## Analysis of Adversarial Code: Problem, Challenges, Results

### Arun Lakhotia

Center for Advanced Computer Studies University of Louisiana at Lafayette www.cacs.louisiana.edu/labs/SRL

> Black Hat Federal 2006 Sheraton Crystal City, Washington DC January 23-26, 2006

## Adversarial Code Analysis

- ACA at UL Lafayette
  - Ongoing research for over 3 years
  - Evolved from analyzing and writing virus detectors
  - Impacted by failures in using traditional analysis
  - Aim: fundamental advances in hardening analysis
    - focus: key (real) problems in malware analysis
    - develop and adapt theoretical approaches
    - build and test prototypes

## Our ACA Approach

- Short term
  - Harden individual steps
  - Use solid theory
- Long term
  - Holistic infrastructure improvement





### Motivation

- Analysis Tools
- Overall ACA Approach
- Results
  - VILO: malware phylogeny generation
  - DOC: detecting obfuscated calls
    - UMPH: reversing metamorphic transforms
  - Future and Conclusions

### Program Analysis: The Old Frontier

- Half a century of program analysis
  - Compilers, optimizers, checkers, refactoring tools
  - Analyze, visualize, transform
- Problem Space
  - Program optimization
  - Profiling, testing, debugging
  - Understanding, Comprehension
  - Reengineering
- Purpose
  - Help programmers help themselves



### Program Analysis: The New Frontier

- Analysis of malicious programs
  - Viruses, worms, Trojans, spyware, adware
- Problems
  - What does the malware do?
  - What attack tools and methods are employed?
  - How did it arrive on this computer?
  - Which other computers did it go to?
  - Who wrote the malware?
- Purpose
  - Help security analysts *defend* computing resources



## Implications of Undecidability











#### **Black Hat Briefings**

14



- Transformations destroy signature/pattern match
  - eg metamorphic viruses: self-transforming
  - Instruction substitution, nop insertion, etc.

## Attacks on Signature Analysis

#### Polymorphic malware

- Code is encrypted
- Carries a decryptor
- Decryptor transformed before propagation
- Metamorphic malware
  - Whole code transformed before propagation
  - So far threat mostly 'in-the-zoo' so far
  - Off-the-shelf metamorphic engines available, improving
- Packed malware
  - Rapidly release variants packed by different packers
  - Overwhelm the security analysts

## **Current AV Infrastructure**

### • Human intensive

- Analysts specialize on specific attacks
  - In leading companies, person(s) dedicated to deal with packers

17

- Knowledge resident in specialists
- High workload
  - Spyware may have few HUNDRED programs
  - About 5-8 email samples per analyst per day

## **Current AV Infrastructure**

- Depend on tools not designed for the trade
  - Disassemblers, debuggers, program monitors
  - No methodical way to organize knowledge
    - Rely on Google
  - The Bright Side
    - Significant advances in dynamic analysis
    - Metamorphic viruses detected by emulation
      - Has its own set of issues

## Talk Contents

Motivation

Analysis Tools

Overall ACA Approach

Results

- VILO: malware phylogeny generation
- DOC: detecting obfuscated calls
  - UMPH: reversing metamorphic transforms

Future and Conclusions



## **Configure Environment**

- Requirements
  - Prevent contamination of production systems
  - Quickly undo damage
  - Allow interaction among multiple systems
- Goals
  - Prep files and operating systems for infection
  - Initialize analysis tools

### Black Hat Briefings

COMMON TOOLS

21

VMWare

MS Virtual PC

## **Static Analysis**

- Goals
  - Quickly identify key program features
    - does it send mail?
    - ... open an IRC channel?
    - ... kill processes?
  - Quickly identify possible malicious intent

Strings	
BinText	
IDA Pro	

**COMMON TOOLS** 

### **Dynamic Analysis**

- Goals
  - Identify process activity
    - are processes created/killed?
  - Identify hard disk actvity
  - Identify network activity
  - Identify registry changes
- Used when deeper understanding is required

