Experiences Using Minos as a Tool for Capturing and Analyzing Novel Worms for Unknown Vulnerabilities

Jedidiah R. Crandall†, S. Felix Wu†, and Frederic T. Chong‡

†University of California, Davis

‡University of California, Santa Barbara

Goals

- Describe Minos and its efficacy as a honeypot technology for automated response
- Analyze attacks captured by Minos in order to estimate the limits of worm polymorphism

Outline

- Vulnerability Landscape
- Minos
- Epsilon-Gamma-Pi Model
- Exploits Caught by Minos
- Future Work

Main Contributions (1)

- Minos as a honeypot technology
  - No false positives after 12 months of operation
  - Has caught all 9 of the actual control data exploits thrown at it without any prior knowledge about the exploit or vulnerability
  - Can catch control data exploits for unknown vulnerabilities in any part of the system: security products, CPL==0 exploits, passive exploits, etc.

Main Contributions (2)

- Epsilon-Gamma-Pi model
  - Epsilon (ε) = Exploit Vector
  - Gamma (γ) = Bogus Control Data
  - Pi (π) = Payload
- Analysis in the paper
  - NOP sleds are not needed in most Windows exploits
  - Quantification of how much polymorphism is possible in γ and π (ε left to future work)

Vulnerability Landscape

- Laws of Vulnerabilities (Gerhard Eschelbeck of Qualys at Blackhat 2004)
  - Half-life of critical vulnerabilities is 21 days
  - Half of the most prevalent are replaced by new vulnerabilities every year
  - Lifespan of some vulnerabilities and worms is unlimited
  - 80% of worms and automated exploits occur in the first two half-lives

Vulnerability Landscape (2)

- Vulnerabilities in security products
  - 2004: 60 critical flaws in security products, almost double the 31 in 2003, 2005 up to May: 23, up 50% over 2004 (Sarah Lacy at BusinessWeek, 17 June 2005)
    - Now outnumber critical Microsoft vulnerabilities
  - Witty worm: ISS products, 2 days from vulnerability disclosure to the worm outbreak
  - “Remote Windows Kernel Exploitation” by Barnaby Jack at eEye describes exploitation of a remote CPL==0 buffer overflow in Symantec Personal Firewall

Vulnerability Landscape (3)

- Remote vulnerabilities in CPL==0
  - eEye paper from the last slide
  - 14 June 2005: Remote heap buffer overflow in Microsoft Windows SMB implementation
  - Much processing of network data occurs in Windows kernel space: 2/3 of LSASS exploit vector, TDIs, RPC, Mailslots, Named Pipes, etc…, even IIS 6.0 HTTP processing
  - Windows (Feb 2005) and Linux (Nov 2004) both had remote SMBFS buffer overflows (but require victim to visit attacker's SMB share)

Vulnerability Landscape (4)

- Passively exploited vulnerabilities
  - SMBFS flaws in Windows and Linux from the last slide
  - Web browsers with buffer overflows, etc.
  - P2P networks

Vulnerability Landscape (5)

- 0day vulnerabilities
  - Of 13 vulnerabilities we studied, none were discovered by the software vendor
  - If 3rd party researchers can discover 0day vulnerabilities, so can attackers
  - May 2005: Zero-day exploits for unknown vulnerabilities in Mozilla Firefox

Automated honeypot technologies must be able to analyze exploits for unknown vulnerabilities in places heretofore not considered.

