Trends”

Address Space Layout Randomization (ASLR)
ISA Randomization
Compiler Specialization
Diverse implementation
   N-version programming, library OSes

Still residual certainty that benefits attackers!

Variation without unpredictability is not enough.
Trustworthy Unpredictability at OS Level

For “good” uses: OS is predictable -> efficiency and reliability

For “bad” uses: OS inefficient and unreliable

Selective Unpredictability
Spectrum Behavior OS

Chameleon

Diverse

Diverse environment
Host and herd protection

Unpredictable

Inconsistent deception
Diverse environment
Host and herd protection

Deceptive

Consistent deception
Active defense
Forensics
Typical Scenario

Bob, 78, living in a retirement community in Florida
not computer savvy, clicks in links from phishing email, installing malware

Malware engage in later DDoS attacks

Bob never notices: malware is active only after 1am.
Chameleon Scenario
Preliminary Work

Assumptions:
Malware is usually poorly written
Robust applications have end-to-end checks

Methodology
Use of ptrace to introduce unpredictability at system call level

Strategies

Strategy 1: Silence the system call
Strategy 2: Change buffer bytes
Strategy 3: Add more wait time
Strategy 4: Change file pointer
Unpredictability Coverage

Only for system calls not critical to process start-up
Keylogger with Unpredictability

Strategies:
  Change `write(fd, *buf, size)` buffer;
  Change `lseek(fd, offset, whence)` pointer;

<table>
<thead>
<tr>
<th>Input</th>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hi, test for Keylogger!</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td></td>
</tr>
<tr>
<td>username password</td>
<td></td>
</tr>
<tr>
<td>&lt;Ret&gt;</td>
<td>hi, testeylogger&lt;Rs&lt;Ret&gt;</td>
</tr>
<tr>
<td>&lt;Lshift&gt;hi, testeylogger</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>&lt;Ret&gt;</td>
</tr>
<tr>
<td>xImtpane passw</td>
<td>&lt;Ret&gt;</td>
</tr>
</tbody>
</table>
Keylogger with Unpredictability

Strategies:
- Change `write(fd, *buf, size)` buffer;
- Change `lseek(fd, offset, whence)` pointer;
Botnet with Unpredictability

Strategies:

- Silence `read(fd, *buf, size);`
- Silence or reduce `len` in `sendto(sockfd, *buf, len, ...)`
What About Benign Software?

Firefox, Thunderbird and Skype

Work normally most of the time
Occasional warnings
Functionalities temporarily unavailable
Concluding Remarks

Holy grail of system design: thwart attacker with less effort than generating attacks
Chameleon makes systems diverse by design and actively secure:
  Diverse + Unpredictable: every instance of system behaves differently

Deceptive: lures adversaries into revealing their strategies

Unpredictability is promising!
Collaborators

Ruimin Sun, PhD Student University of Florida

Don Porter
Stony Brook

Matt Bishop
UC Davis

Natalie Ebner
University of Florida
University of Florida is Rising!

Patrick Traynor
Mobile Security

Juan Gilbert
Electronic Voting

Mark Tehranipoor
Supply Chain Security

Kevin Butler
Cyber Physical Systems

Tom Shrimpton
Crypto

Domenic Forte
Hardware Trojan Prevention

Swarup Bhunia
Hardware Trojan Detection

Damon Woodard
Biometrics/fingerprinting
Thank you!

daniela@ece.ufl.edu