The Remote Metamorphic Engine
Detecting, Evading, Attacking the AI and Reverse Engineering

Amro Abdelgawad / DEFCON 24
The Remote Metamorphic Engine

- Security as undefined expression
- Flux binary mutation
- Resisting Reverse Engineering
- Evading AI machine learning
- Artificial Immunity
Security Patterns

Division by Zero | Division by Infinity

Isolation | Randomization
The Undefined Expression

Security as Undefined & indeterminate expression

-∞ = 1/0 = ∞

RE Time

The Remote Metamorphic Engine
The Unbreakable Code

Unpredictable
un·pre·dict·a·ble
adjective:/ˌənprɛˈdɪktəb(ə)l/

 Likely to change suddenly and without reason
 and therefore not able to be predicted
 (= expected before it happens)
The Breakable Code

The Fixed Static Code Problem

Static Code Dynamic Data

Core security weakness in all today’s software

Enables all sorts of replicable software security exploits
Unpredictable Code Evolution

Dynamic Code Dynamic Data

- Code evolution across time
- Functionality evolution across location
- Self contained autonomous code
- Unpredictable
- Self aware
Code Evolution

Resisting Reverse Engineering

- Locate the Code
- Remote Execution
  - Not locatable
- Analyze the Code
- Short Lifetime
  - Flux Mutation
- Break the Code
  - Self aware
  - Unbreakable
The Remote Metamorphic Engine

Remote Flux Mutation

Trusted Zone
Remote Mutation
Mutation Engine
Challenge

Untrusted Zone
Thread/Process
Morphed Code Execution
Response
Why Remote?
The Remote Metamorphic Engine

Challenge Response Metamorphic Protocol

- **Trusted Zone**
  - Remote Mutation
  - Mutation Engine

- **Challenge**
  - 4 bytes size
  - Code
  - Clock Synced

- **Response**

- **Untrusted Zone**
  - Thread/Process
  - Morphed Code Execution

Communication protocol made of morphed clock synchronized machine code rather than data
Why Metamorphic?
The Remote Metamorphic Engine

Remote Code Slicing

The Reverse Engineer Side

Known to the reverse engineer

The Engine Side

Unknown to the reverse engineer
Demo 1
Mutation Engines

AV Signature Evasion

Polymorphic Engines
morphed body encryption

Metamorphic Engines
body polymorphic
## Signature Evasion

Morphing Techniques Evading Signature Signature

<table>
<thead>
<tr>
<th>Instruction reordering</th>
<th>Code Permutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subroutine permutation</td>
<td>Instruction Substitution</td>
</tr>
<tr>
<td>Subroutine Inlining</td>
<td>Dead Code Insertion</td>
</tr>
<tr>
<td>Subroutine Outlining</td>
<td>Changing Control Flow</td>
</tr>
<tr>
<td>Expansion</td>
<td>Transposition</td>
</tr>
</tbody>
</table>

Can not resist reverse engineering
Remote Code Evolution

Flux Mutation Goals

- Extend Trust
- Ensure Trusted Remote Execution
- Evade Signature
- Evade AI Machine Learning
- Detect & Evade RE
- Detect Tampering Attempts
Trusted Mutation

Trusted Challenge Response Mutation

Remote Mutation → Morphing Engine

Morphed Function → Challenge

Function

Morphed Function

Head

Unused Code

Return value

Tail

Response Mutation
Structure Obfuscation

All functions look the same before and during execution
Structure Obfuscation

Self modifying basic block Edges
Demo 2
RE Evasion

Morphing Techniques

- Metamorphic + Polymorphic
- Self modifying mutation
- Code structure obfuscation
- Clock synchronized execution
- Challenge-Response Mutation
- Functionality Mutation
- Decoupled Reversible Mutation
- Slices Permutation
- Code size magnification
Remote Code Evolution

Morphing Techniques

```
_start:
push 0
pushad
mov reg1, [fs:dword 0x30]
movzx reg2, byte [reg1+2]
mov dword [esp+32], reg2
popad
pop eax
ret
```

end:
Remote Code Evolution

Morphing Techniques

```asm
_start:
push 0
    xor reg1, reg1
    push reg1
    pushad
    mov reg1, [fs:dword 0x30]
    movzx reg2, byte [reg1+2]
    mov dword [esp+32], reg2
    popad
    pop eax
    ret
end:
```
Remote Code Evolution
Morphing Techniques

