Cryptographic Function Identification in Obfuscated Binary Programs

REcon 2012 Joan Calvet – j04n.calvet@gmail.com





Presentation Outline

Introduction to the Problem

Proposed Solution

Examples

What's Next ?

INTRODUCTION TO THE PROBLEM

What's this ?

884818 884818	23 sub esp, 1 26 mov [ebp+v 20 mov eax, [30 mov ecx, [32 mov [ebp+v 35 mov edx, [38 mov eax, [38 mov eax, [38 mov [ebp+v 35 mov [ebp+v 38 mov [ebp+v 45 mov [ebp+v 48 sh1 ecx, 5	4h ar_8], 20h ebp+v] eax] ar_4], ecx ebp+v] edx+4] ar_C], eax ar_14], 9E3779B9h ebp+var_14]
	□□ → □ □ 0840104E 0840104E 0840104E 08401051 08401051 08401057 08401057 08401057 08401057 08401057 08401057	0104E: edx, [ebp+var_8] eax, [ebp+var_8] eax, 1 [ebp+var_8], eax edx, edx short loc_4010BC
00401061 shl 00401064 nov 00401067 add 0040106A nov 0040106D add 00401070 xor 00401072 nov 00401075 shr 00401075 shr 00401078 nov 0040107E xor 0040107E xor 0040107E xor	ecx, [ebp+var_4] ecx, 4 edx, [ebp+arg_4] ecx, [edx+8] eax, [ebp+var_4] eax, [ebp+var_10] ecx, eax edx, [ebp+var_4] edx, 5 eax, [ebp+var_4] edx, [eax+0Ch] ecx, edx edx, [ebp+var_C] edx, ecx	Image: performance partial state 084010BC 084010BC nov eax, [ebp+v] 084010BF nov eax, [ebp+v] 084010BF nov eax, [ebp+var_4] 084010C2 nov [eax], ecx 084010C4 nov edx, [ebp+v] 084010C7 nov eax, [ebp+v] 084010C7 nov eax, [ebp+var_C] 084010CA nov [edx+4], eax 084010CD nov esp, ebp 084010CF pop ebp 084010D6 retn 084010D6 retn 084010D0 sub_401020 endp 084010D0 sub_401020 endp
00401088 nov 00401088 shl 0040108E nov 00401091 add 00401093 nov 00401096 add 00401096 add 0040109E shr 0040109E shr 00401081 shr 00401084 add 00401087 xor 00401087 xor 00401087 sub	<pre>[ebp+var_C], edx eax, [ebp+var_C] eax, 4 ecx, [ebp+arg_4] eax, [ecx] edx, [ebp+var_C] edx, [ebp+var_10] eax, edx ecx, [ebp+var_C] ecx, 5 edx, [ebp+arg_4] ecx, [edx+4] eax, ecx ecx, [ebp+var_4], ecx</pre>	
004010B1 nov 004010B4 sub 004010B7 nov	edx, [ebp+var_10] edx, [ebp+var_14] [ebp+var_10], edx short loc_40104E	

What's this ?

00401023 sub esp, 14h 00401026 mov [ebp+var_8], 20h 00401020 mov eax, [ebp+v] 00401030 mov ecx, [eax] 00401032 mov [ebp+var_4], ecx 00401035 mov edx, [ebp+v] 00401035 mov edx, [ebp+v] 00401035 mov edx, [ebp+v] 00401038 mov edx, [edx+4] 00401038 mov [ebp+var_C], eax 00401038 mov [ebp+var_14], 9E3779B9h 00401045 mov ecx, [ebp+var_14] 00401048 sh1 ecx, 5 00401048 mov [ebp+var_18], ecx	779B9h
<pre>w == c</pre>	5

Tiny Encryption Algorithm - Wikipedia, the free encyclopedia en.wikipedia.org/wiki/Tiny_Encryption_Algorithm - Traduire cette page The magic constant, 2654435769 or **9E3779B9**16 is chosen to be 232/φ, where φ is the golden ratio. TEA has a few weaknesses. Most notably, it suffers from ...

Properties - Versions - Reference code - See also Vous avez consulté cette page 15 fois. Dernière visite : 07/04/12

Internet Security: Cryptographic Principles, Algorithms and Protocols - Résultats Google Recherche de Livres books.google.fr/books?isbn=0470852852...

Man Young Rhee - 2003 - Computers - 405 pages ... b7e!5163 + 9e3779b9 = 5618cblc 5[2] = 5[1] + Q32 = 5618cblc + 9e3779b9 = f45044d5 5[3] = S[2] + Q32 = f45044d5 + 9e3779b9 = 9287be8e S[25] = S[24] + ...

Changeset 329 – CrypTool 2.0 🔾

https://www.cryptool.org/trac/CrypTool2/changeset/329

28 May 2009 – The magic constant, 2654435769 (Decimal) or **9E3779B9** (Hex) is chosen to be (2^32 / phi) where phi is the golden ratio.</Run> ...

TEA Encryption Algorithm, Source code O

www.shokhirev.com/nikolai/.../uTeaSet_pas.html - Traduire cette page

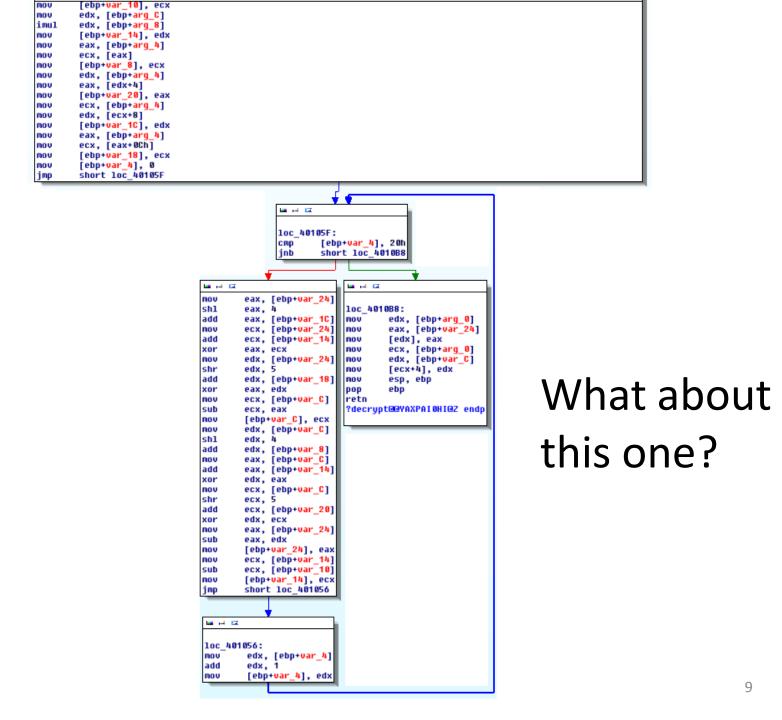
... XTeaEncrypt/XTeaDecrypt Thanks to Pedro Gimeno Fortea <parigalo@formauri.es>

} interface const Delta: longword = **\$9e3779b9**; type TLong2 = array[0.

Tools	Answer		
Crypto Searcher	"TEA"		
Draca v0.5.7b	"TEA/RC5/RC6"		
Findcrypt v2	Ø		
Hash & Crypto Detector v1.4	"TEA/XTEA/TEAN"		
PEiD KANAL v2.92	"TEA/N, RC5, RC6"		
Kerckhoffs	Ø		
Signsrch 0.1.7	"TEA"		
SnD Crypto Scanner v0.5b	Ø		

Tools	Answer		
Crypto Searcher	"TEA"		
Draca v0.5.7b	"TEA/RC5/RC6"		
Findcrypt v2	Ø		
Hash & Crypto Detector v1.4	"TEA/XTEA/TEAN"		
PEiD KANAL v2.92	"TEA/N, RC5, RC6"		
Kerckhoffs	Ø		
Signsrch 0.1.7	"TEA"		
SnD Crypto Scanner v0.5b	Ø		

That's indeed the Tiny Encryption Algorithm!



	<pre>nov [ebp+var_10], ecx nov edx, [ebp+arg_C] imul edx, [ebp+arg_C] nov [ebp+var_14], edx nov ecx, [eax] nov ecx, [eax] nov [ebp+var_8], ecx nov edx, [ebp+arg_4] nov eax, [edx+4] nov ecx, [ebp+arg_4] nov ecx, [ebp+arg_4] nov edx, [ecx+8] nov [ebp+var_1C], edx nov ecx, [ebp+arg_4] nov ecx, [ebp+arg_4] nov ecx, [ebp+arg_4] nov ecx, [ebp+arg_4] nov ecx, [ebp+arg_4], ecx nov [ebp+var_18], ecx nov [ebp+var_4], 0 jmp short loc_40105F</pre>		u 14 IX		
nov nov imul nov nov nov nov nov nov nov nov nov nov	<pre>[ebp+var_10], ecx edx, [ebp+arg_C] edx, [ebp+arg_8] [ebp+var_14], edx eax, [ebp+arg_4] ecx, [eax] [ebp+var_8], ecx edx, [ebp+arg_4] eax, [edx+4] [ebp+var_20], eax ecx, [ebp+arg_4] edx, [ecx+8] [ebp+var_1C], edx eax, [ebp+arg_4] ecx, [eax+0Ch] [ebp+var_18], ecx [ebp+var_4], 0 short loc_40105F</pre>	NOV eax, [e sh1 eax, 4 add eax, [e add ecx, [e add ecx, [e add ecx, [e xor eax, ec nov edx, [e shr edx, 5 add edx, [e xor eax, ed nov ecx, [e sub ecx, eax nov [ebp+ua nov eax, [e sh1 edx, 4 add eax, [e xor edx, 6 sh1 edx, 4 add eax, [e xor edx, 6 sh1 edx, 4 add eax, [e xor edx, 6 sh1 edx, 4 add eax, [e xor edx, 6 xor	jnb short 10 2007-00-00-00-00-00-00-00-00-00-00-00-00-	What this o	about
N	lo particular	jmp short 1			
	constants	nov edx, [e add edx, 1	ebp+var_4] ar_4], edx		10

Tools	Answer
Crypto Searcher	Ø
Draca v0.5.7b	Ø
Findcrypt v2	Ø
Hash & Crypto Detector v1.4	Ø
PEID KANAL v2.92	Ø
Kerckhoffs	Ø
Signsrch 0.1.7	Ø
SnD Crypto Scanner v0.5b	Ø

Tools	Answer
Crypto Searcher	Ø
Draca v0.5.7b	Ø
Findcrypt v2	Ø
Hash & Crypto Detector v1.4	Ø
PEID KANAL v2.92	Ø
Kerckhoffs	Ø
Signsrch 0.1.7	Ø
SnD Crypto Scanner v0.5b	Ø

Sigh.. That was still TEA!

What Can We Do ?

• How to recognize different TEA implementations in a more reliable way ?

 Is there something such implementations have to share ?

(If so, we could use it in obfuscated programs!)

Input-Output Relationship

For a key K and an encrypted text C, any TEA implementation produces the same decrypted text C'.

Input-Output Relationship

For a key K and an encrypted text C, any TEA implementation produces the same decrypted text C'.

Could we identify TEA implementations by using their deterministic I/O relationship ? *(or any other cipher)*

PROPOSED SOLUTION

How To Use Input-Output Relationship?

Let's say P is a program implementing an unknown cryptographic algorithm.

How To Use Input-Output Relationship?

- Let's say P is a program implementing an unknown cryptographic algorithm.
- First idea: execute P on *all* possible input states and check if the outputs are the same than a known cryptographic algorithm.

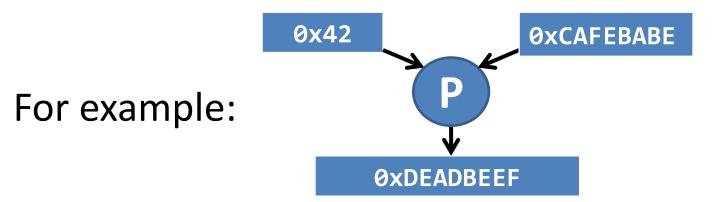
(not realistic!)

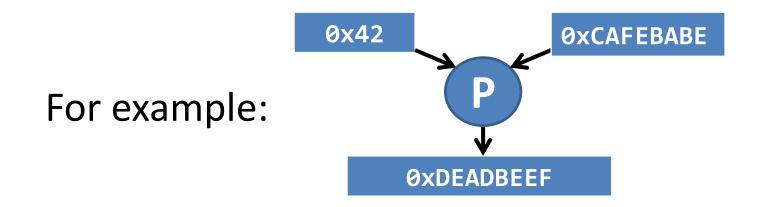
How To Use Input-Output Relationship?

- Let's say P is a program implementing an unknown cryptographic algorithm.
- First idea: execute P on all possible input states and check if the outputs are the same than a known cryptographic algorithm.

(not realistic!)

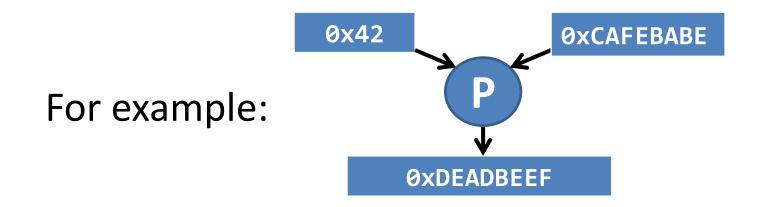
But we can observe one particular P execution and collect its input-output parameter values...





• Now imagine that when we execute a <u>reference</u> <u>implementation</u> of **TEA** with the key 0x42 and the input text 0xCAFEBABE, it produces 0xDEADBEEF.

What does it mean for **P**?



• Now imagine that when we execute a <u>reference</u> <u>implementation</u> of **TEA** with the key 0x42 and the input text 0xCAFEBABE, it produces 0xDEADBEEF.

What does it mean for **P**?

 It proves that P implements TEA <u>on these</u> <u>particular input values</u>.

Final Goal

• We are going to prove that a particular program **P** behaves like a known cryptographic algorithm <u>during a particular execution</u>.

 It means that we are **not** going to prove a general semantic equivalence between **P** and a cryptographic algorithm.

Workflow

Given a program **P**:

- Step 1: Collect P execution trace.
- <u>Step 2</u>: Extract possible cryptographic algorithms with their parameters from P execution trace (here is the magic).
- <u>Step 3:</u> Identify these algorithms by comparing their I/O relationship with those of known algorithms.

STEP 1: COLLECT EXECUTION TRACE

Execution Trace

 Pin: Dynamic Binary Instrumentation framework.

Address	Instruction	Read Registers	Written Registers	Read Memory	Written Memory
4012b3	push ebp	ebp 0012de28 esp 0012bd98	esp 0012bd94		12bd94 0012de28
4012b4	mov ebp, esp	esp 0012bd94	ebp 0012bd94		
4012b6	push ebx	ebx 02f00010 esp 0012bd84	esp 0012bd80		12bd80 2f00010

...

STEP 2: CRYPTOGRAPHIC ALGORITHM EXTRACTION

How To Find Crypto Code ? (1)

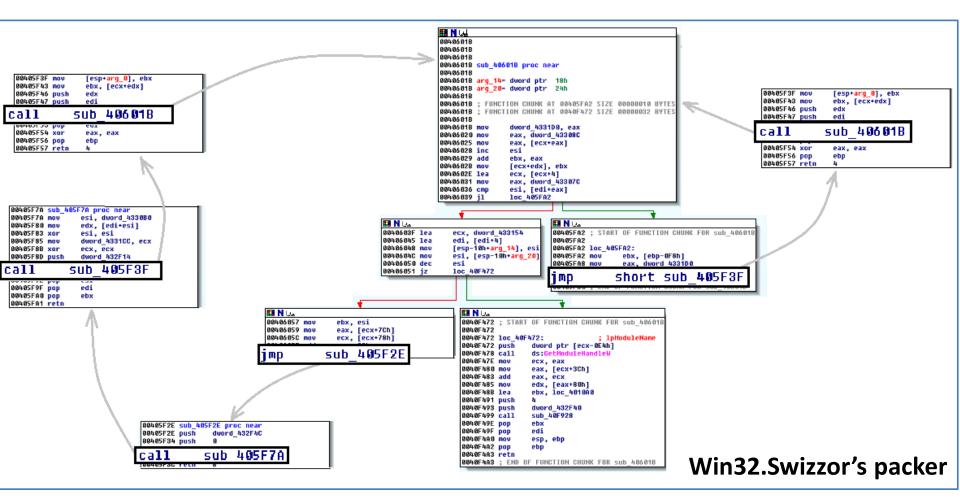
• Cryptographic code constitutes only a part of programs, we need a way to find it.