#### **COMMON TOOLS**

Process Explorer FileMon

RegMon

RegShot

ProcDump

IDA Pro

OllyDbg

# Talk Contents

- Motivation
- Analysis Tools
- Overall ACA Approach
- Results
  - VILO: malware phylogeny generation
  - DOC: detecting obfuscated calls
    - UMPH: reversing metamorphic transforms
  - Future and Conclusions



- Assumption #1:
  - Programmers and analysis tools have common goal

### Reality

 Programmer (writer of code under scrutiny) and tools are adversaries

- Assumption #2:
  - Malware authors have the benefit of surprise

- Reality
  - High degree of reuse and plagiarism
    - Jaschan (2004), author of Sasser, copied from Lovesan

29

• Others immediately picked up on his ideas.



- Reality (cont.)
  - No big bang; malware also evolves
    - Beagle versions from A, B, .., AA, to ED
    - Each version introduced small change
  - Inventions discussed in Blackhat forums
    - Format string attack; attack on Oracle
  - Vulnerability and exploits often first found by security analysts
  - Implication
    - Utilize knowledge outside of code under scrutiny

- Assumption #3
  - Undecidability a hindrance

- Reality
  - Analysis tools can live with 'statistical' equivalence
    - Need statistical 'safety', not theoretical safety
  - Undecidability is a two-edged sword
    - Self-transforming code must analyze itself
    - Must deal with undecidability too
      - Metamorphic virus W32.Evol does not use any disassembly attack
    - Not easy to exploit
      - Trend has moved to packed malware
  - Complete obfuscation is impossible [Barak et al. 2001]
  - Implication
    - Develop targeted deobfuscators

### Vision for an Analysis Infrastructure

- Day in the life of an analyst
  - Arrive at work
  - Analyze a sample
    - Sample pre-analyzed, relation with other malware annotated
    - Review, verify annotations
    - Move on to next sample, if satisfied with findings
    - Analyze un-annotated parts
      - Use an integrated environment with dynamic/static tools
      - Apply various deobfuscators to discover hidden meaning
    - Add annotations of finding into the knowledge base
  - Move on to the next sample

## **Core Capabilities Needed**

- Hardened static analysis
  - All phases should be:
    - semantic driven
    - interleaved
    - utilize knowledge base
  - New questions, new algorithms
    - Can a variable have a certain value on some path?
    - What if traditional procedural units do not exist?
    - Probabilistic analysis

## **Core Capabilities Needed**

- Integrated security analysis environments
  - Integrate dynamic and static analysis
  - Knowledge base
  - Comparison of code fragments
    - Catch evolutionary relation between families, and within family
  - Deobfuscators, targeted
    - Undo call obfuscations, key to determining behavior
    - Undo transformations






#### How to Name and Classify

# **Symantec** W32.NetSky.A W32.NetSky.B

W32.NetSky.D W32.Beagle.A@mm W32.Beagle.J@mm W32.Beagle.AO@mm W32.Beagle.U@mm

W32.Klez.E@mm.enc W32.Klez.F@mm W32.Klez.l@mm

W32/NetSky.A W32/NetSky.B W32/Bugbear.17916intd W32/Bagle.a@mm W32/Bagle.j@mm

**McAfee** 

W32/Bagle.aq@mm W32/Bagle.u@mm

W32/Klez.e@MM W32/Klez.f@MM W32/Klez.i@MM

#### Black Hat Briefings

39

VILO

## Generating Phylogeny Model

*phylogeny*: evolutionary relationships between organisms

Use cluster analysis

- Key need
  - A similarity measure
    - N-grams based measure not-effective
      - Cannot account for permutations
    - Developed N-perm similarity measure
      - Influenced by bio-informatics