_start:

push 0
{xor reg1, reg1
push reg1
pushad
Insertion→sub reg1, reg1
mov reg1, [fs:dword 0x30]
movzx reg2, byte [reg1+2]
mov dword [esp+32], reg2
popad
pop eax
ret

end:
Remote Code Evolution
Morphing Techniques

```assembly
_start:
    push 0
    {  xor reg1, reg1
      push reg1
      pushad
      Insertion → sub reg1, reg1
      mov reg1, [fs:dword 0x30]
      Insertion → add reg2, reg2
      movzx reg2, byte [reg1+2]
      mov dword [esp+32], reg2
      popad
      pop eax
      ret
    }

end:
```
Remote Code Evolution
Morphing Techniques

_start:
push 0 {
xor reg1, reg1
push reg1
pushad
Insertion → sub reg1, reg1
mov reg1, [fs:dword 0x30]
Insertion → add reg2, reg2
movzx reg2, byte [reg1+2]
Insertion → mov reg3, reg4
mov dword [esp+32], reg2
popad
pop eax
ret
}
Remote Code Evolution

Morphing Techniques

_start:

push 0

{xor reg1, reg1
push reg1
pushad
sub reg1, reg1
mov reg1, [fs:dword 0x30]

Insertion ➔ sub reg1, reg1
mov reg1, [fs:dword 0x30]

Insertion ➔ add reg2, reg2
movzx reg2, byte [reg1+2]

Insertion ➔ mov reg3, reg4
mov dword [esp+32], reg2

n*nop ➔ popad
pop eax
ret

dw 3
pushad
pushf
call line95_1

mov cl, 0xe9
mov byte [eax], cl
xor edx, 0
mov ecx, 0x00000057
mov dword [eax+1], ecx
ret

Remote Code Evolution

Morphing Techniques

_start:

push 0

{xor reg1, reg1
push reg1
pushad
sub reg1, reg1
mov reg1, [fs:dword 0x30]

Insertion ➔ sub reg1, reg1
mov reg1, [fs:dword 0x30]

Insertion ➔ add reg2, reg2
movzx reg2, byte [reg1+2]

Insertion ➔ mov reg3, reg4
mov dword [esp+32], reg2

n*nop ➔ popad
pop eax
ret

dw 3
pushad
pushf
call line95_1

mov cl, 0xe9
mov byte [eax], cl
xor edx, 0
mov ecx, 0x00000057
mov dword [eax+1], ecx
ret
Remote Code Evolution

Morphing Techniques

_start:

push 0

{ xor reg1, reg1
    push reg1
    pushad

    Insertion → sub reg1, reg1
    mov reg1, [fs:dword 0x30]

    Insertion → add reg2, reg2
    movzx reg2, byte [reg1+2]

    Insertion → mov reg3, reg4
    mov dword [esp+32], reg2

    n*nop
    popad
    pop eax
    add esp, 36
    push reg2
    ret

    end:

    sub esp, 32
Remote Code Evolution
First Morphing Stage

_start:
    xor reg1, reg1
    push reg1
    pushad
    sub reg1, reg1
    mov reg1, [fs:dword 0x30]
    add reg2, reg2
    movzx reg2, byte [reg1+2]
    mov reg3, reg4
    mov dword [esp+32], reg2
    popad
    nop
    pop eax
    nop
    ret

end:
Remote Code Evolution
Second Morphing Stage

```assembly
line1:
    xor edi, edi
    jmp long line2
line2:
    push edi
    jmp long line3
line3:
    pushad
    jmp long line4
line4:
    sub edi, edi
    jmp long line5
line5:
    jmp long line6

line6:
    add ebx, ebx
    jmp long line7
line7:
    movzx ebx, byte [edi+2]
    jmp long line8
line8:
    mov ecx, edx
    jmp long line9
line9:
    mov dword [esp+32], ebx
    jmp long line10
line10:
    nop
    jmp long line11

line11:
    popad
    jmp long line12
line12:
    nop
    jmp long line13
line13:
    pop eax
    jmp long line14
line14:
    nop
    jmp long line15
line15:
    ret
    jmp long line16
```
Remote Code Evolution

Third Morphing Stage

```assembly
line1:    xor edi, edi
           jmp long line2

line6:    add ebx, ebx
           jmp long line7

line8:    mov ecx, edx
           jmp long line9

line15:   ret
           jmp long line16

line11:   popad
           jmp long line12

line14:   nop
           jmp long line15

line13:   pop eax
           jmp long line14

line3:    pushad
           jmp long line4

line4:    sub edi, edi
           jmp long line5

line9:    mov dword [esp+32], ebx
           jmp long line10

line12:   nop
           jmp long line13

line5:    jmp long line6

line7:    movzx ebx, byte [edi+2]
           jmp long line8

line2:    push edi
           jmp long line3

line10:   nop
           jmp long line11```

Third Morphing Stage
Remote Code Evolution
Self Modifying Body Polymorphism

Forth Morphing Stage

Random Obfuscation Keys

| 5 | 1 | -1 | 0 | 27 | 4 | 3524080526 | 0 | 7 | 2 |

Self modifying instructions

| mov eax, 93 |
| add ecx, eax |
| mov eax, ecx |
| mov ebx, 0x11223344 |
| not ebx |
| mov [ecx], ebx |
| add ecx, 4 |
| xor dword [ecx], 0x11223344 |
| add ecx, 4 |
| xor dword [ecx], 0x11223344 |
| add ecx, 4 |
| xor dword [ecx], 0x11223344 |
| add eax |
| jmp eax |

Self Modifying
Self Modifying Blocks

Fifth Morphing Stage

Obfuscation Keys

One block per morphed instruction

All blocks have same identical structure

Self modifying code
Self Modifying Blocks