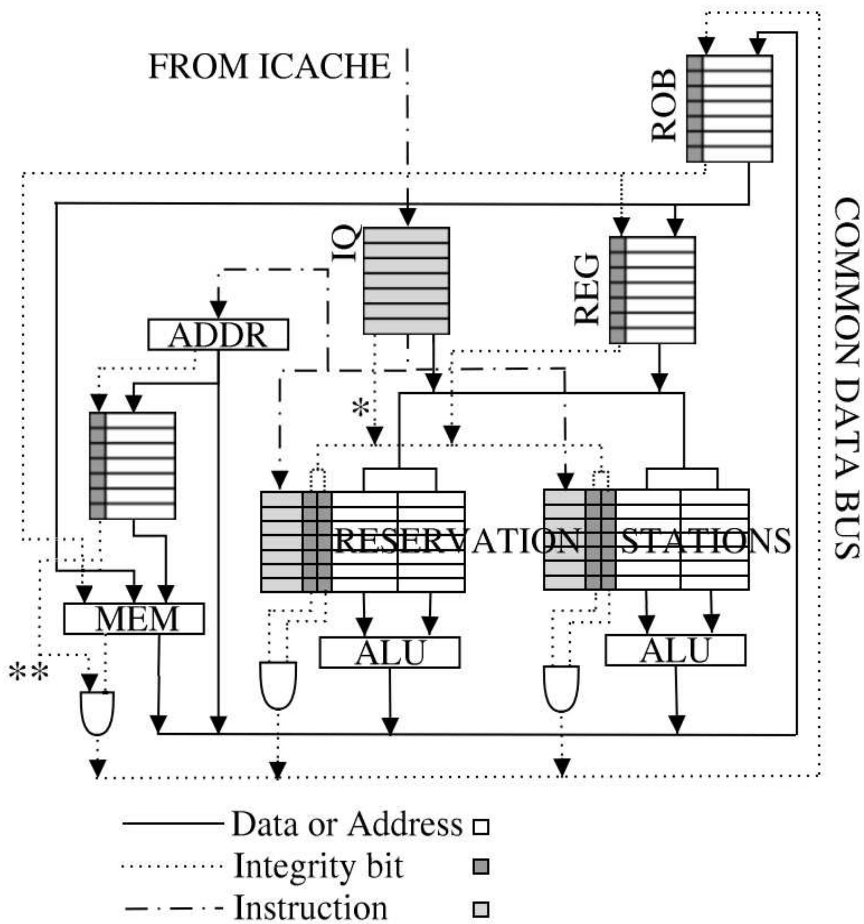
Minos

- The Minos architecture was introduced in [Crandall, Chong. MICRO 2004]
- Bochs emulations of Minos serve as excellent honeypots
  - Linux
  - Windows XP/Whistler (not as secure without kernel modifications, but good enough)
- Attacks in this paper were either on 1 on-campus honeypot in the summer of 2004 or 3 off-campus honeypots between Dec 2004 and Feb 2005
- Some (such as CRII, Slammer, Blaster, and Sasser) occur daily and, at times, hourly

What is control data?

- Any data which is loaded into the program counter on control flow transfer, or any data used to calculate such data
- Executable code is **not** control data
- Minos catches control data attacks (buffer overflows, format strings, double free()s, etc.)
  - Control data attacks constitute the majority of remote intrusions
  - Minos has some limitations described in MICRO2004
  - Minos was not designed to catch directory traversal, default passwords, high-level control flow hijacking like the Santy worm, or the attacks described in [Chen et. al., USENIX 2005].

How Minos Works



- Tag bit for every data word
- Biba's low-water-mark policy
- 8/16-bit loads/stores and immediates are low integrity
- Changes to Linux kernel detailed in MICRO2004, analysis is done with gdb
- No changes to Windows at all, network card port I/O is assumed low integrity, analysis done with Bochs debugger