#### Example: Permuted Netsky worn-

VILO

I2D2:	push	ecx	1144:	push	<b>ecx</b>	
	push	4		push	4	
	рор	ecx		рор	ecx	
	push	ecx		push	ecx	
<b>I2D7</b> :	rol	edx, 8	1149:	mov	di, al	
	mov	dl, al		and	dl, 3Fh	
	and	dl, 3Fh		rol	edx, 8	
	shr	eax, 6		shr	ebx, 6	
	loop	l2D7		loop	1149	
	рор	ecx		рор	ecx	
	call	s319		call	s52F	
	xchg	eax, edx		xchg	ebx, edx	
	stosd			stosd		
	xchg	eax, edx		xchg	ebx, edx	
	inc	[ebp+v4]		inc	[ebp+v4]	
<u> </u>	cmp	[ebp+v4], 12h		cmp	[ebp+v4], 12h	
	jnz	short I305		jnz	short I18	
						41
	I a c k	e a te erte	tinds			









### Evaluation

- Question
  - Are the models useful for classifying new malware?
  - Process
    - 170 known malware
      - From VXHeavens archive
    - Three unknown worms (A, B, C)
      - Captured by AV scanner on mail gateway
    - Place unknown samples using n-grams and nperms

#### Black Hat Briefings

VILO

### Results

VILO

47

- N-perm classification better in:
  - -Clustering distinct malware classes
  - Classifying unknown clusters with close relatives
  - Identifying naming conflicts

	10-pe	rm P	hylog	eny	VILO
		VX Heavens	Norton	McAfee	ClamAV
- 5		I-Worm.Mydoom.q		M in the second second	Worm.Mydoom.S-unp
10 10		I-Worm.Mydoom.u	MyDoo	ms m	Worm.Mydoom.Gen-unp
Ŋ		I–Worm.Mydoom.g	waz.myadom.G@mm	myaaam.gen@mm	Worm.Mydoom.Gen-unp
9		Win32.Elkern.a	W32.ElKern.gen	W32.ElKern.cav.a	Worm.Klez.E
20		I–Worm.Klez.a	W32.Klez.A@mm	Klez.worm.gen	Worm.Klez.E
ci.	8	I-Worm.Klez.i	Klez/Elk	erns	Worm.Klez.H
	6.	I-Worm.Klez.f			Worm.Klez.E
4	٥	I-Worm.Klez.e	W32.Klez.E@mm	Klez.e@mm	Worm.Klez.E
<b>-</b>	3	Specimen–C *	W32.Elkern.4926	W32.Elkern.cav.c	Worm.Bagle.Gen–dll
	.6	I–Worm.Bagle.al	W32.Beagle.AO@mm	Bagle.dll.dr	Worm.Bagle.Al
		Specimen_A *	not detected	Bagle.gen@mm	Trojan.Spamtool.Small.F
	<sup>45</sup> [9]	Specimen-B *	Deer		Trojan.Spamtool.Small.F
g	<u>.</u>	I-Worm.Bagle.s	Beagi	es <sub>n</sub>	Worm.Bagle.Gen-dll
		I–Worm.Bagle.a	W32.Beagle.A@mm	Bagle.a@mm	Worm.Bagle.Gen-dll
	.9	I-Worm.Bagle.i	W32.Beagle.J@mm	Bagle.j@mm	Worm.Bagle.Gen-dll
	٥.	I–Worm.Bagle.j	W32.Beagle.gen	Bagle.k@mm	Worm.Bagle.K-unp
		<b>10-pe</b>	Independence  VX Heavens    VX Heavens  I-Worm.Mydoom.q    I-Worm.Mydoom.q  I-Worm.Mydoom.q    I-Worm.Mydoom.q  I-Worm.Mydoom.q    I-Worm.Klez.a  I-Worm.Klez.a    I-Worm.Klez.a  I-Worm.Klez.a    I-Worm.Klez.a  I-Worm.Klez.a    I-Worm.Klez.a  I-Worm.Klez.a    I-Worm.Klez.a  I-Worm.Klez.a    I-Worm.Klez.a  I-Worm.Bagle.a    I-Worm.Bagle.a  I-Worm.Bagle.a    I-Worm.Bagle.a  I-Worm.Bagle.a    I-Worm.Bagle.a  I-Worm.Bagle.a    I-Worm.Bagle.a  I-Worm.Bagle.a	VX Heavens  Norton	Instruction  Instruction  Marchage    VX Heavens  Norton  McAfee    VX Heavens  Norton  McAfee    VX Heavens  I-Worm.Mydoom.q  MyDooms  m    VY Heavens  I-Worm.Mydoom.q  Woz.Hydoom.q  Mydoom.qengm  m    VY Heavens  I-Worm.Mydoom.q  Woz.Hydoom.qengm  Mydoom.qengm  Mydoom.qengm    VY Heavens  I-Worm.Klez.i  I-Worm.Klez.i  Win32.Elkern.ae  Win32.Elkern.gen  Wisz.Elkern.cav.a    VY Howrm.Klez.i  I-Worm.Klez.i  I-Worm.Klez.i  Wisz.Elkern.426  Wisz.Elkern.cav.c    VY Howrm.Klez.i  I-Worm.Bagle.ai  Specimen-C*  Wisz.Elkern.426  Wisz.Elkern.cav.c    VY Mydoom.u  Specimen-C*  I-Worm.Bagle.ai  Nisz.Elkern.426  Wisz.Elkern.cav.c    VY Mydoom.u  Specimen-A*  Not detected  Bagle.gen@mm  Bagle.gen@mm    Mydoom.u  I-Worm.Bagle.i  I-Worm.Bagle.i  Wisz.Beagle.J@mm  Bagle.j@mn    Mydoom.u  I-Worm.Bagle.i  Wisz.Beagle.J@mm  Bagle.j@mm  Bagle.j@mm