```assembly
PUSHAD
PUSHFD
CALL 00240C59
ADD AL, 4
XCHG EAX, EDI
CMP DH, BYTE PTR DS:[EAX]
ROL BYTE PTR DS:[ECX], 1
ADD EDI, EDI
INC DWORD PTR SS:[ESP+EDX]
DEC EBP
INT 0F0
NOP
MOV ECX, 54
```
Response Time

[+] mutated code size: 15110 bytes
[+] encrypted response: 0x09575e31 | 156720689
[+] decrypted response: 0x00000001 | 1
[+] remote execution response time: 6.685972 ms

[+] mutated code size: 17771 bytes
[+] encrypted response: 0x5820b6b5 | 1478538933
[+] decrypted response: 0x00000001 | 1
[+] remote execution response time: 6.040096 ms

[+] mutated code size: 23814 bytes
[+] encrypted response: 0x5d844e9a | 1568951962
[+] decrypted response: 0x00000001 | 1
[+] remote execution response time: 6.897926 ms

[+] mutated code size: 19768 bytes
[+] encrypted response: 0x818af8d8 | -2121598760
[+] decrypted response: 0x00000001 | 1
[+] remote execution response time: 6.177187 ms
Variable Code Size

[+]
mutated code size: 15110 bytes
encrypted response: 0x09575e31 | 156720689
decrypted response: 0x00000001 | 1
remote execution response time: 6.685972 ms

[+]
mutated code size: 17771 bytes
encrypted response: 0x5820b6b5 | 1478538933
decrypted response: 0x00000001 | 1
remote execution response time: 6.040096 ms

[+]
mutated code size: 23814 bytes
encrypted response: 0x5d844e9a | 1568951962
decrypted response: 0x00000001 | 1
remote execution response time: 6.897926 ms

[+]
mutated code size: 19768 bytes
encrypted response: 0x818af8d8 | -2121598760
decrypted response: 0x00000001 | 1
remote execution response time: 6.177187 ms
```assembly
mov ecx, [esp]
nop
nop
mov dl, 0xe9
test edx, edx
mov byte [ecx], dl
xor eax, 0
mov edx, 0x00000067
mov dword [ecx+1], edx
ret