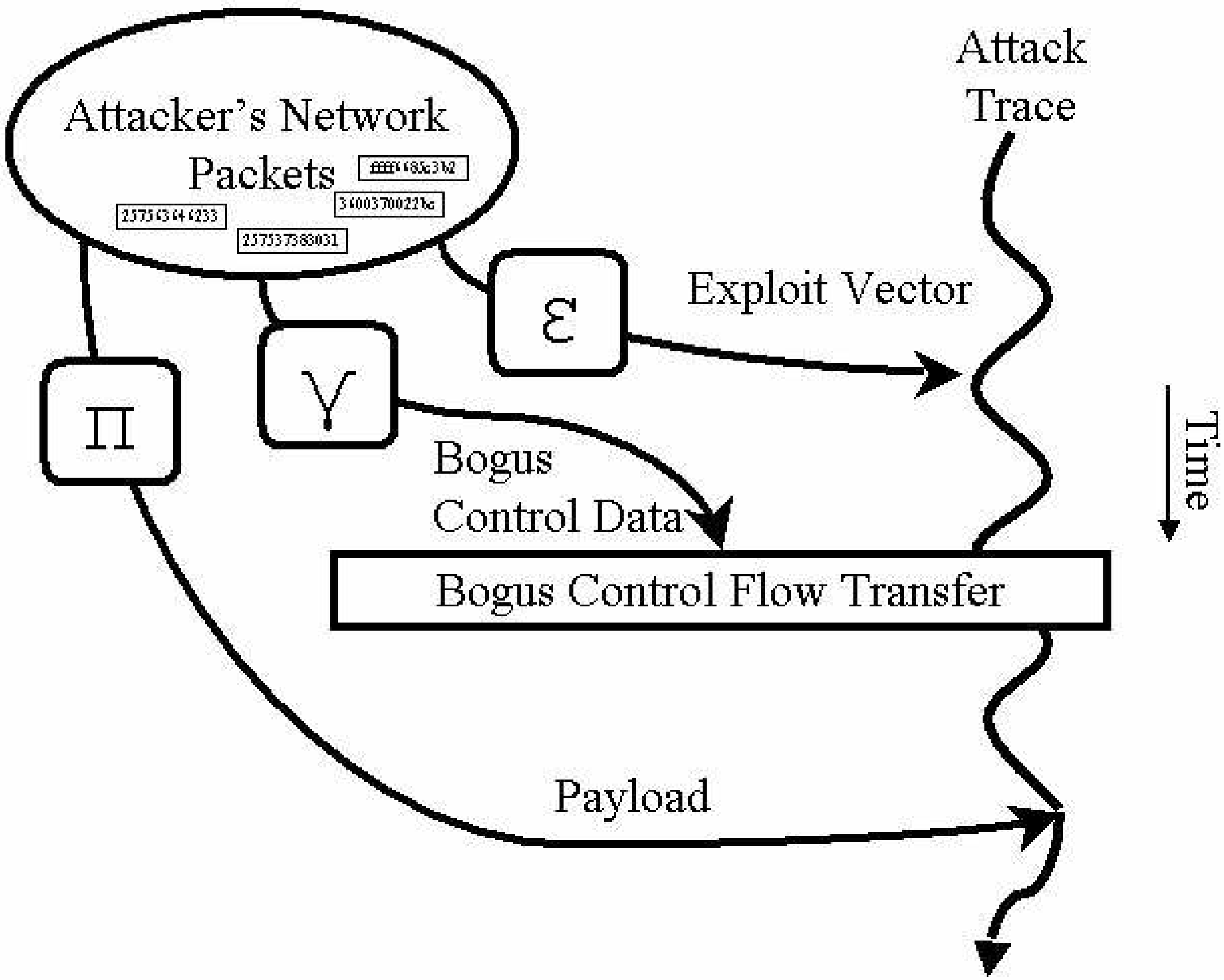
The Epsilon-Gamma-Pi Model (1)

- Main motivation for this model was to be able to discuss polymorphism more clearly and precisely
- Attacks are split into three distinct phases (ε , γ, and π) because for each phase the polymorphic techniques are different

The Epsilon-Gamma-Pi Model (2)

- Epsilon, Gamma, and Pi are mappings to capture the differences between data as it passes over the network and data as it is processed in the physical machine
  - i.e. for Code Red II the row space of γ is “25 75 63 62 64 33 25 75 37 38 30 31” and the range is “d3 cb 01 78”, both representations of 0x7801cbd3
  - WORM vs. WORM [Castaneda et al. WORM2004] assumed the row spaces and ranges of ε , γ, and π were disjoint sets of bytes and thus parts of the “black worm” may be left behind in the “white worm”

The Epsilon-Gamma-Pi Model (3)



Gratuitous Von Clausewitz Quote

"Where two ideas form a true logical antithesis, each complementary to the other, then fundamentally each is implied in the other."