How To Find Crypto Code ? (1)

• Cryptographic code constitutes only a part of programs, we need a way to find it.

• As we want to play with *obfuscated* programs, IDA functions will not be enough...

In obfuscated programs, such things can happen:



How To Find Crypto Code ? (2)

• Cryptographic algorithms usually apply *a same treatment* on their input-output parameters.

How To Find Crypto Code ? (2)

• Cryptographic algorithms usually apply *a same treatment* on their input-output parameters.

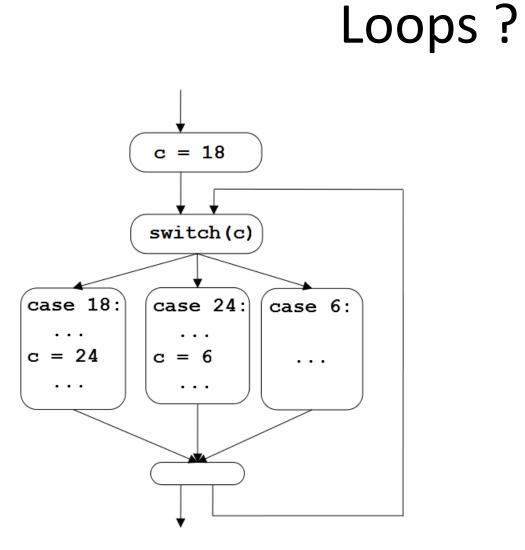
• It makes **loops** a cryptographic code feature.

How To Find Crypto Code ? (2)

• Cryptographic algorithms usually apply *a same treatment* on their input-output parameters.

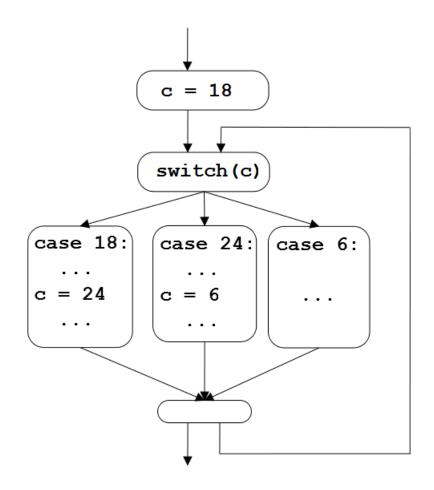
• It makes **loops** a cryptographic code feature.

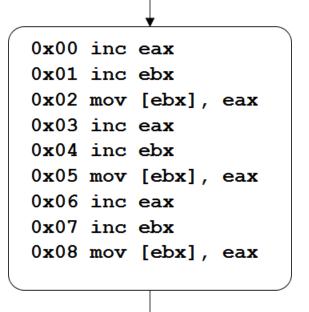
• But there are loops everywhere, not only in crypto... What kind of loops are we looking for ?



Win32.Mebroot

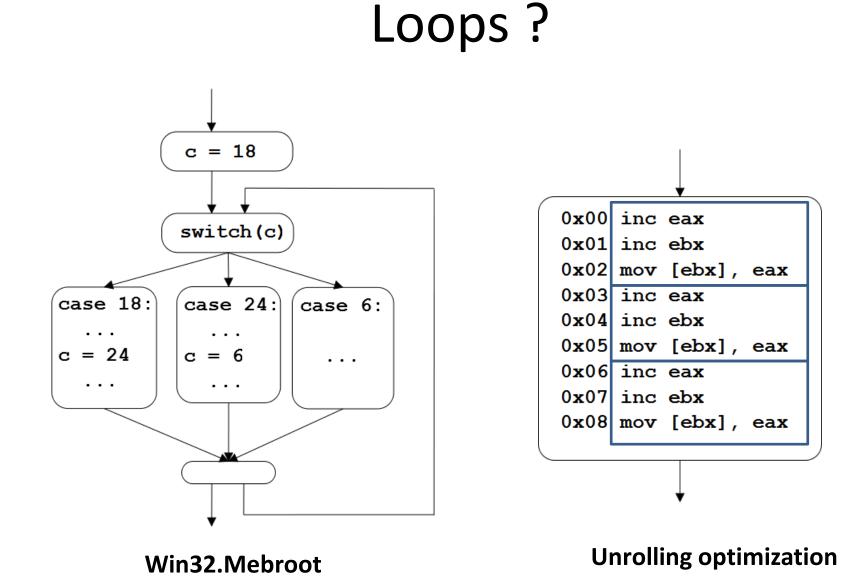






Win32.Mebroot

Unrolling optimization



Loooops

• We look for the same operations applied repeatedly on a set of data.

Loooops

 We look for the same operations applied repeatedly on a set of data.

"A loop is the repetition of a same sequence of machine instructions at least two times."

(This sequence of instructions is the loop body.)

Example

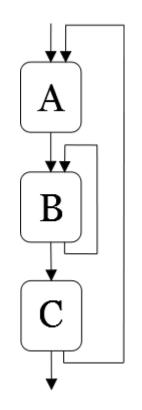
•••	•••
401325	add ebx, edi
401327	sub edx, ebx
401329	dec dword ptr [ebp+0xc]
40132c	jnz 0x401325
401325	add ebx, edi
401327	sub edx, ebx
401329	dec dword ptr [ebp+0xc]
40132c	jnz 0x401325
•••	•••

Example

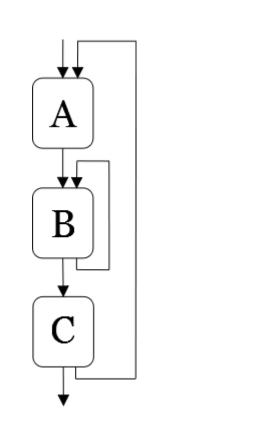
•••		
401325	add ebx, edi	
401327	sub edx, ebx	Iteration 1
401329	dec dword ptr [ebp+0xc]	
40132c	jnz 0x401325	
401325	add ebx, edi	
401327	sub edx, ebx	Iteration 2
401329	dec dword ptr [ebp+0xc]	
40132c	jnz 0x401325	
•••		•

Example

	•••		
401325	add ebx, edi		
401327	sub edx, ebx	Iteration 1	
401329	dec dword ptr [ebp+0xc]		– Loop
40132c	jnz 0x401325		
401325	add ebx, edi		2000
401327	sub edx, ebx	Iteration 2	
401329	dec dword ptr [ebp+0xc]		
40132c	jnz 0x401325	_	
•••			

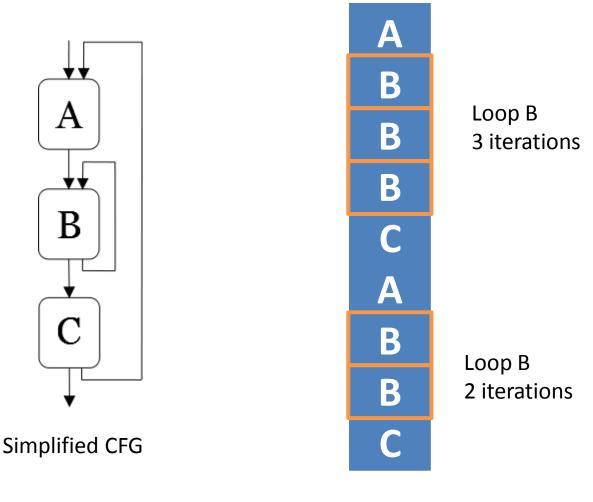


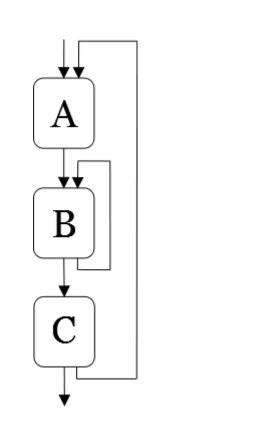
Simplified CFG



Simplified CFG

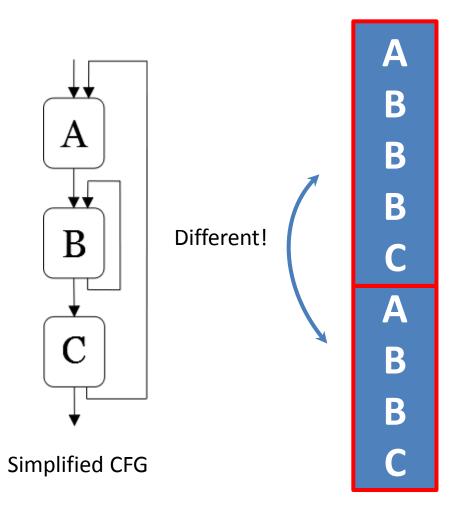


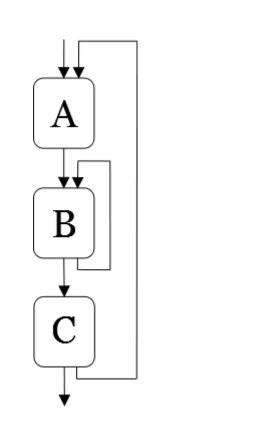




Simplified CFG

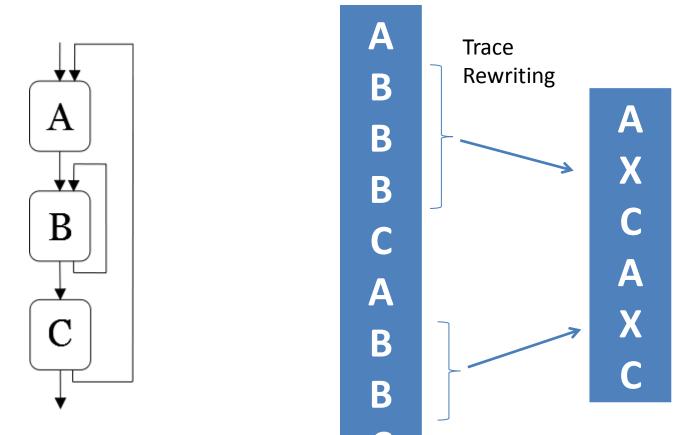




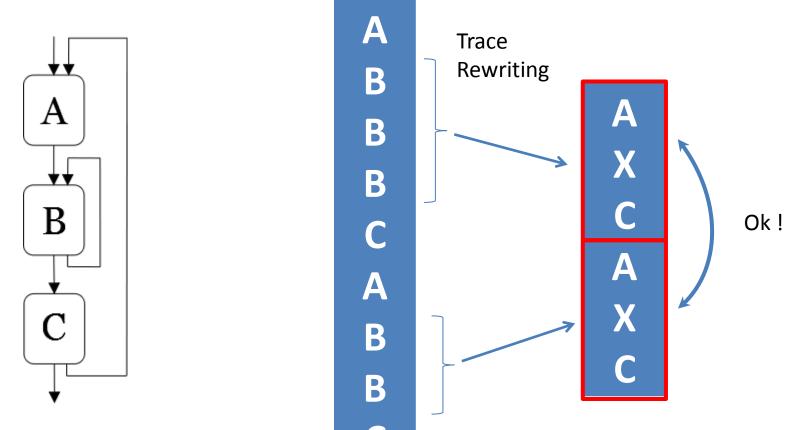


Simplified CFG





Simplified CFG



Simplified CFG

Loop Detection Algorithm

1. Detects two repetitions of a loop body in the execution trace.

(non trivial, language w.w is non-context-free)

- 2. Replaces in the trace the detected loop by a symbol representing their body.
- 3. Goes back to step 1 if new loops have been detected.

What's Next ?

• We extracted possible cryptographic code from execution traces thanks to a particular loop definition.

What's Next ?

- We extracted possible cryptographic code from execution traces thanks to a particular loop definition.
- For the moment, we assume that each possible cryptographic algorithm corresponds to *one single* loop.

What's Next ?

- We extracted possible cryptographic code from execution traces thanks to a particular loop definition.
- For the moment, we assume that each possible cryptographic algorithm corresponds to *one single* loop.
- How can we define *parameters* from the *bytes* read and written in the execution trace ?

- Distinction between input and output bytes in the execution trace:
 - Input bytes have been read without having been previously written.
 - Output bytes have been written.

 We want to group together bytes belonging to the same cryptographic parameter (key, input text...).

• We want to group together bytes belonging to the same cryptographic parameter (key, input text...).

What criteria can we use ?

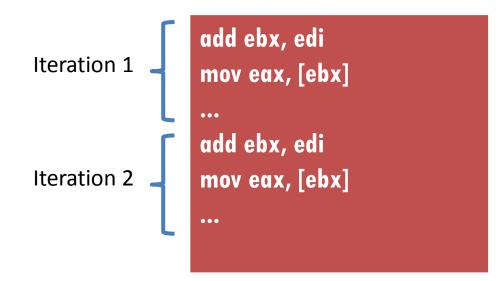
- Grouping of several bytes into the same parameter:
 - 1. If they are **adjacent in memory** (too large!)

- Grouping of several bytes into the same parameter:
 - 1. If they are **adjacent in memory** (too large!)
 - 2. And if they are manipulated by the same instruction in the loop body.

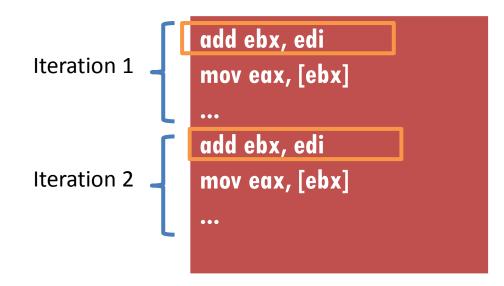
- Grouping of several bytes into the same parameter:
 - 1. If they are **adjacent in memory** (too large!)
 - 2. And if they are manipulated by the same instruction in the loop body.

add ebx, edi mov eax, [ebx]	
 add ebx, edi mov eax, [ebx]	
•••	

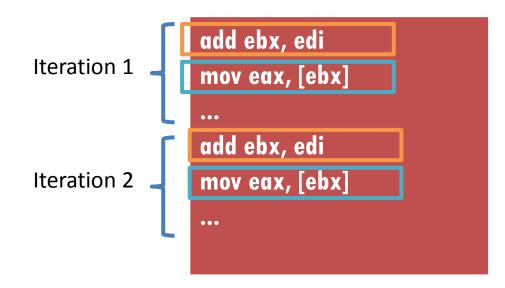
- Grouping of several bytes into the same parameter:
 - 1. If they are **adjacent in memory** (too large!)
 - 2. And if they are manipulated by the same instruction in the loop body.



- Grouping of several bytes into the same parameter:
 - 1. If they are **adjacent in memory** (too large!)
 - 2. And if they are manipulated by the same instruction in the loop body.



- Grouping of several bytes into the same parameter:
 - 1. If they are **adjacent in memory** (too large!)
 - 2. And if they are manipulated by the same instruction in the loop body.