pushad
pushf
call line95_1
db 7
db 3
dd 838225172
db 2
dd 4211932376
db 4
dd 2520091426
db 3
dd 946381070
db 2
dd 3318121790
db 2
dd 1375432265
db 1
dd -1
mov ebx, 92
add eax, ebx
mov ebx, eax
sub dword [eax], 0xe82c334d
add eax, 4
add dword [eax], 0xa1723594
add eax, 4
xor dword [eax], 0xb1c21343
add eax, 4
sub dword [eax], 0x111111ee
add eax, 4
add dword [eax], 0xaaccee22
add eax, 4
jmp ebx
line95_2:
popf
popad
nop
jmp long line96
line95_1:
mov eax, [esp]
nop
nop
xor eax, eax
xor ecx, ecx
xor edx, edx
mov cl, 0xe9
mov byte [eax], cl
xor edx, 0
mov ecx, 0x00000057
mov dword [eax+1], ecx
ret
```
Decoupled Reversible Mutation

Response Mutation

Morphing Engine → Mutated Challenge

Remote Mutation

Morphed Function

Function → Morphed Function

Head

Morphed Function

Unused Code

Return value → Reversible Mutation

Tail
Decoupled Reversible Mutation

Reversible Instructions

- add(value1)
- sub(value2)
- not()
- xor(value3)
- rol(value4)
- ror(value5)

- rol(value5)
- ror(value4)
- xor(value3)
- not()
- add(value2)
- sub(value1)
### The Remote Metamorphic Engine

**Artificial Immunity | Detecting the non-self**

<table>
<thead>
<tr>
<th>Mutations</th>
<th>Responses</th>
<th>Decrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>156720689</td>
<td>0</td>
</tr>
<tr>
<td>2nd</td>
<td>147853893</td>
<td>0</td>
</tr>
<tr>
<td>3rd</td>
<td>15689519</td>
<td>0</td>
</tr>
<tr>
<td>4th</td>
<td>-21215987</td>
<td>0</td>
</tr>
<tr>
<td>5th</td>
<td>10778328</td>
<td>137106</td>
</tr>
<tr>
<td>6th</td>
<td>-689519</td>
<td>0</td>
</tr>
<tr>
<td>7th</td>
<td>11979087</td>
<td>0</td>
</tr>
</tbody>
</table>

**Tampered non-self**
The Remote Metamorphic Engine

Artificial Immunity | Detecting the non-self

<table>
<thead>
<tr>
<th>Mutations</th>
<th>Response Time</th>
<th>Time Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>47 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>2nd</td>
<td>65 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>3rd</td>
<td>52 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>4th</td>
<td>106 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>5th</td>
<td>579 ms</td>
<td>&gt;500 ms</td>
</tr>
<tr>
<td>6th</td>
<td>39 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>7th</td>
<td>53 ms</td>
<td>&lt;500 ms</td>
</tr>
</tbody>
</table>

Emulated

non-self

1st
2nd
3rd
4th
5th
6th
7th

Response Time

Time Threshold

Mutations

Emulated non-self

<500 ms
<500 ms
<500 ms
<500 ms
>500 ms
<500 ms
<500 ms

47 ms
65 ms
52 ms
106 ms
579 ms
39 ms
53 ms

The Remote Metamorphic Engine

Artificial Immunity | Detecting the non-self

<table>
<thead>
<tr>
<th>Mutations</th>
<th>Response Time</th>
<th>Time Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>47 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>2nd</td>
<td>65 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>3rd</td>
<td>52 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>4th</td>
<td>106 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>5th</td>
<td>579 ms</td>
<td>&gt;500 ms</td>
</tr>
<tr>
<td>6th</td>
<td>39 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>7th</td>
<td>53 ms</td>
<td>&lt;500 ms</td>
</tr>
</tbody>
</table>

Emulated

non-self

1st
2nd
3rd
4th
5th
6th
7th

Response Time

Time Threshold

Mutations

Emulated non-self

<500 ms
<500 ms
<500 ms
<500 ms
>500 ms
<500 ms
<500 ms

The Remote Metamorphic Engine

Artificial Immunity | Detecting the non-self

<table>
<thead>
<tr>
<th>Mutations</th>
<th>Response Time</th>
<th>Time Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>47 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>2nd</td>
<td>65 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>3rd</td>
<td>52 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>4th</td>
<td>106 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>5th</td>
<td>579 ms</td>
<td>&gt;500 ms</td>
</tr>
<tr>
<td>6th</td>
<td>39 ms</td>
<td>&lt;500 ms</td>
</tr>
<tr>
<td>7th</td>
<td>53 ms</td>
<td>&lt;500 ms</td>
</tr>
</tbody>
</table>

Emulated

non-self

1st
2nd
3rd
4th
5th
6th
7th

Response Time

Time Threshold

Mutations

Emulated non-self

<500 ms
<500 ms
<500 ms
<500 ms
>500 ms
<500 ms
<500 ms
The Remote Metamorphic Engine

Artificial Immunity | Detecting the non-self

<table>
<thead>
<tr>
<th>Mutations</th>
<th>Response Time</th>
<th>Time Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>521 ms</td>
<td>&gt;200 ms</td>
</tr>
<tr>
<td>2nd</td>
<td>608 ms</td>
<td>&gt;200 ms</td>
</tr>
<tr>
<td>3rd</td>
<td>492 ms</td>
<td>&gt;200 ms</td>
</tr>
<tr>
<td>4th</td>
<td>567 ms</td>
<td>&gt;200 ms</td>
</tr>
<tr>
<td>5th</td>
<td>65 ms</td>
<td>&lt;200 ms</td>
</tr>
<tr>
<td>6th</td>
<td>622 ms</td>
<td>&gt;200 ms</td>
</tr>
<tr>
<td>7th</td>
<td>545 ms</td>
<td>&gt;200 ms</td>
</tr>
</tbody>
</table>

Emulated | non-self
Evading AI Machine Learning

Mixing Morphed Blocks

Disabling the AI from differentiating functions before, during and after execution
The Remote Metamorphic Engine

Anti-Emulation

- In memory code integrity check
- Execution environment integrity check
- In memory APIs code integrity check
- Detect hooks
- Clock synchronization
- Detect debuggers
- Detect Virtual Machines
- Collect Machine IDs
Conclusion
Questions?

remote.metamorphic.engine@gmail.com