#### **Black Hat Briefings**

48

### Summary of VILO

VILO

49

- Impact
  - It is feasible to utilize historic knowledge
  - Can even be used on the desktop
    - Detect new malware

**Black Hat Briefings** 

– Improve forensics

### Summary of VILO

- Open issues
  - Scaling for  $O(10^4-10^5)$  data set
  - Visualization for exploring large space of relations
  - Online/incremental classification

#### Black Hat Briefings

VILO





### Problem

- Determine unconventional control transfers statically
  - Implicit calls
- Determine "bogus" returns statically
  - Return address modification

#### **Black Hat Briefings**

### Approach

- **Abstract Interpretation** 
  - Operations are interpreted to operate over an *abstract domain* (rather than on real data)
  - Real-world properties are translated into abstract properties of interest

#### Black Hat Briefings





### Our Domain

- Concrete domain
  - Runtime stack
    - tracks actual program data
- Abstract domain
  - Abstract Stack Graph (ASG)
    - tracks all stack-manipulation (push, pop, call, etc.)

#### Black Hat Briefings

### Abstract Stack

- Holds <u>addresses</u> of <u>instructions</u> pushing data onto stack
  - not the data
  - not the instruction





#### Uses of ASG

- Detect obfuscations
  - call obfuscations (e.g., push-push-ret)
  - obfuscation of parameters to a call
  - obfuscated return
  - manipulation of return address
- Match call / return instructions
  - return instruction need not follow entry point

#### Black Hat Briefings

### Prototype

🗐 inp	ut.asm 🔀						Stack 🕅
40	1000h	PUSH	402000h				
<b>G</b> 40	1005h	CALL	KERNEL32.SetCurrentDi	rectoryA		П	N1
40	100ah	PUSH	40200dh	-		≡	Value: (402000
40	100fh	PUSH	402005h				Creator: 40100
<b>G</b> 40	1014h	CALL	KERNEL32.FindFirstFil	eΑ			
40	1019h	CMP	eax, -1				↓
40	101ch	JE	401042h				NO
40	101eh	MOV	dword ptr ds: [4202827	], eax			Value: (T,
40	1023h	PUSH	402039h				
<b>G</b> 40	1028h	CALL	KERNEL32.DeleteFileA				
40	102dh	PUSH	40200dh				
40	1032h	PUSH	dword ptr ds: [4202827	1			
9 40	1038h	CALL	KERNEL32.FindNextFile	A			
40	103dh	CMP	eax, 0				
40	1040h	JNZ	401023h				
	10420	CALL	U VEDNET 22 EwitDacases				
9 4	TOAAU	CALL	ALKWEL32.EXITPROCESS			~	
					2		]
Messag	s 🕞 Va	lid Call-Ret	rn Sites 🛛 💦 bfuscated Calls 🛛 O	ofuscated Returns	Associated Instructi		
Line	in mucti	on			1		
17	CALL KE	RNEL32.EX	tProcess				
2	CALL KE	RNEL32.Se	tCurrentDirectoryA				
5	CALL KE	RNEL32.Fi	IdFirstFileA				
10		RNEL32.De	leterileA dNextFileA				
1.5	CALL NE	INNELUZ.FI					
<u> </u>							
1 1							
						_	