• A parameter is then defined by:

– An identifier: "(memory address | register name):size"

– A value

Let's Recap With a Use-Case

- Tiny Encryption Algorithm:
 - Block cipher
 - 16-byte key
 - 8-byte input text
 - Magic constant *delta* (0x9E3779B9)
- We built a toy program calling the TEA decryption function on:

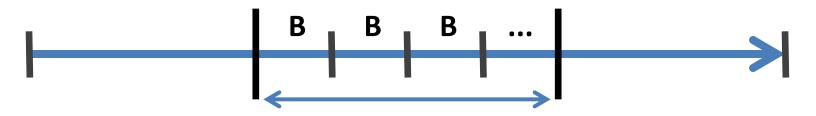
- Key : ØxDEADBEE1...DEADBEE4

– Encrypted text: **0x0123456789ABCDEF**

Step 1 : Gather Execution Trace

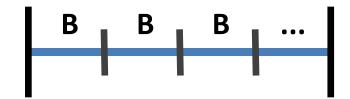


Step 2 : Recognize Loops

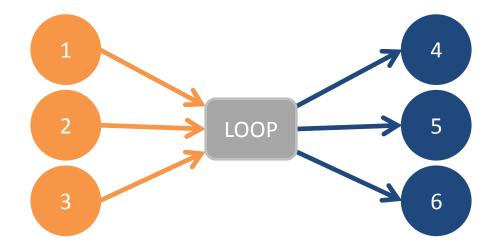


Machine instruction sequence B is repeated

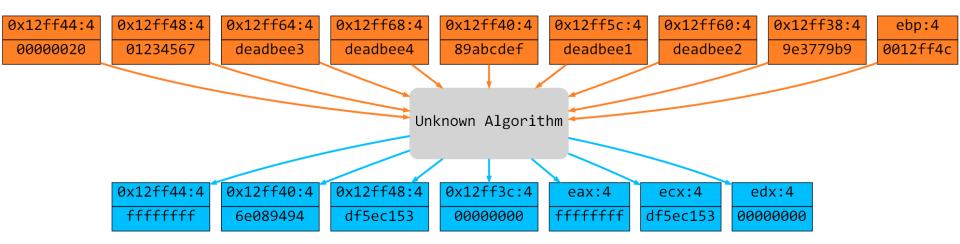
Step 2 : Recognize Loops

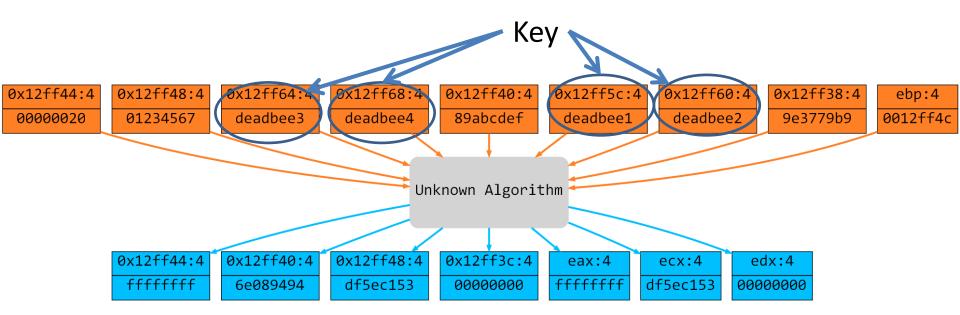


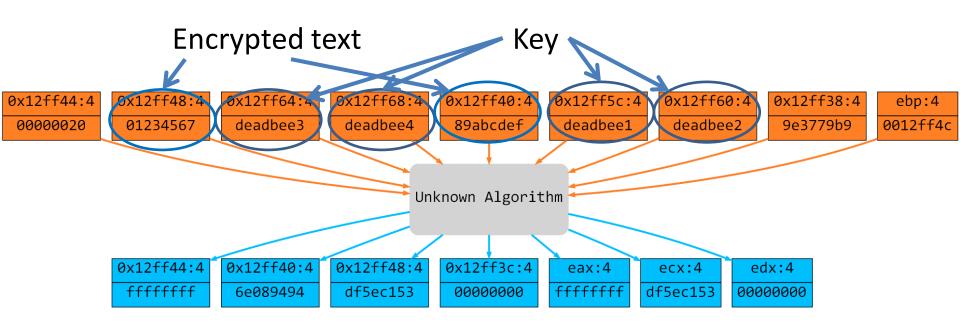
Step 3 : Define Loop Parameters

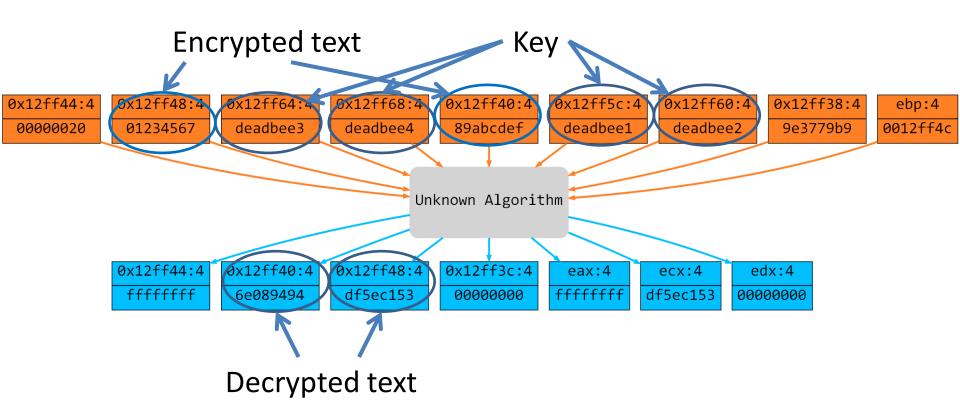


Each loop is then a possible cryptographic algorithm!



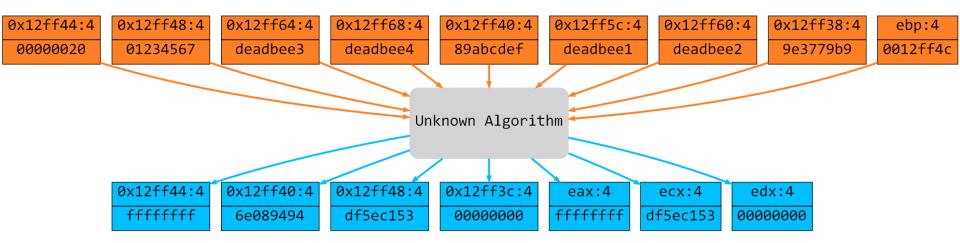




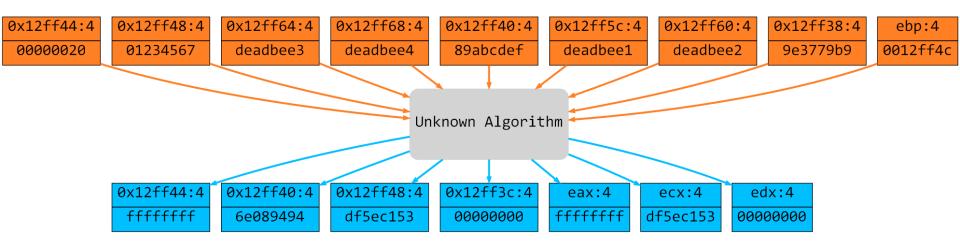


STEP 3: CRYPTO ALGORITHM IDENTIFICATION

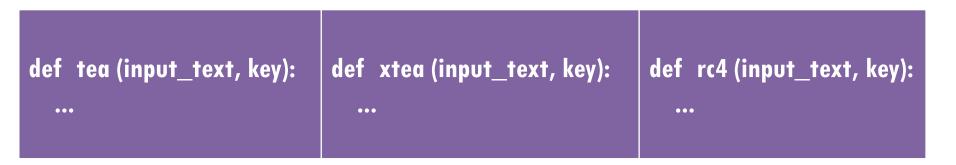
Input 1: unknown algorithm **A** with its parameter values



Input 1: unknown algorithm A with its parameter values



Input 2: reference implementations for common crypto algo



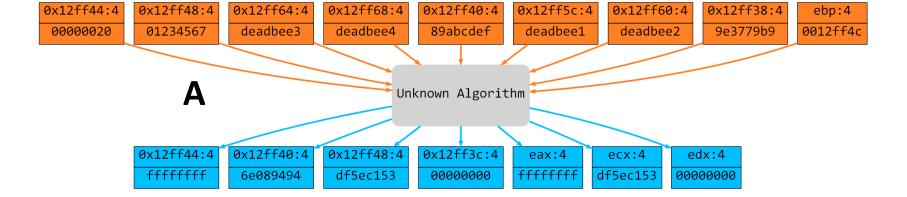
 Is there a way to combine A input values such that *tea(), xtea()* or *rc4()* would produce a combination of A output values ?

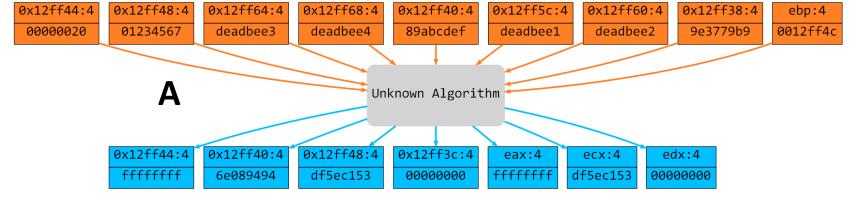
- Is there a way to combine A input values such that *tea(), xtea()* or *rc4()* would produce a combination of A output values ?
- Some difficulties:
 - Parameter division: a same cryptographic parameter can be divided into several loop parameter.

- Is there a way to combine A input values such that tea(), xtea() or rc4() would produce a combination of A output values ?
- Some difficulties:
 - Parameter division: a same cryptographic parameter can be divided into several loop parameter.
 - Parameter order: no particular order for A parameters.

- Is there a way to combine A input values such that tea(), xtea() or rc4() would produce a combination of A output values ?
- Some difficulties:
 - Parameter division: a same cryptographic parameter can be divided into several loop parameter.
 - Parameter order: no particular order for A parameters.
 - Parameter number: we collect more than the cryptographic parameters.

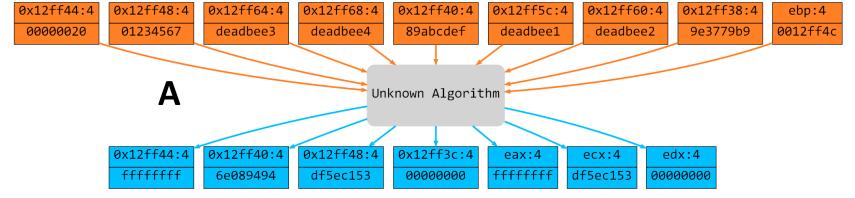
Brute-Force!





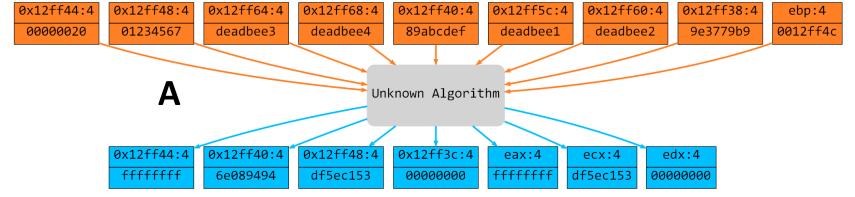
1. Generate all possible values with A input parameters:

- 1. Length 4: 00000020, 01234567, deadbee3...
- 2. Length 8: 0000002001234567, 00000020deadbee3,..
- 3. ..



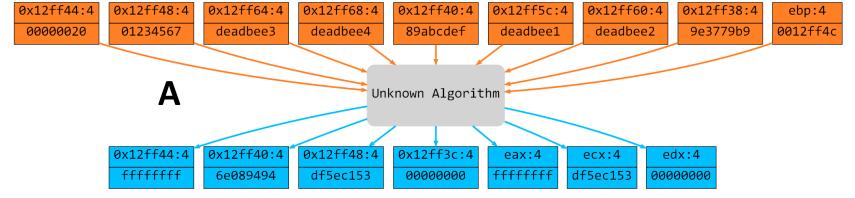
1. Generate all possible values with A input parameters:

- 1. Length 4: 00000020, 01234567, deadbee3...
- 2. Length 8: 0000002001234567, 00000020deadbee3,..
- 3. ...
- 2. Same thing with A output parameters.



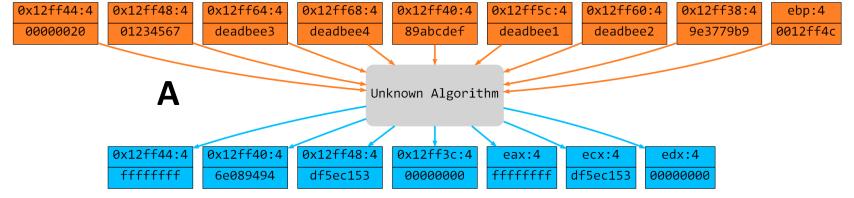
1. <u>Generate all possible values with A input parameters:</u>

- 1. Length 4: 00000020, 01234567, deadbee3...
- 2. Length 8: 0000002001234567, 00000020deadbee3,..
- 3. ...
- 2. Same thing with A output parameters.
- 3. <u>For TEA reference implementation:</u>
 - 1. Possible input texts (8 bytes): 0000002001234567,...



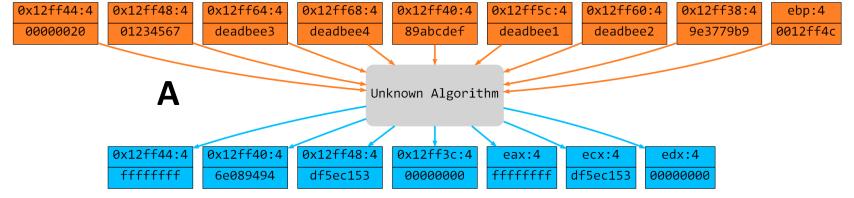
1. Generate all possible values with A input parameters:

- 1. Length 4: 00000020, 01234567, deadbee3...
- 2. Length 8: 0000002001234567, 00000020deadbee3,..
- 3. ...
- 2. Same thing with A output parameters.
- 3. <u>For TEA reference implementation:</u>
 - 1. Possible input texts (8 bytes): 0000002001234567,...
 - 2. Possible keys (16 bytes): ...



1. <u>Generate all possible values with A input parameters:</u>

- 1. Length 4: 00000020, 01234567, deadbee3...
- 2. Length 8: 0000002001234567, 00000020deadbee3,..
- 3. ...
- 2. Same thing with A output parameters.
- 3. <u>For TEA reference implementation:</u>
 - 1. Possible input texts (8 bytes): 0000002001234567,...
 - 2. Possible keys (16 bytes): ...
 - Execute our TEA reference implementation on each possible pair (input text, key)



1. <u>Generate all possible values with A input parameters:</u>

- 1. Length 4: 00000020, 01234567, deadbee3...
- 2. Length 8: 0000002001234567, 00000020deadbee3,..
- 3. ...
- 2. Same thing with A output parameters.
- 3. <u>For TEA reference implementation:</u>
 - 1. Possible input texts (8 bytes): 0000002001234567,...
 - 2. Possible keys (16 bytes): ...
 - Execute our TEA reference implementation on each possible pair (input text, key)
 - 4. If the output has been produced during step 2: success! ⁸⁷

```
STARTED AT
2012-04-08 08:59:58.284000
** Crypto Algorithm Identification starting !**
9 input parameters
7 output parameters
All possible input values generation... Done!
                                                              ~ 2 minutes
All possible output values generation... Done!
Build internal structure... Done!
Comparison phase starting... Test for TEA decryption...
** Found TEA decryption **
 ===> Key (16 bytes) : deadbee1deadbee2deadbee3deadbee4
 ===> Crypted text (8 bytes) : 0123456789abcdef
===> Decrypted text (8 bytes) : df5ec1536e089494
ENDED AT
2012-04-08 09:01:37.832000
```

Malware And TEA

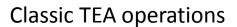
EXAMPLES!

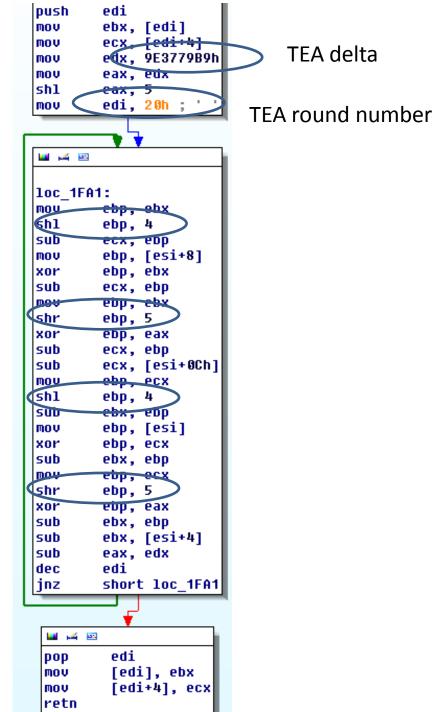
Storm Worm

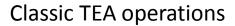
• Several internet references about the use of TEA in the Storm Worm packer (aka Tibs).