--Carl von Clausewitz, *On War*, 1832

Actual Attacks Caught by Minos

Name	Vuln	Type	First Hop	Port
SQL Hello	SQL 2000	Buff. Over.	Register Spring*	1433 TCP
Slammer	SQL 2000	Buff. Over.	Register Spring*	1434 UDP
Code Red II	IIS 4.0-5.0	Buff. Over.	Register Spring*	80 TCP
DCOM (Blaster)	Windows	Buff. Over.	Register Spring	135 TCP
LSASS (Sasser)	Windows	Buff. Over.	Register Spring*	445 TCP
ASN.1	Windows	Heap B.O.	Register Spring	445 TCP
wu-ftpd	Linux	Dbl. Free()	unlink() macro*	21 TCP
ssh	Linux	Buff. Over.	NOP sled	22 TCP

*confirmed that NOP sled is not necessary

Since DIMVA camera-ready deadline: Unidentified on 135 TCP (RPCSS?)

Observations

- NOP sleds are largely unnecessary for Windows exploits due to *register springs*
- Register springs, among other techniques, allow for a great deal of polymorphism in γ
- Simple polymorphic decryptors for π would probably range from 19 to 32 bytes long
  - Short enough to evade many string matching approaches (for example in Earlybird [Singh et al. OSDI 2004], β==40)
  - Abstract Payload Execution [Toth and Kruegel. RAID 2002] saw MELs in HTTP traffic of 14

Polymorphism in π

```
mov eax,030a371ech     ; b8ec71a339
add eax,0fd1d117fh     ; 057f111dfd
add eax,0b00c383fh     ; 053f380cb0
push eax               ; 50
add eax,03df74b4bh     ; 054b4bf73d
add eax,0e43bf9ceh     ; 05cef93be4
push eax               ; 50
...
add eax,02de7c29dh     ; 059dc2e702
add eax,014b05fd8h     ; 05d85fb014
push eax               ; 50
add eax,06e7828dah     ; 05da28786e
call esp               ; ffd4
```

Polymorphism in γ

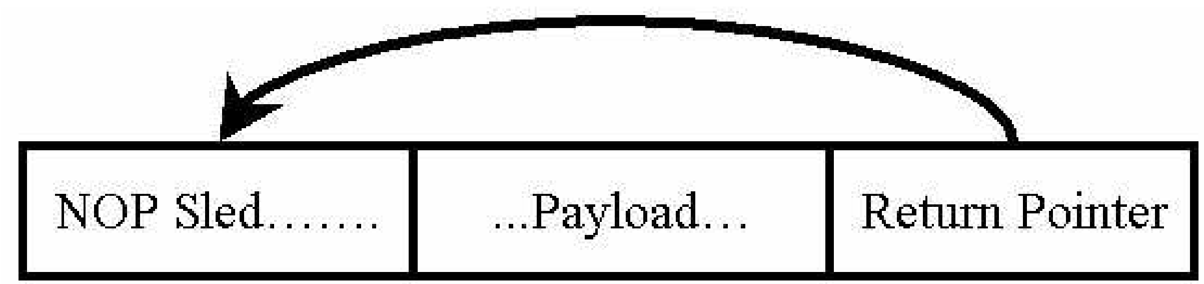
- Buttercup [Pasupulati et al. NOMS 2004]
- Hundreds or thousands of register springs are usually possible (11,009 for EBX in DCOM, 353 for ESP in Slammer)
  - Variance across service packs is not really a problem
- Format string attacks: “%100d%100d%100d” can be rewritten as “%80p%90f%130x”

Future Work

- Polymorphism in ε
  - DACODA
  - Signature generation
- Minos as an active honeypot seeking passive exploits (P2P, web browser, …)
- Performance (QEMU instead of Bochs?)

Conclusions (1)

- Emphasis on the NOP sled in polymorphic worm studies may not be appropriate for Windows exploits
- This figure does not capture the complexity of real exploits:



Conclusions (2)

- Minos is a very capable honeypot technology looking ahead to the new vulnerability landscape
- Much polymorphism is available in γ and π, should look at ε instead