61

#### Prototype

DOC

62

-					
		- 🍋			⊡ Obfuscation
E	input.asm 🛛				Stack 🛛
	401000h	PUSH	402000h	<b>^</b>	
G	401005h	CALL	KERNEL32.SetCurrentDirectoryA		N12
	40100ah	PUSH	40200dh		Value: (40200dh, T)
	40100fh	PUSH	402005h		Creator: 401039h
	401014h	PUSH	40101fh		
	401019h	PUSH	KERNEL32.FindFirstFileA		V
⇒	40101eh	RET			N0
	40101fh	CMP	eax, -1		Value: (T. T)
	401022h	JE	401054h		
	401024h	MOV	dword ptr ds:[4202827], dx		
	401029h	PUSH	402039h		
	40102eh	PUSH	401039h		
	401033h	PUSH	KERNEL32.D leteFileA		
⇒	401038h	RET			
	401039h	PUSH	40200dh		
	40103eh	PUSH	dword ptr ds:[4202827]		
	401044h	PUSH	40104fh		
	401049h	PUSH	KERNEL32.FindNextFileA		
⇒	40104eh	RET			
	104fh	CMP	eax, 0		
	401952h	JNZ	401029h		
	40105.h	PUSH	0		
G	401056h	CALL	KERNEL32.ExitProcess	▼	
	<			>	
Ma	seages Valid C	all-Deturn			
THC.					
-		/11		^	
14	RET				
19	RET				
67	7 RET 8				
69	RET 4				

**Black Hat Briefings** 

### Summary of DOC

- Impact
  - Detected all call obfuscations in W32.Evol
  - Initial step towards semantic disassembler

#### **Black Hat Briefings**

### Summary of DOC

- Open issues
  - Indirect stack operations
    - through memory and other registers
  - Attacks on abstract interpretation
    - Explode size of state
    - Hide in over approximation

#### Black Hat Briefings



#### Metamorphic malware



QUESTION: If T is non-deterministic, is detection by signature possible?

**Black Hat Briefings** 

66

UMPH

#### Example

UMPH

push ecx



### Goal

- Reduce variants to unique "normal" form
- Detect all variants using a single signature

### Approach

- Extract transformations used by malware
- Model mutation engine as term rewriting system (T)
- Construct a Normalizer (N) for T
  - Use length-reducing,
  - Lexicographic ordering to re-orient,
  - And yield finite length-reducing rewriting system.
- Apply reverse transforms to unmorph the malware
  - Staged Priority without completion (WC)
  - Staged Priority with manual completion
  - Simply use the automatically completed set
    - Knuth-Bendix completion procedure





### Evaluation

- Case study
  - Unmorph W32.Evol
- Process
  - Created 72 variants over six generations
    - Chose 26 variants for reversal
  - Extracted rules used by W32.Evol
    - 55 rules (with non-ground terms)
  - Reverted rules and added completion rules
## **Evaluation** (cont)

- Results
  - With manual completion of rules
    - All 26 variants reverted to a single, unique normal form
  - Without completion (WC)
    - Normal forms of all 26 variants showed more than 98% similarity
    - Can be exploited to extract a single signature to match all

# Results: Without Completion

Generation	Eve	2	3	4	5	6
Avg. size of original	2182	3257	4524	5788	6974	8455
Max. size of normal form	2167	2167	2184	2189	2195	2204
Avg. size of normal form	2167	2167	2177	2183	2191	2204
Lines not in common	0	0	10	16	24	37
% in common	100.0	100.0	99.54	99.27	98.90	98.32
Execution time (ms)	2469	3034	4264	6327	7966	11219
Rule Counts	16	533	980	1472	1902	2481
			and the second second	2		74