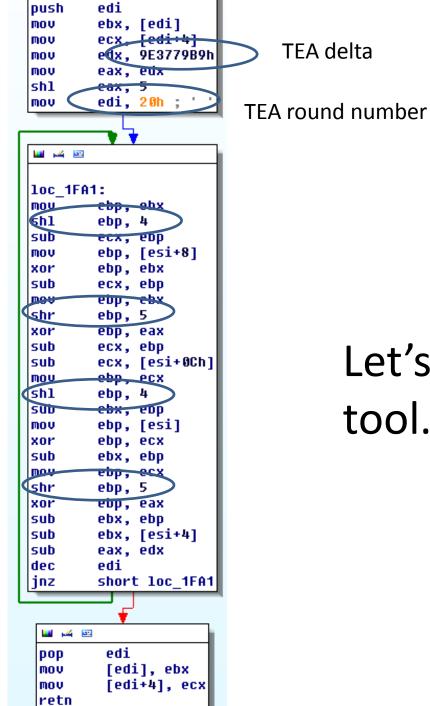
• Let's take a look to the code...

push	edi		
mov		[edi]	
mov	ecx,	[edi+4]	
mov	edx,	9E3779B9h	
mov	eax,	edx	
shl	eax,	5	
mov	edi,	20h ;	
🖬 🖂	52		
10C 1	IEA1-		
mov_	ebp,	ehx	
sh1	ebp,		
sub	ecx,	ehn	
mov	eta,	[esi+8]	
xor	ebp,	epx [corrol	
Sub	ecx,	ehn	
mnu	ebp,		
shr	ebp,		
xor	ebp,		
sub	ecx,		
sub		[esi+0Ch]	
mov	ebp,		
shl	ebp,		
sub	ebx,		
mov		[esi]	
xor	ebp,		
sub	ebx,		
mov	ebp,		
shr	ebp,		
xor	ebp,		
sub	eby,	ehn	
sub	ebx,	[esi+4]	
sub	eax,		
dec	edi	Cun	
jnz		t loc 1FA1	
1.1.2	5101	100_1111	
-			
pop	edi		
MOV			
MOV		i+4], ecx	
retr	1		

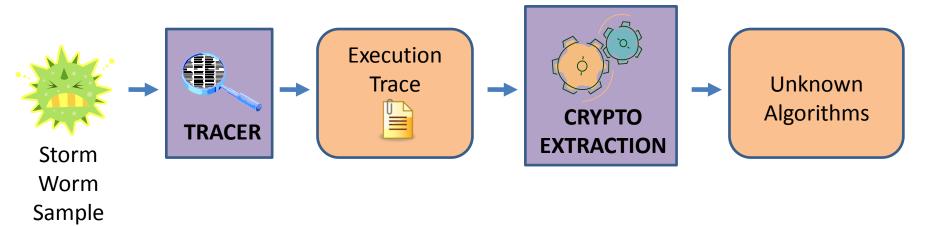




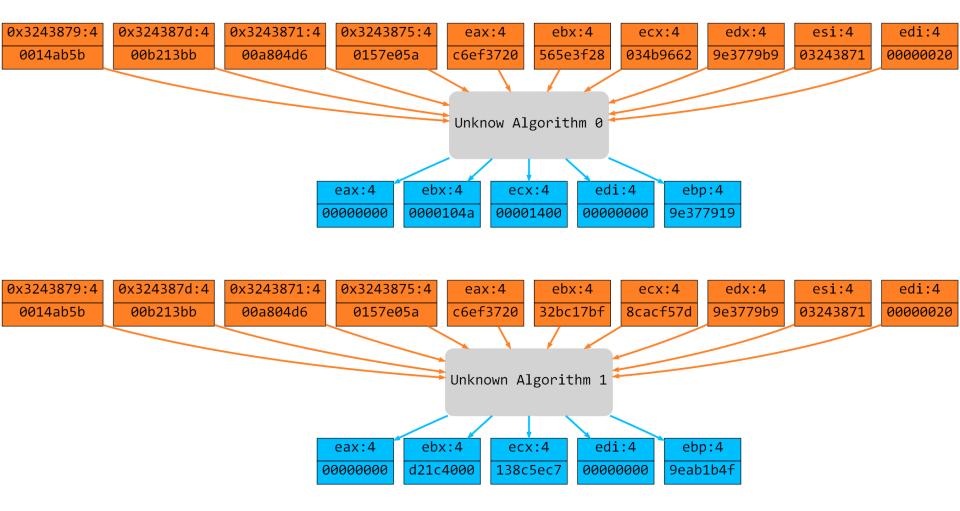




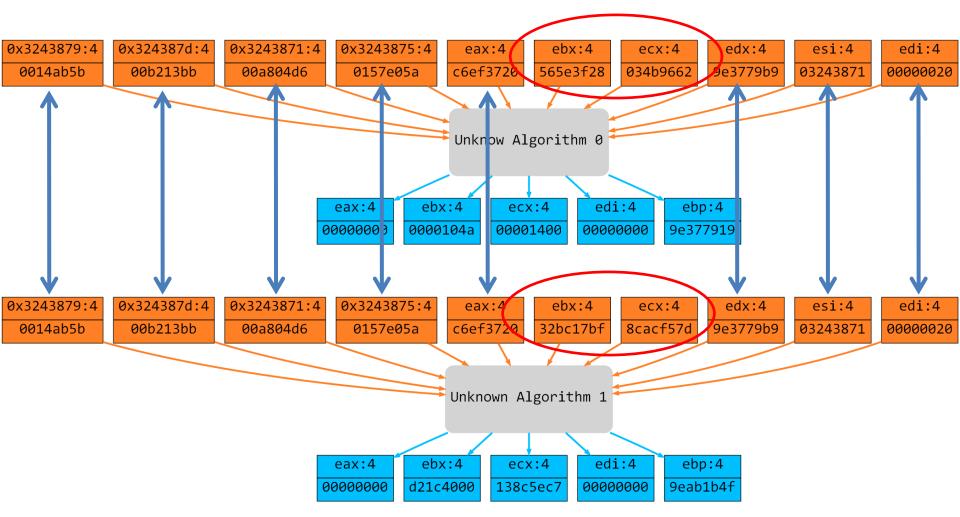
Let's try our tool...



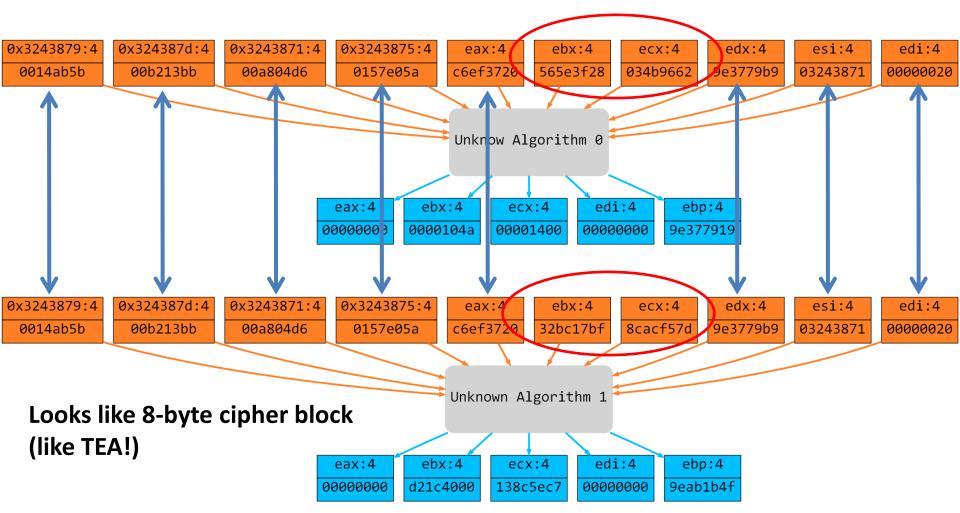
• For the previous loop, we extracted many unknown algorithms like these ones:

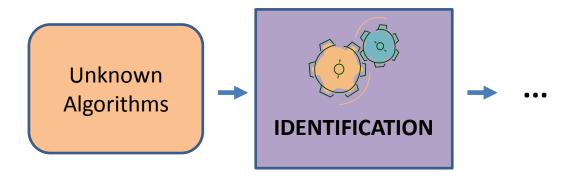


• For the previous loop, we extracted many unknown algorithms like these ones:



• For the previous loop, we extracted many unknown algorithms like these ones:





STARTED AT 2012-04-08 14:20:30.858000 ** Crypto Algorithm Identification starting !** 8 input parameters 5 output parameters All possible input values generation... Done! All possible output values generation... Done! Build internal structure Done! Comparison phase starting... Test for TEA decryption... Unknown Algorithm stays unkwown! Done! ENDED AT 2012-04-08 14:21:11.328000

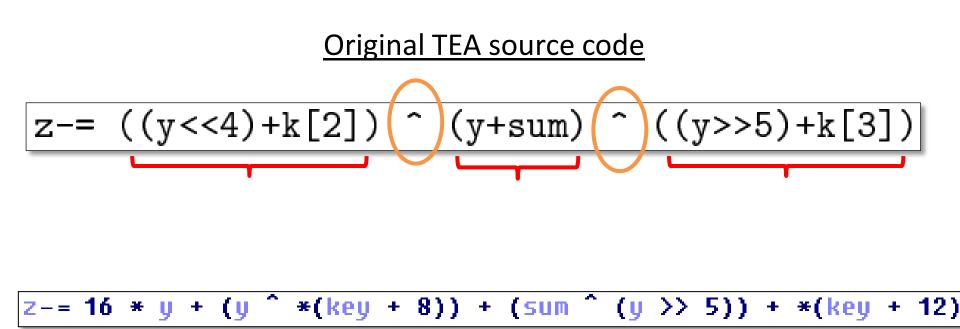
WTF?

Original TEA source code

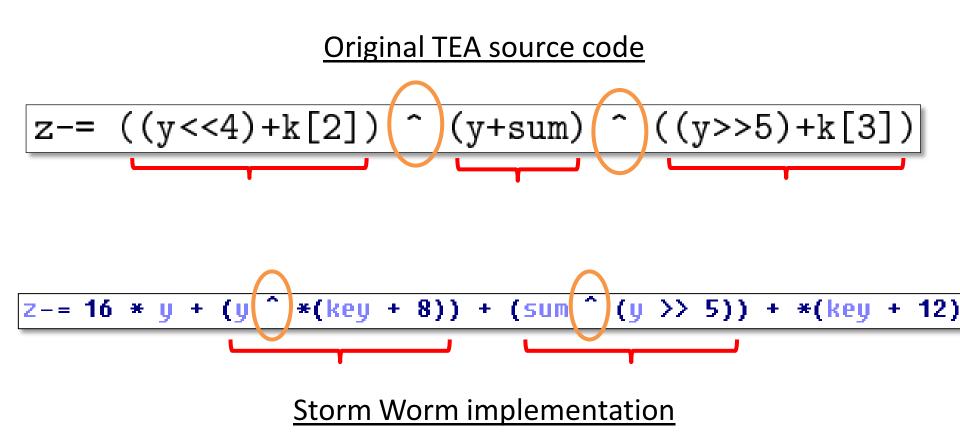
z-= ((y<<4)+k[2]) ^ (y+sum) ^ ((y>>5)+k[3])

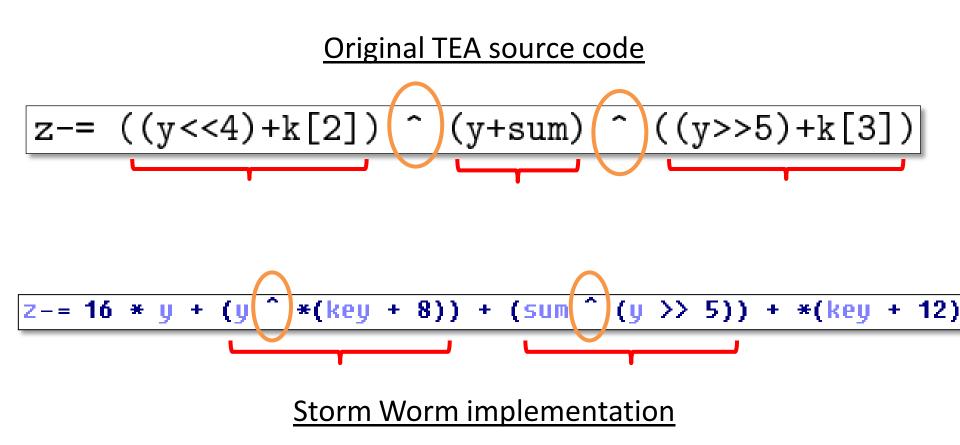
z-= 16 * y + (y ^ *(key + 8)) + (sum ^ (y >> 5)) + *(key + 12)

Storm Worm implementation



Storm Worm implementation





This is not TEA: parenthesis at the wrong place!

```
STARTED AT
2012-04-10 13:22:02.192000
** Crypto Algorithm Identification starting !**
8 input parameters
5 output parameters
All possible input values generation... Done!
All possible output values generation... Done!
Build internal structure... Done!
Comparison phase starting... Test for *Storm Worm* TEA decryption...
** Found *Storm Worm* TEA decryption **
 ===> Key (16 bytes) : 00a804d60157e05a0014ab5b00b213bb
                                                                     (\mathbf{\cdot})
 ===> Crypted text (8 bytes) : 565e3f28034b9662
 ===> Decrypted text (8 bytes) : 0000104a00001400
ENDED HI
2012-04-10 13:22:13.877000
```

Ok, Storm Worm implementation added to the base... (this is *not* TEA)

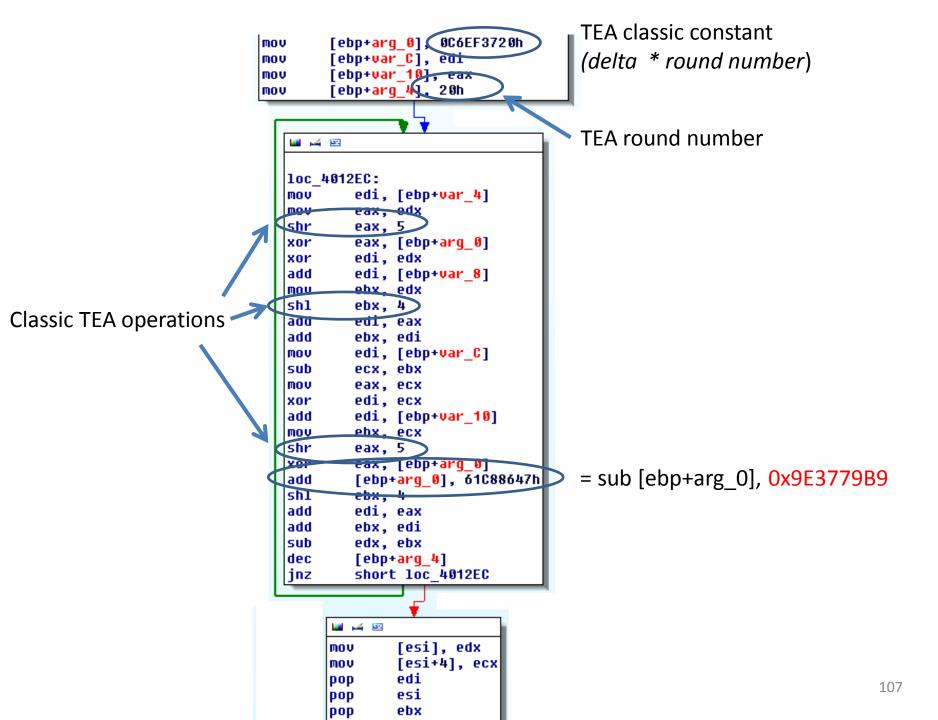
Trojan.SilentBanker

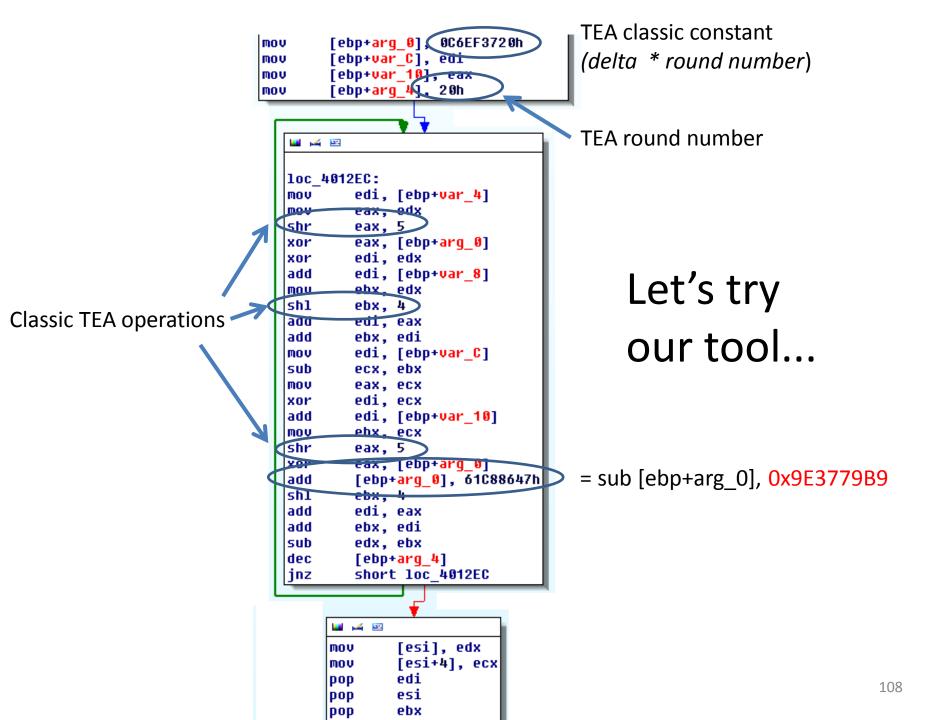
 Several internet references about the use of TEA in SilentBanker.

• Let's take a look to the code...

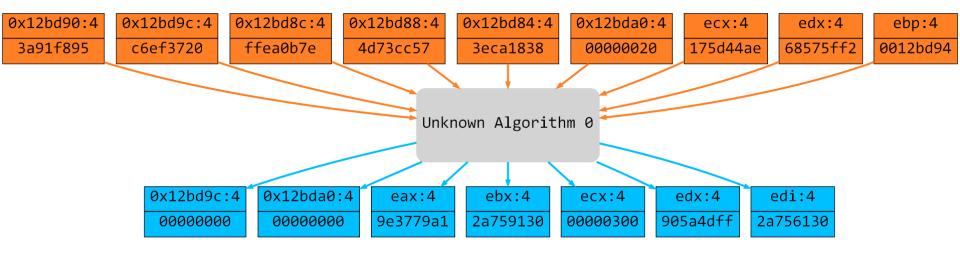
(sounds familiar, isn't it ?)

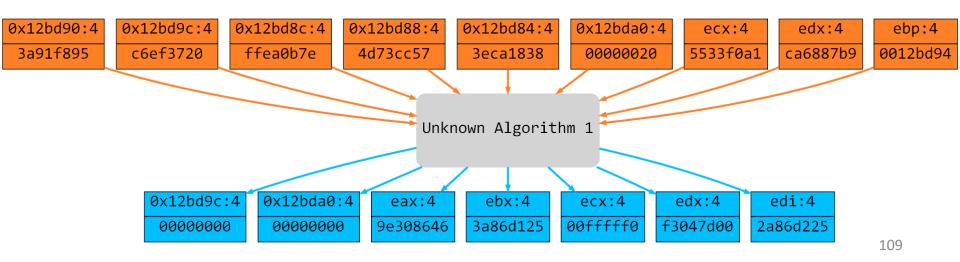
mov	[ebp+ <mark>arg_0],</mark> 0C6EF3720h
MOV	[ebp+var_C], edi
MOV	[ebp+var_10], eax
nov	[ebp+ <mark>arg_4],</mark> 20h
u ,4	
100	4012EC:
mov	
mov	eax, edx
shr	eax, 5
xor	eax, [ebp+ <mark>arg_0</mark>]
xor	edi, edx
add	
mov	
shl	
add	· · · · · · · · · · · · · · · · · · ·
add	ebx, edi
mov	
sub	ecx, ebx
mov	
xor	
add	edi, [ebp+var_10]
mov	ebx, ecx
shr	eax, 5
xor	eax, [ebp+arg_0]
add	[ebp+ <mark>arg_0</mark>], 61C88647h
shl	ebx, 4
add	edi, eax ebx, edi
sub	edx, edx
dec	[ebp+arg_4]
jnz	short loc 4012EC
1.1.2	
	mov [esi], edx
	mov [esi; eux mov [esi+4], ecx
	pop edi
	pop esi
	pop ebx



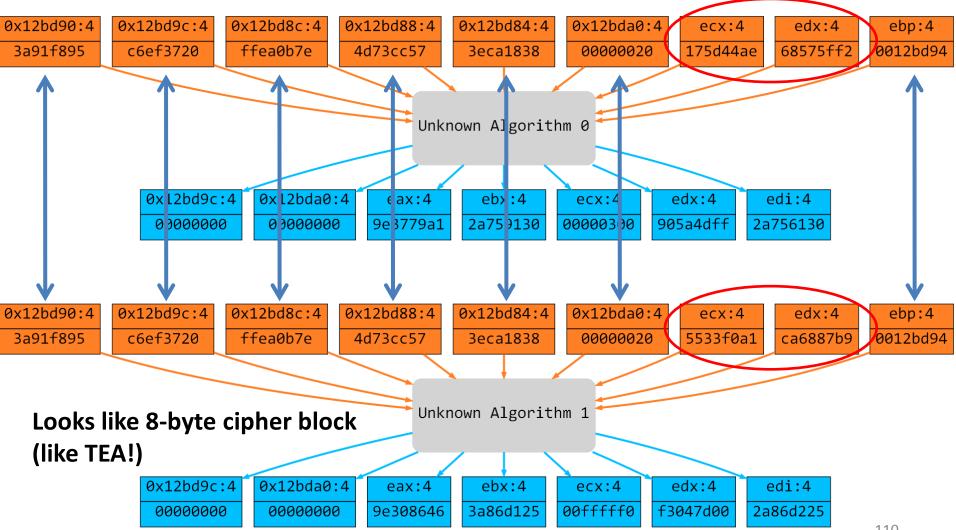


• For the previous loop, we extracted many unknown algorithms like these ones:





• For the previous loop, we extracted many unknown algorithms like these ones:



STARTED AT
2012-04-08 16:05:14.288000
** Crypto Algorithm Identification starting !**
9 input parameters
7 output parameters
All possible input values generation Done!
All possible output values generation Done!
Puild internal ctructure Deno!
Comparison phase starting Test for TEA decryption Unknown Algorithm stays
unkwown!
Done !

Fail.. Again !?

```
STARTED AT
2012-04-08 16:15:22.255000
** Crypto Algorithm Identification starting !**
9 input parameters
 output parameters
All possible input values generation... Done!
All possible output values generation... Done!
Build internal structure... Done!
Comparison phase starting Test for *Storm Worm* TFA decruption...
** Found *Storm Worm* TEA decryption **
 ===> Key (16 bytes) : 4d73cc573eca18383a91f895ffea0b7e
 ===> Crupted text (8 butes) : 68575ff2175d44ae
 ===> Decrypted text (8 bytes) : 905a4dff300
ENDED AT
2012-04-08 16:16:22.552000
```

Same implementation than in the Storm Worm!

• They probably both copied/pasted a wrong source code from the internet.

Started to look for it: Google, TEA Wikipedia page,... nothing!

• At some point, I remembered something these two malware families have in common...



They both came from Russia!



Recent changes

Contact Wikipedia

- Toolbox
- Print/export
- Languages Deutsch Español Français Hrvatski Italiano Doleki Русский

Tiny Encryption Algorithm

From Wikipedia, the free encyclopedia

In cryptography, the Tiny Encryption Algorithm (TE few lines of code. It was designed by David Wheeler Fast Software Encryption workshop in Leuven in 1994

The cipher is not subject to any patents.

Contents [hide] **1** Properties 2 Versions 3 Reference code 4 See also 5 Notes 6 References 7 External links

Properties

TEA operates on two 32-bit unsigned integers (could I a suggested 64 rounds, typically implemented in pairs exactly the same way for each cycle. Different multiple rounds. The magic constant, 2654435769 or 9E3779

TEA has a few weaknesses. Most notably, it suffers fr effective key size is only 126 bits.^[3] As a result, TEA i hacking Microsoft's Xbox game console, where the ci requires 2²³ chosen plaintexts under a related-key pai designed.

Ссылки

- Страница алгоритма шифрования ТЕА 🚱 📿
- Roger M. Needham and David J. Wheeler. «TEA, a Tiny Encryption Algorithm»(PDF)
- Andew Hand "Hacking the Xbex: an introduction to reverse engineering" [1] 2 O

• Журнал «Хакер Онлайн», ТЕА: блочный шифр своими руками, (2004)[2] 🗗 Ο 🔿

- Paris Kitsos, Yan Zhang. «RFID Security: Techniques, Protocols and System-On-Chip Design»[3] & O
- David J. Wheeler and Roger M. Needham. «Correction to xtea.» Technical report, Computer Laboratory,
- Roger M. Needham and David J. Wheeler. «Tea extensions.» Technical report, Computer Laboratory, Ur
- Test vectors for TEA 🖉 🔾

JavaScript implementation of XXTEA with Base64 & O



Журнал	Новости	Форум	Видео
Bugtrack	Статьи	Блог	Поиск



Взлом как образ мысли

Интервью с человеком, который, как оказалось, является не только талантливым пентестером в одной из крупных ИБкомпании, но и хакером-ветераном, который уверенными шагами вышел на свет и прикоснулся к истории российской хак-сцены....



Полный гі голосован

Конкурсы с гс посетителей, интерес вызы лакомые приз

ТЕА: блочный шифр своими руками

Дата: 22.04.2004 📇 🖂 🖓 | 👤 slon

В данном тексте хотелось бы затронуть такую животрепещущую тему, как шифрование файлов. Вообще нужно различать два вида шифрования файлов:

- шифрование для себя (чтобы ваши файлы никто, кроме вас не "читал")
- шифрование для других (чтобы ваши файлы "читал" только адресат)



Russian Website TEA source code

push edi ; Сохраняем mov ebx,v0 ; Кладём mov ecx,v1 ; В ecx кл mov edx,9e3779b9h ; mov eax,edx ; Кладём shl eax,5; Сдвиг eax mov edi,32; Кладём DLoopR: mov ebp,ebx ; Кладёк shl ebp,4 ; Сдвиг ebp sub ecx,ebp; Отнима mov ebp,k2 ; Кладём xor ebp,ebx ; XOR'им sub ecx,ebp; Отнима mov ebp,ebx ; Кладёг shr ebp,5; Сдвиг ebp xor ebp,eax ; XOR'им sub ecx,ebp; Отнима sub ecx,k3; Отнимае mov ebp,ecx ; Кладём shl ebp,4 ; Сдвиг ebp sub ebx,ebp; Отнима mov ebp,k0 ; Кладём xor ebp,ecx ; XOR'им sub ebx,ebp; Отнима mov ebp,ecx ; Кладём shr ebp,5 ; Сдвиг ebp xor ebp,eax ; XOR'им sub ebx,ebp; Отнима sub ebx,k1; Отнимае sub eax,edx ; Отнима dec edi ; Уменьшаем jnz DLoopR ; Дешифр

pop edi ; Вынимаем и mov v0,ebx ; Кладём mov v1,ecx ; В отведё ret ; Возврат из подп

push edi ; Сохраняем mov ebx,v0 ; Кладём mov ecx,v1; В ecx кл mov edx,9e3779b9h ; mov eax,edx ; Кладём shl eax,5; Сдвиг eax mov edi,32; Кладём DLoopR: mov ebp,ebx ; Кладёк shl ebp,4 ; Сдвиг ebp sub ecx,ebp; Отнима mov ebp,k2 ; Кладём xor ebp,ebx ; XOR'им sub ecx,ebp; Отнима mov ebp,ebx ; Кладёг shr ebp,5; Сдвиг ebp xor ebp,eax ; XOR'им sub ecx,ebp; Отнима sub ecx,k3; Отнимае mov ebp,ecx ; Кладём shl ebp,4 ; Сдвиг ebp sub ebx,ebp; Отнима mov ebp,k0 ; Кладём Website xor ebp,ecx ; XOR'им sub ebx,ebp ; Отнима **TEA** source mov ebp,ecx ; Кладём shr ebp,5 ; Сдвиг ebp xor ebp,eax ; XOR'им sub ebx,ebp; Отнима sub ebx,k1; Отнимае sub eax,edx ; Отнима dec edi ; Уменьшаем jnz DLoopR ; Дешифр рор edi ; Вынимаем и mov v0,ebx ; Кладём mov v1,ecx ; В отведе ret ; Возврат из подп

Russian

code

push	edi				
mov	ebx, [edi]				
mov	ecx, [edi+4]				
mov					
mov					
shl					
mov	edi, 20h ; ' '				
111 14 8	2				
10c_1F	A1:				
mov	ebp, ebx				
shl	ebp, 4				
sub	ecx, ebp				
mov	ebp, [esi+8]				
xor	ebp, ebx				
sub	ecx, ebp				
mov	ebp, ebx				
shr	ebp, 5				
xor	ebp, eax				
sub	ecx, ebp				
sub	ecx, [esi+OCh]				
mov	ebp, ecx				
shl	ebp, 4				
sub	ebx, ebp				
mov	ebp, [esi]				
xor	ebp, ecx				
sub	ebx, ebp				
shr	ebp, ecx ebp, 5				
xor	ebp, eax				
Sub	ebp, eax ebx, ebp				
sub	ebx, [esi+4]				
sub	eax, edx				
dec	edi				
jnz	short loc 1FA1				
🖬 🖂	E				
рор	edi				
mov	[edi], ebx				
mov [edi+4], ecx					
retn	-				

Storm Worm

MORE EXAMPLES!

RC4 (1)

- RC4 algorithm:
 - Stream cipher
 - Variable-length key
 - Two loops generate a pseudorandom stream into a <u>256 bytes</u> substitution-box (S-BOX).
 - A final loop does the actual decryption.
- We have to extend our model to regroup different loops into a same algorithm.

Interlude: Loop Data Flow

- Two loops *L1* and *L2* are in the same algorithm:
 - If *L1* started before *L2* in the trace.
 - If L2 uses as input parameter an output parameter of L1.

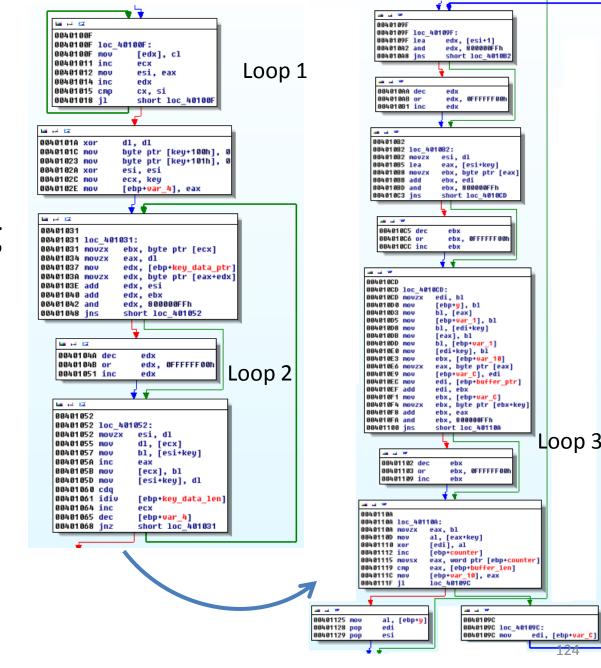
(or the contrary!)