**Black Hat Briefings** 

### **Example - Reversed**

UMPH

片는



## Summary of UMPH

- Impact
  - Better than we expected
  - Can raise the bar very high for malware authors

#### **Black Hat Briefings**

UMPH

## Summary – Open Issues

#### How to extract/gather transformation rules?

- Studying samples 'in-the-zoo'
- Creating own equivalent transformations
- How to deal with semantic non-preserving transformations?
  - Malware may introduce dead/irrelevant code
  - Reversing the rules may be problematic
    - W32.Evol had such a rule
    - We gave least priority to its reverse rule
- How to complete the rules?
  - Knuth-Bendix procedure is not guaranteed to terminate
  - Use rule set specific knowledge
    - We added only TWO rules for completion

# Talk Contents

Motivation Analysis Tools **Overall ACA Approach** Results VILO: malware phylogeny generation DOC: detecting obfuscated calls **UMPH:** reversing metamorphic transforms Future and Conclusions

## ACA in the Future

- Beginnings of movement in academic research
  - Seeing a few papers on relevant topics
    - disassembly, de-obfuscation, phylogeny
    - largely ignored by academic community
  - Some appreciation of ACA vision
    - feeding back to prior stages
    - history-directed analysis

## ACA in the Future

- Our focus: work with industry
  - refine vision, keep focus on important issues
    - believe we can drive important research this way
    - VILO, DOC, and UMPH building blocks

# Credits

Software Research Lab Center for Advanced Computer Studies University of Louisiana at Lafayette

> Arun Lakhotia Director

Andrew Walenstein Research Scientist

Michael Venable Software Engineer and Alumni

Ph.D. Students Mohamed Chouchane Md Enamul Karim

> M.S. Student Rachit Mathur

#### Alumni

Nitin Jyoti, Avertlabs Aditya Kapoor, McAfee Erik Uday Kumar, Authentium Moinuddin Mohammed, Microsoft Prashant Pathak, Symantec Prabhat Singh, Symantec

Funded by: Louisiana Governor's IT Initiative

## References

- A. Lakhotia, E. U. Kumar, and M. Venable, "A Method for Detecting Obfuscated Calls in Malicious Binaries," *IEEE Transactions on Software Engineering*, 2005 (in print).
- M. Karim, A. Walenstein, A. Lakhotia, and L. Parida, "Malware Phylogeny Generation using Permutations of Code," *European Research Journal of Computer Virology*, 2005 (in print).
- M. Venable, M. R. Chouchane, Md. E. Karim, and A. Lakhotia, "Analyzing Memory Accesses in Obfuscated x86 Executables," K. Julisch and C. Kruegel (Eds.): *Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) 2005, LNCS 3548*, Springer-Verlag Berlin Heidelberg, pp. 1 18, 2005
- A. Lakhotia, M. E. Karim, A. Walenstein, and L. Parida, "Malware Phylogeny using Maximal Pi-Patterns", *EICAR 2005 Conference: Best Papers Proceedings*, Malta, pp. 156-174, April-May, 2005.
- A. Lakhotia and M. Mohammed, "Imposing Order on Program Statements and its implication to AV Scanners," in *Proceedings of 11th IEEE Working Conference on Reverse Engineering*, Delft, The Netherlands, November 2004, pp. 161-171.
- U. K. Eric, A. Kapoor, and A. Lakhotia, "DOC- Answering the Hidden 'Calls' of Virus," *Virus Bulletin,* April 2005.
- A. Lakhotia, A. Kapoor, and E. U. Kumar, "Are Metamorphic Viruses Really Invincible?" *Virus Bulletin*, December 2004 and January 2005.
- M. Venable, P. Pathak, and A. Lakhotia, "Getting into Beagle's Backdoor," *Virus Bulletin*, July 2004, pp. 9-13.
- A. Lakhotia and P. K. Singh, Challenges in getting 'Formal' with viruses, *Virus Bulletin*, September 2003.

#### Black Hat Briefings