RC4 (2)

- We built a toy program calling the RC4 decryption function on:
 - Key : "SuperKeyIsASuperKey" (19 bytes)

- Encrypted text: "AAA....AA" (1024 bytes)

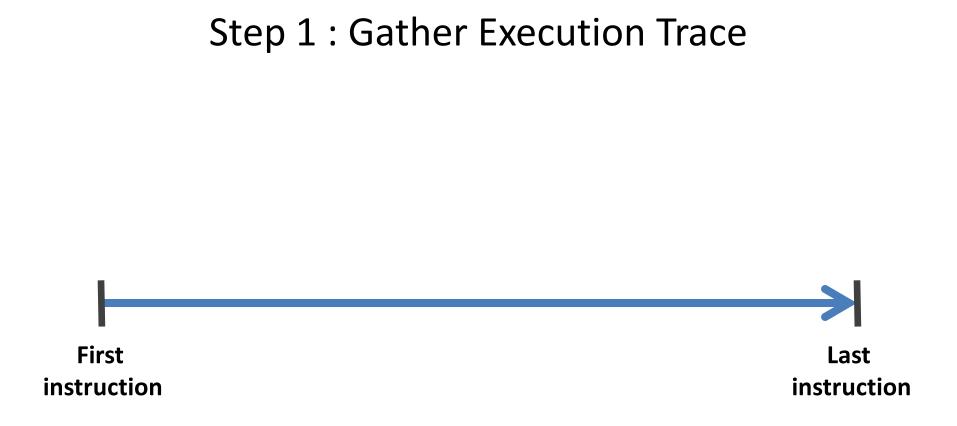
Statically speaking it looks like this...



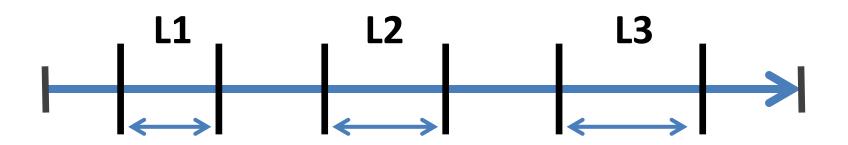
Tools	Answer
Crypto Searcher	Ø
Draca v0.5.7b	Ø
Findcrypt v2	Ø
Hash & Crypto Detector v1.4	Ø
PEID KANAL v2.92	Ø
Kerckhoffs	Ø
Signsrch 0.1.7	Ø
SnD Crypto Scanner v0.5b	Ø

Tools	Answer
Crypto Searcher	Ø
Draca v0.5.7b	Ø
Findcrypt v2	Ø
Hash & Crypto Detector v1.4	Ø
PEiD KANAL v2.92	Ø
Kerckhoffs	Ø
Signsrch 0.1.7	Ø
SnD Crypto Scanner v0.5b	Ø

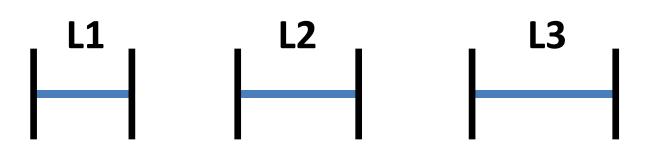
Let's try our tool...



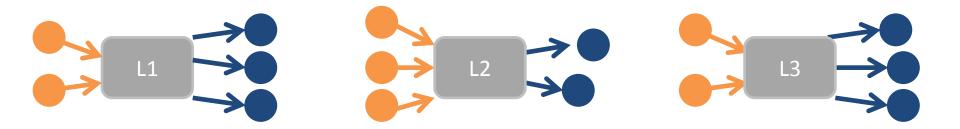
Step 2 : Recognize Loops



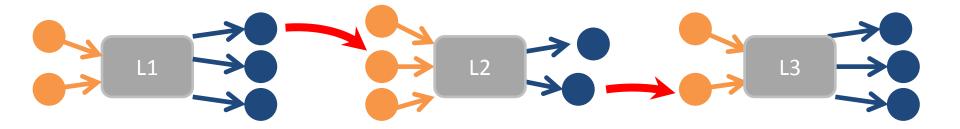
Step 2 : Recognize Loops



Step 3 : Define Loop Parameters



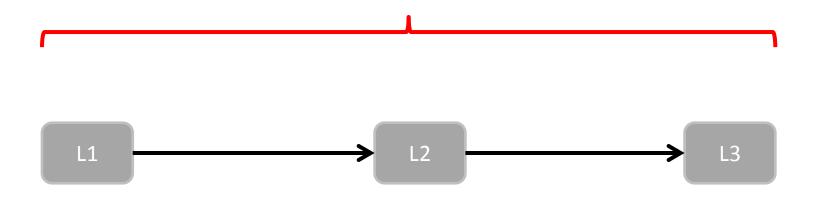
Step 4 : Connect Loops With Data-Flow

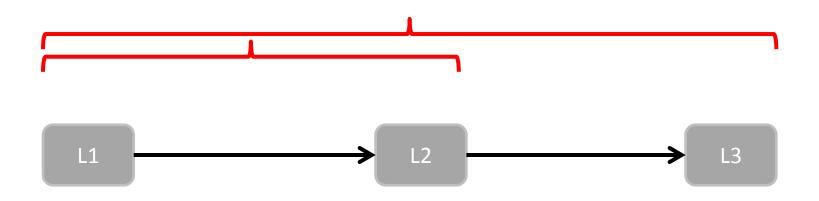


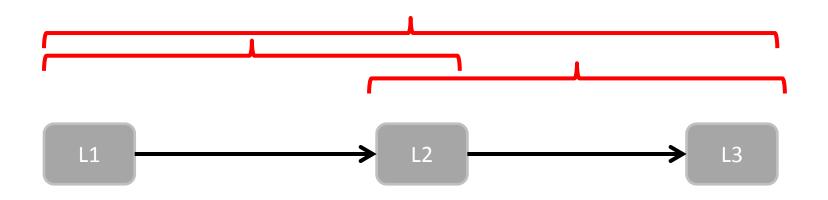
Loop Data Flow Graph (oriented, acyclic)

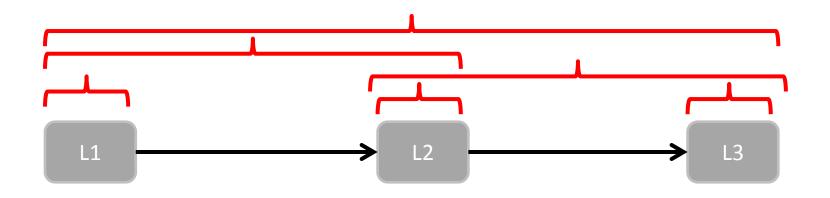


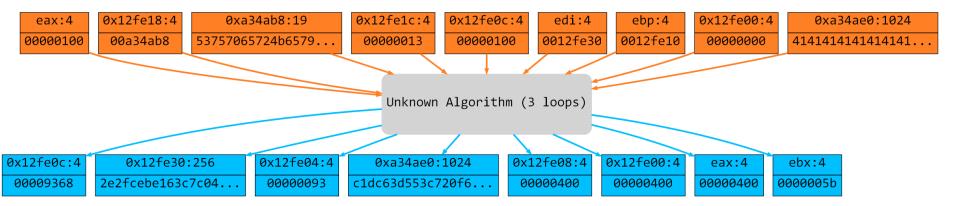


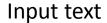


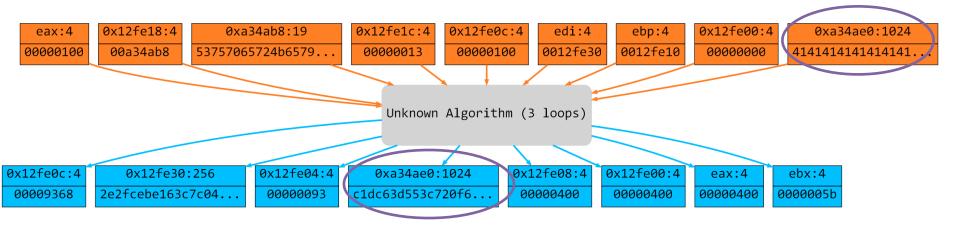




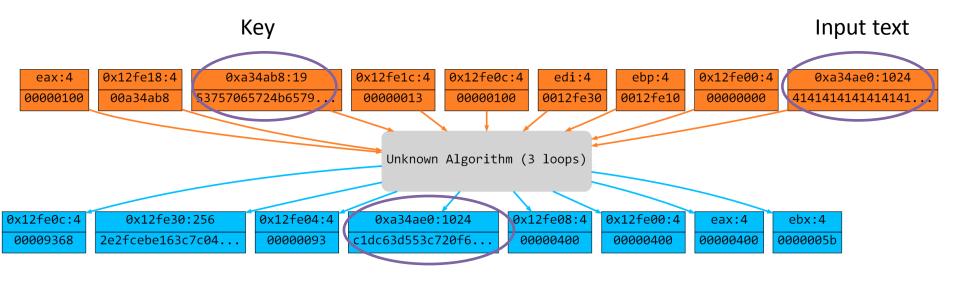




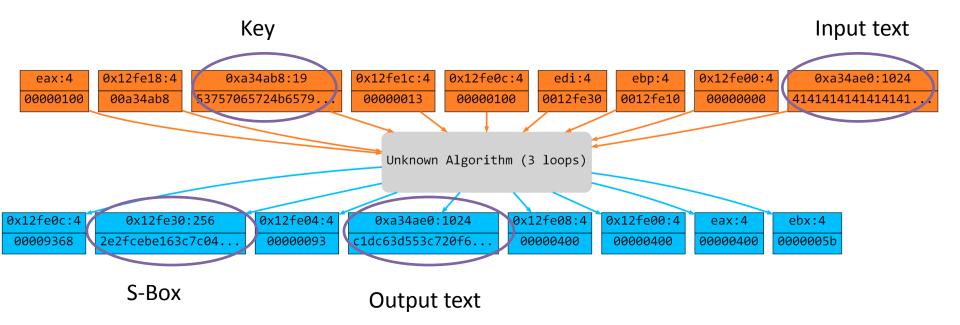




Output text



Output text



```
STARTED AT
2012-04-08 19:34:12.959000
** Crypto Algorithm Identification starting !**
3 input parameters
3 output parameters
All possible input values generation... Done!
All possible output values generation... Done!
Build internal structure... Done!
Comparison phase starting... Test for RC4...
** Found RC4 **
 ==> Key (19 bytes) : 53757065724b657949734153757065724b6579
 ===> Crypted text (1024 bytes) : 414141414141414141...
 ===> Decrypted text (1024 bytes) : c1dc63d553c720f6...
ENDED HI
2012-04-08 19:34:12.990000
```

Win32.Sality.AA

• Several internet references about the use of RC4 in Sality.AA protection layers.

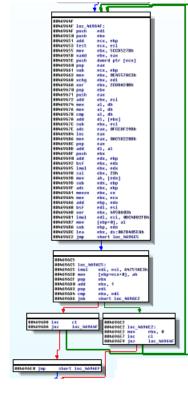
• Let's take a look...

(suspense...)



READYSTM REA





Loop 2

Loop 3

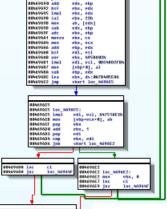
	· · · · · · · · · · · · · · · · · · ·
88469722	
@8469722 loc_45	
02469722 inc 02469725 push	d1 18788255h
88469729 max	[esp-h-arg_0], esi
8846972C pash	edi
88469720 add	cop, edx
0846972F neg	at
88469731 max	al, dh
88469733 mov 88469735 add	al, 808h
88469738 tub	bl, [shp+0] shp, sdx
BBA69738 xadd	eax, esi
88469730 mov	esi, 1960F3Ch
88469743 mov	al, 580
08469746 push	
88469748 pap	eas
BBA69749 add	obp, edx
esado748 test esado751 shid	ecz, 8070485565 esi, ecz, cl
ggad975% test	eas, ecs
08469756 mov	al, [ebp+#]
08469759 sab	cop, edx
08469758 SDb	edi, edx
08469750 shrd	edi, esi, BAZh
88469761 xchg	edi, esi
88469763 add 88469765 may	ebs, ebp
08469768 inc	edi, BOFAC78C2h esi
8846976C rep no	
0846976F 540	cox, cop
88469771 mov	edi, BF79405FAD
88469777 bswp	esi
88469779 max	edi, 80FACSD22h
8846977F add	etz, ebp edi, 38b
BBA69781 PCP BBA69785 BOUSS	edi, 38h edi, dx
88469787 bswap	eti
88469789 mov	[ebx], al
08469788 Stb	cox, cop
88469780 mov	edi, edx
8846978F bt	edi, esi
88469792 dec	edi
BBA69793 add BBA69795 bswap	edz, ebp
88469797 db	#F3h
88469797 rep no	
08469798 Sub	eds, chp
08469790 imel	edi, esi
88469788 max	edi, 572935060
BBA69786 bsF	esi, ecz
BBA60789 add BBA60788 pap	al, ch edi
BBA6978C add	edi, 1
BBA6978F BOX	al, [ebp+cax+#]
88469783 xor	[edi], al
08469785 seb	e1, 1
88469788 pap	esi
88469789 dec	1/1
88469788 jaz	1ac_469722

		L	
🖬 🖂 🖾			
0046932E			
0046932E	loc 4693	32E:	
0046932E	inc	ecx	
0046932F	shl	ecx,	1
00469331			7B7Dh
00469337	and	ecx,	edi
00469339	inc	ecx	
0046933A			eax, cl
0046933D		eax,	6151h
00469343		ecx	
00469344	and	ecx, eax,	26DF4C7Dh
00409340	sub	edx,	0E3C8h
00469352		edx,	0BE572435h
00469358		ecx,	
0046935B			0A5h
0046935E	sub	eax,	
00469364		ecx,	
00469366		ecx,	15h
0046936A	bts	ecx,	OCDh
0046936E	sub		ØFB16h
00469374	and	ecx,	
00469376	jmp	shor	t loc_469379
🖬 🖂 🖾			
00469379			
00469379	1oc_4693	79:	
	imul	ecx,	eax
	sub		739Ch
00469382			ds:168FBCAD
00469388			0AE 079465h
0046938E	lea		ds:467F6C1D
00469394			OD3DBh
0046939A 0046939E	imul	-	eax, OD5h
0046939E 004693A1		ecx, ecx,	
004093A1 004693A4		eax,	100B67D7h
004693AA	shld		eax, OCCh
004693AE	and	ecx,	
004693B0	jmp		t 1oc 4693B3
	2.1		



HM4073H HM4073 .





Loop 2

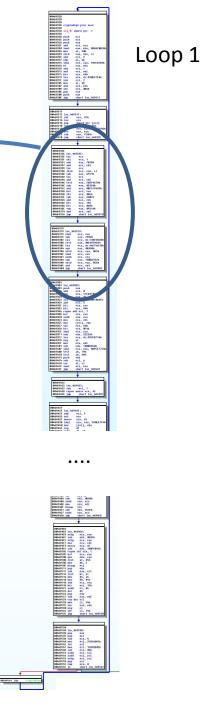
shert los_6696

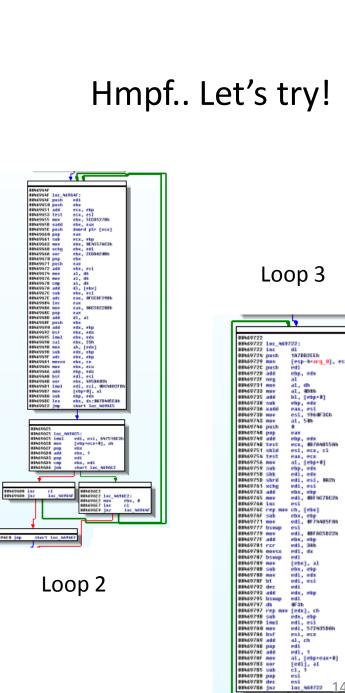
885 696E0 jap

Loop 3

88469722	
88469722 loc_451 88469722 inc	722:
BBA69725 push	1070R255h
88469729 mov	[esp-hearg_0], esi
8846972C push	eti
08469720 add	cop, edx
8846972F 085	al
88469731 mov	al, di
88469733 mov 88469735 add	al, 808h bl, [shp+8]
88469738 sub	ebp, edx
GEA69738 xadd	eax, esi
88469730 mov	eax, esi esi, 1960Fach
88469743 mov	al, 500
08469746 push	
BEA65748 pap BEA65749 add	eax ebp, edx
BBA69748 test	PCE, 807648556b
98469751 shld	esi, ecx, cl
BBAG 975k test	eax, ecx
08469756 mov	al, [ehp+0]
08469759 Stb	cop, edx
08469758 SDb	edì, edx
esh69750 shrd esh69761 xchg	edi, esi, 002h
98469763 add	edi, esi ebz, ebp
88469765 max	edi, BDF%C7BC2h
BBA69758 inc	esi
BBA6976C rep not	
0846976F 50b	cos, cop
88469771 mov 88469777 bswap	edi, 8F7WA05FAN esi
88469779 max	edi, BUFACSD22h
8846977F add	ebz, ebp
88469781 rcr	edi, 38h
BBA6978k mouse	edi, dx
88469787 bswap	edi
88469789 mov	[ebx], al
08469788 Sub 08469780 mov	ebs, ebp edi, eds
9846978F bt	edi, esi
@8469792 dec	edi
88469792 add	edz, obp
88469795 bswp	edi
88469797 db 88469797 rep not	0F3h
88469798 Stb	eds, ch
08469790 ime1	edi, esi
88469788 mov	edi, 572405060
88469786 bsF	esi, ecx
esadoras add	al, ch
BBA6978E pap	edi
BBA697AE add BBA697AF nov	edi, 1 al, [ebp+eax+#]
88469783 xor	[edi], al
88469785 seb	c1, 1
88469755 pep	es1
88469789 dec	^{mi}
88469788 jaz	1ac_469722

🛄 🖂 🖾				
0046932E				
0046932E	10c_4693	32E:		
0046932E	inc	ecx		
0046932F	shl	ecx,	1	
00469331		eax,		h
00469337		ecx,	edi	
00469339		ecx		
0046933A		-	eax,	
0046933D			6151	n
00469343		ecx		
00469344 00469346		ecx,		60706
	sub	-	0E30	4C7Dh
00409340				72435h
00469358		ecx,	Pay	724050
0046935B		ecx.	0A5h	
0046935E	sub		3007	
00469364		ecx,		
00469366		ecx,		
0046936A	bts	ecx.	OCDh	1
0046936E	sub		ØFB1	
00469374	and	ecx,		
00469376	jmp	shor	t 100	_469379
🖬 🖂 🖼				
00469379				
00469379	1oc_4693	79:		
00469379	imul	ecx,	eax	
0046937C	sub	eax,	7390	h
	lea			68FBCADI
00469388	test	-		79465h
0046938E	lea	ecx,		67F6C1D
00469394	sub		0D3DI	
0046939A	-			0D5h
	imul uadd	ecx,		
004693A1	xadd	ecx,	400D	67D7h
004693A4 004693AA	sub shld			
004093HH 004693AE	and	ecx, ecx,		OCCh
004093HE 004693B0	imp			4693B3
00107000	J. W.	Short	. 100	

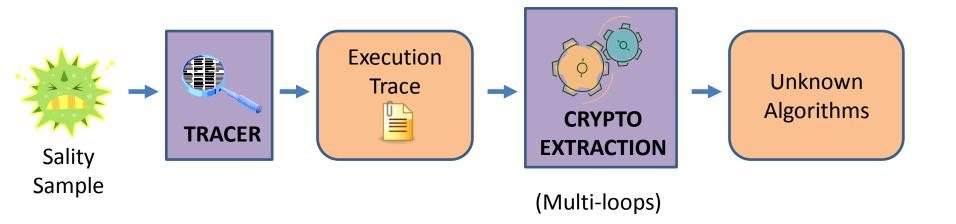


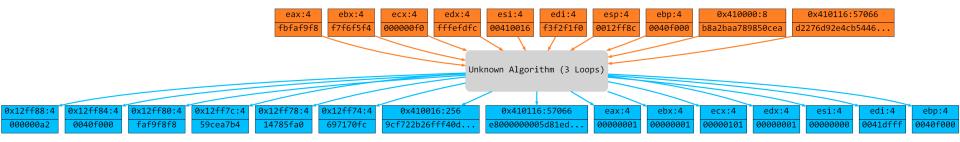


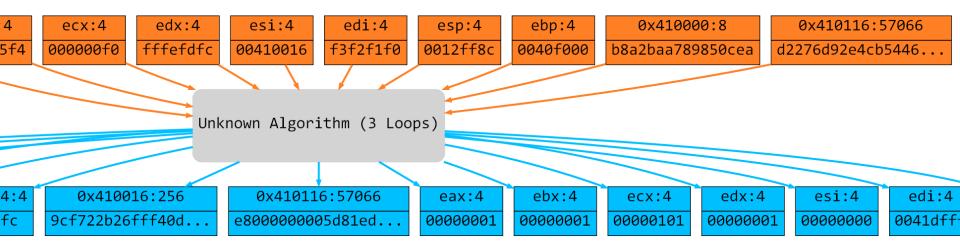
88469788 pap 88469789 dec

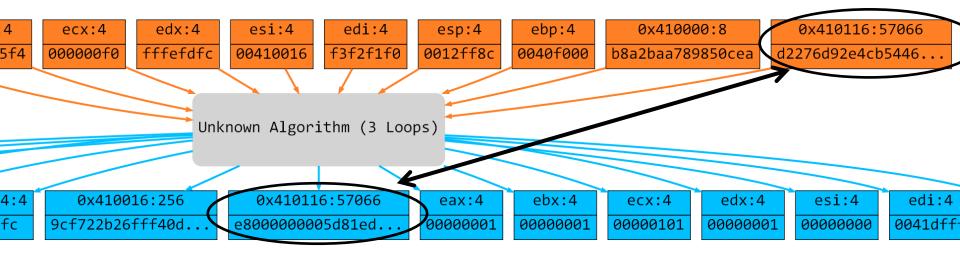
BRAGOTER jaz

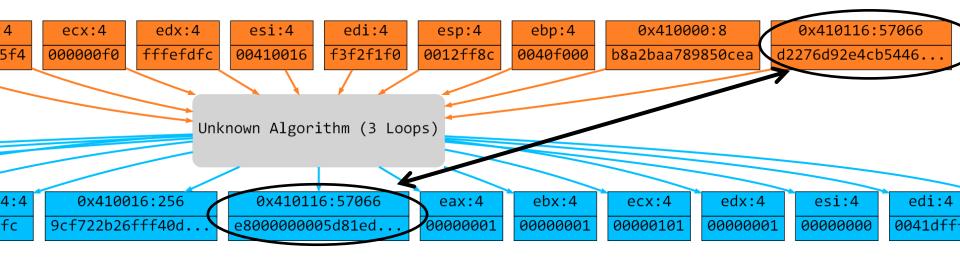
14



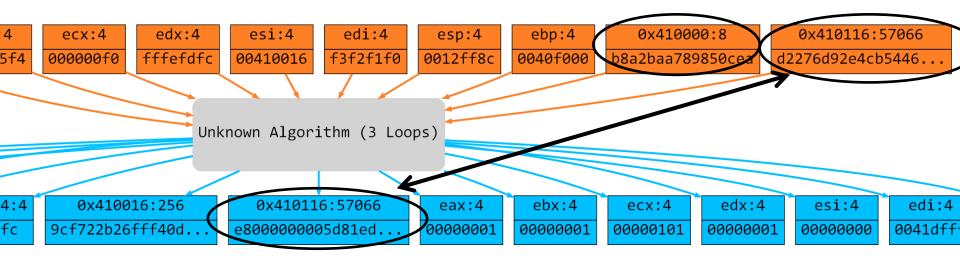




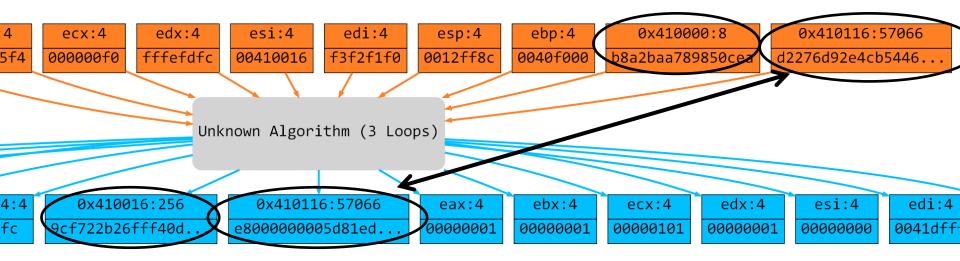




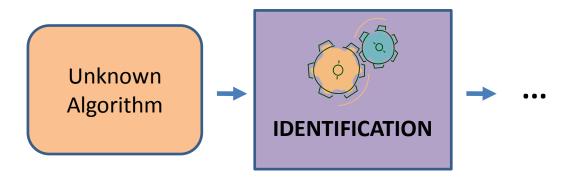
X86 ExecutableCode!



X86 ExecutableCode!



X86 ExecutableCode!

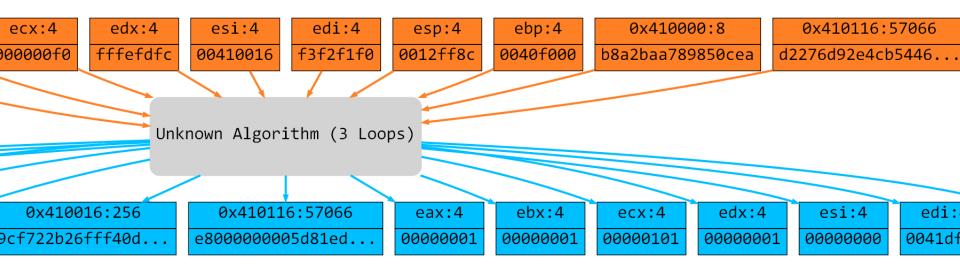


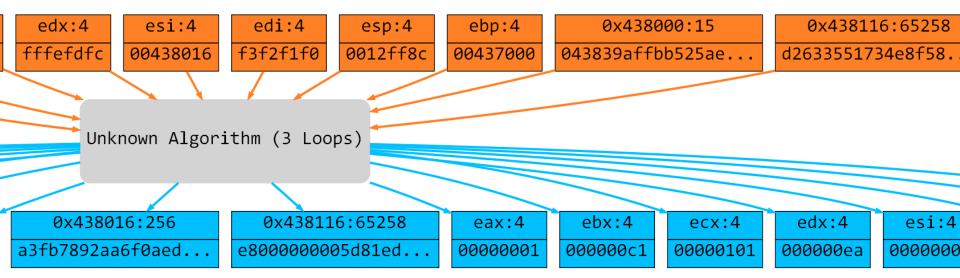
```
STARTED AT
2012-04-10 16:29:05.135000
** Crypto Algorithm Identification starting !**
5 input parameters
4 output parameters
All possible input values generation... Done!
All possible output values generation... Done!
Build internal structure... Done!
Comparison phase starting... Test for RC4...
** Found RC4 **
 ===> Key (8 bytes) : b8a2baa789850cea
 ===> Crypted text (57066 bytes) : d2276d92e4cb5446...
```

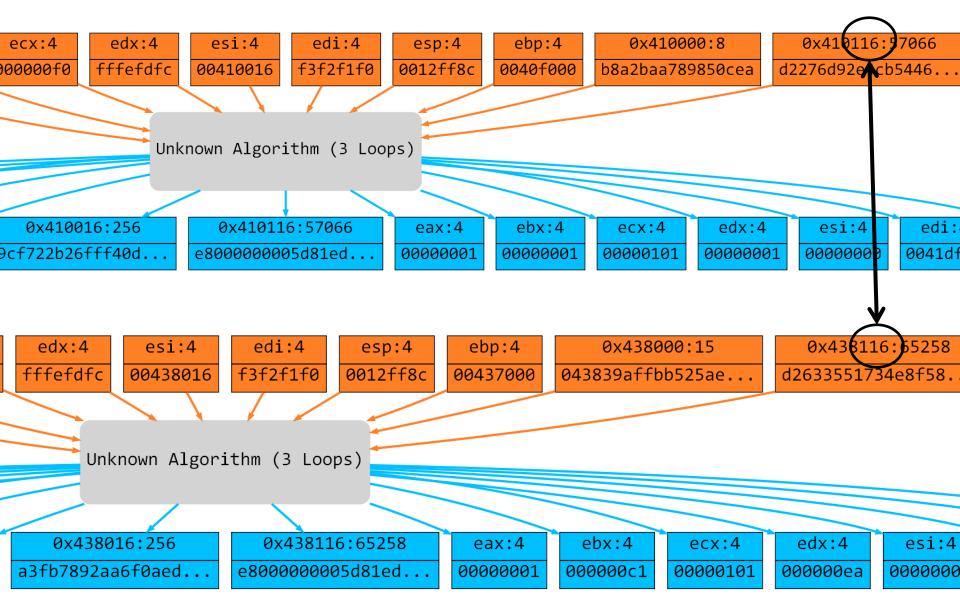
===> Decrypted text (57066 bytes) : e8000000005d81ed...

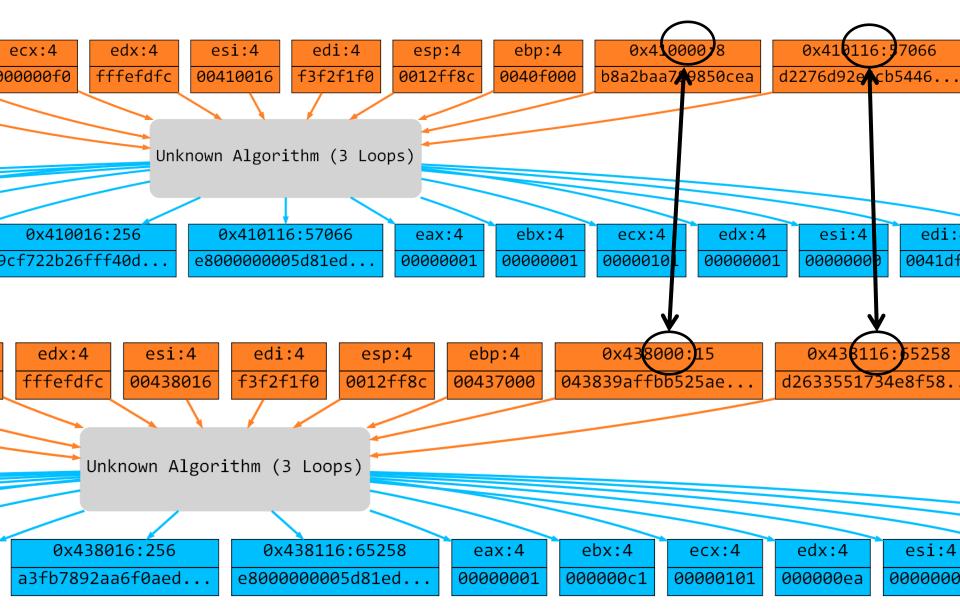
ENDED AT

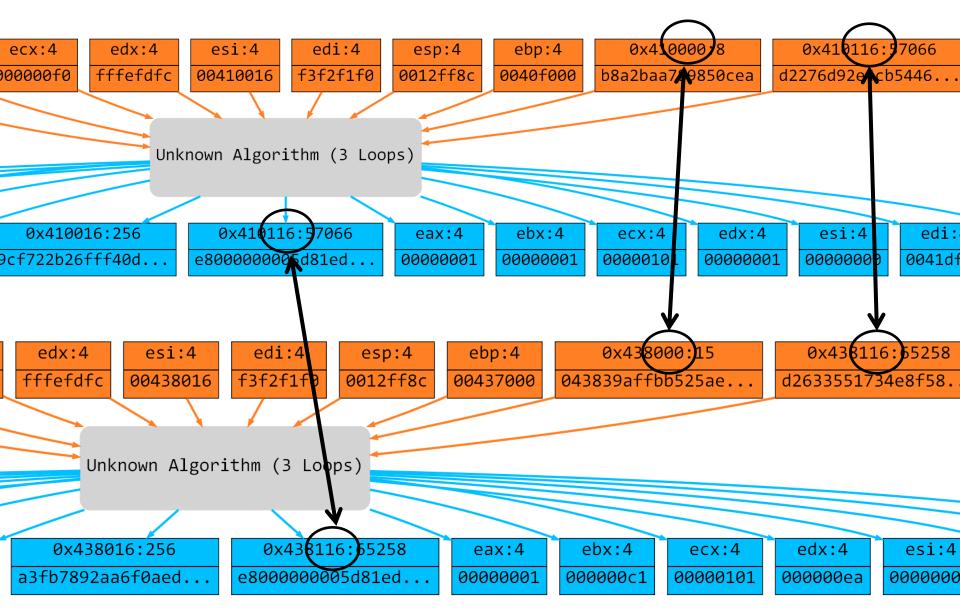
2012-04-10 16:29:06.929000

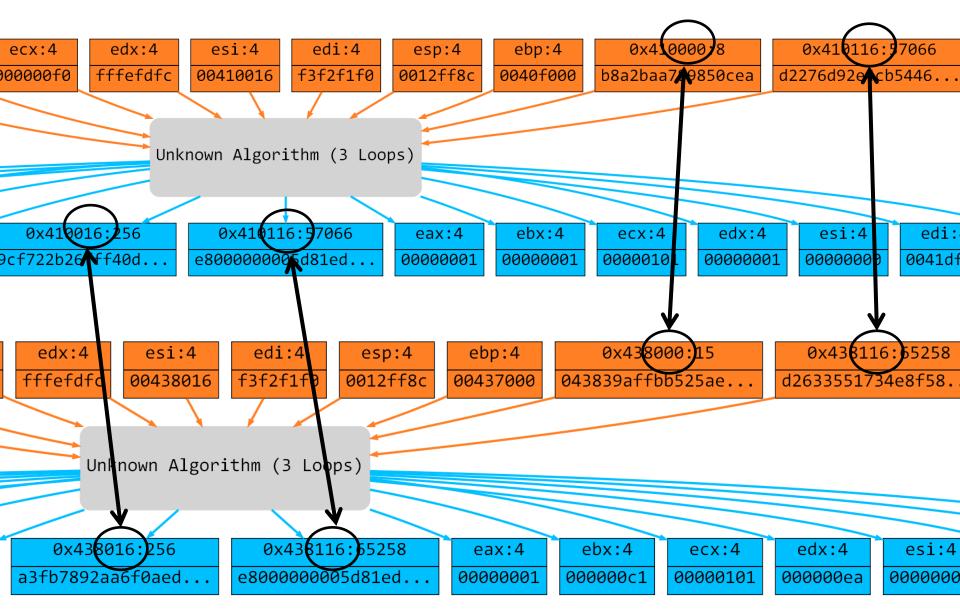


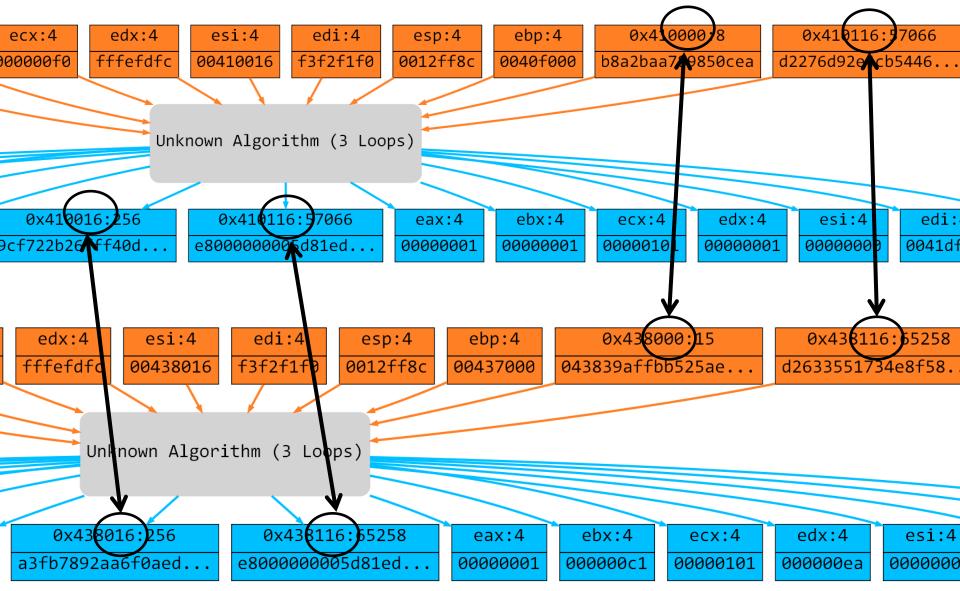










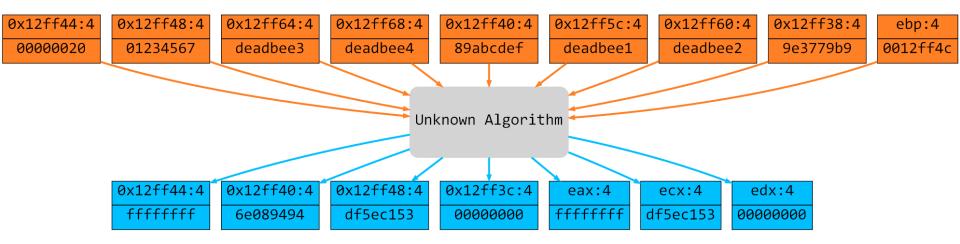


Crypto parameters always at the same offsets!

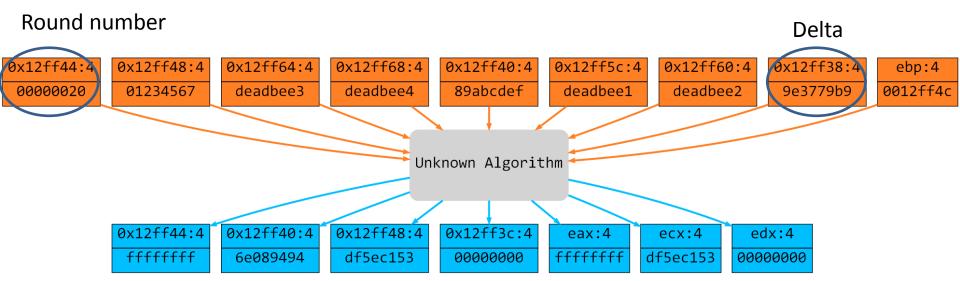
Modified TEA

MORE EXAMPLES!

Remember This ?



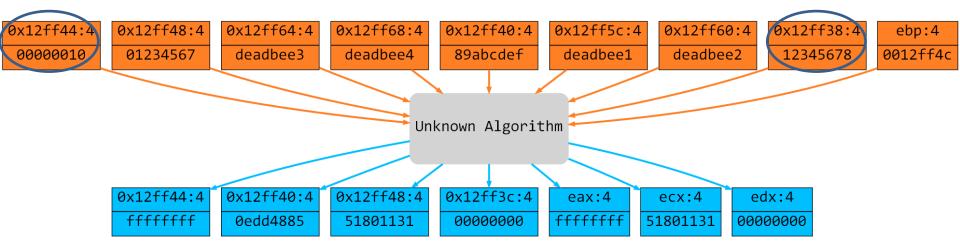
Remember This ?



The magic TEA constant (*delta*) and the round number are seen as input parameters, because they are initialized *before* the loop and used inside.

Modified TEA Implementation

- delta = 0x12345678 (normally 0x9E3779B9)
- round number = 16 (normally 32)



• TEA reference implementation extended:

def tea (input_text, key):

•••

• TEA reference implementation extended:

def tea (input_text, key, delta, round_number):

•••

• TEA reference implementation extended:

•••

def tea (input_text, key, delta, round_number):

```
STARTED AT
2012-04-08 13:16:27.918000
** Crypto Algorithm Identification starting !**
8 input parameters
3 output parameters
All possible input values generation... Done!
All possible output values generation... Done!
Build internal structure... Done!
                            Ta
                                  for TEO xmodifiedx decruption
** Found *modified* TEA decryption **
 ===> Key (16 bytes) : deadbee1deadbee2deadbee3deadbee4
 ===> Delta (4 bytes) :
                         12345678
 ===> Round number (4 bytes) : 00000010
 ===> Crupted text (8 bytes) : 0123456789abcdef
===> Decrypted text (8 bytes) : 51801131edd4885
ENDED AT
2012-04-08 13:21:11.562000
```

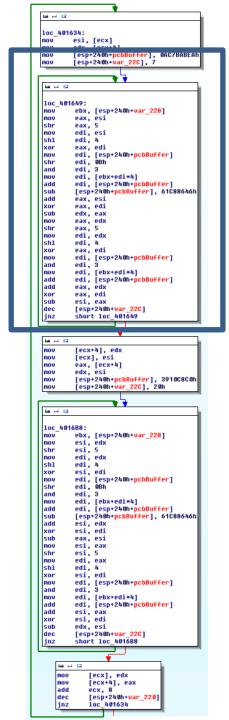
Example: Mozilla CTF

• Challenge "Awesome Corp. Secured Ranges"

• Binary program protected by PE Spin

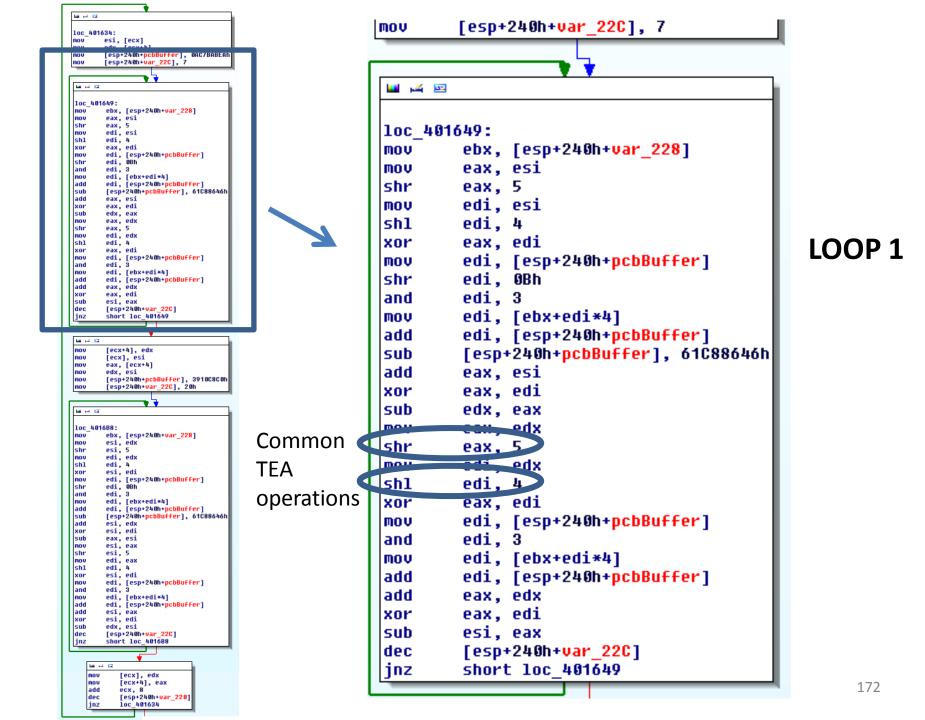
• In the core binary, a strange algorithm...

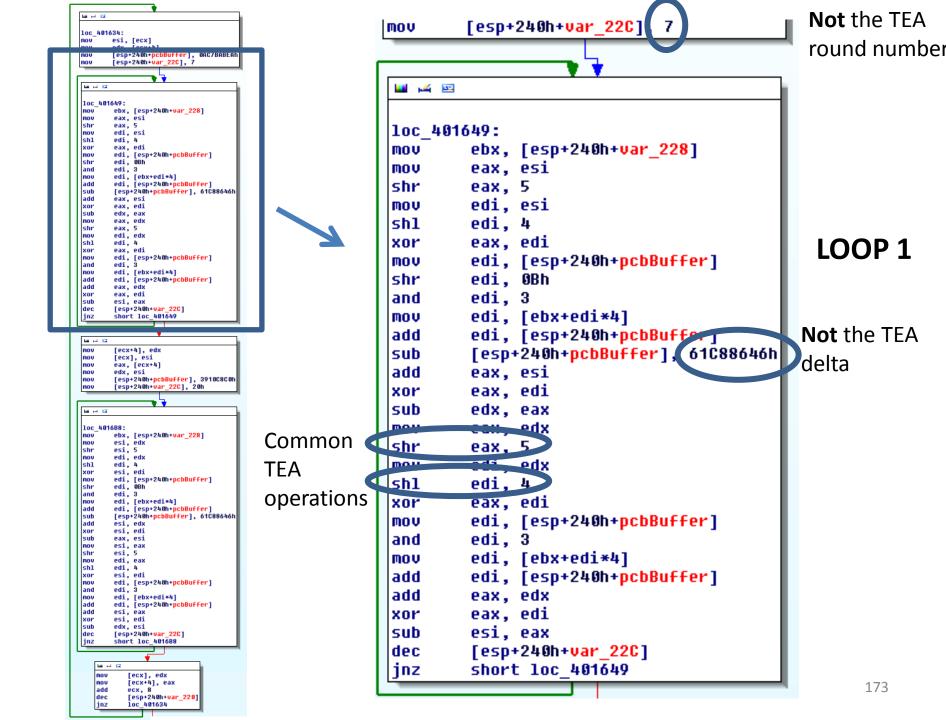
<pre> Loc_40163A: nov ecs, [ecx] nov [esp+240h+pcbBuffer], 0AC7BABEAN nov [esp+240h+var_22], 7 loc_401649: nov ecs, [esp+240h+var_228] nov ecs, esi nov ecs, esi sh1 edi, 4 xor ecs, edi sub ecs, esi nov ecs, esi sub ecs, esi sub ecs, esi sub ecs, esi sub ecs, esi nov ecs, ecsi sub ecs, eax di sub ecs, ecs nov ecs, edi sub ecs, eax sub ecs, ecs nov ecs, ecdi sub ecs, ecs</pre>		
<pre>nov esi, [ecx*] nov [esp+240h+pcbBuffer], 0AC7BABEAN nov [esp+240h+var_22C], 7 iw if c ic_401649: nov ebx, [esp+240h+var_228] nov eax, esi shr eax, 5 nov edi, esi shr edi, 0Bh and edi, [esp+240h+pcbBuffer] shr edi, 0Bh and edi, [esp+240h+pcbBuffer], 61C88646h add eax, esi shr eax, 5 nov edi, eax nov eax, edi sub edi, eax nov eax, edi sub edi, [esp+240h+pcbBuffer] and eax, edi sub edi, edx shr eax, edi sub esi, eax nov edi, [esp+240h+pcbBuffer] add eax, edi sub esi, eax nov edi, esi nov [ecx+4], edx nov [ecx+4], edx nov edi, esi nov edi, esi nov edi, esi shr edi, 4 xor esi, edi sub edi, esi shr edi, 4 xor esi, edi anov edi, [esp+240h+pcbBuffer] add edi</pre>	₩ •4 	τ
<pre>nov edx, [ecx+4] nov [esp+240h+var_22C], 7 nov [esp+240h+var_22C], 7 nov eax, esi shr eax, 5 nov edx, esi shr eax, 5 nov edi, [esp+240h+pcbBuffer] shr edi, 9Bh ad edi, [esp+240h+pcbBuffer] shr edi, 9Bh ad edi, [esp+240h+pcbBuffer] shr edi, 12 nov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] add eax, esi shr eax, 5 nov edi, [esp+240h+pcbBuffer] add eax, esi shr eax, 5 nov edi, [esp+240h+pcbBuffer] add eax, esi sub edi, eax shr eax, 6di sub edi, [esp+240h+pcbBuffer] add eax, edi sub edi, [esp+240h+pcbBuffer] add eax, edi sub esi, eax dec [esp+240h+pcbBuffer] add eax, edi sub esi, eax dec [esp+240h+pcbBuffer], 3910C8C0h nov [ecx], esi nov eax, [edi shr eax, 5 nov edi, [esp+240h+var_222] nov eax, [edi shr eax, 5 nov edi, [esp+240h+var_222], 20h nov [ecx], esi nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+var_222], 20h nov [esp+240h+pcbBuffer], 3910C8C0h nov [ecx], esi nov edi, [edx shr esi, 5 nov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi sub eax, esi nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] add edi, [e</pre>		1634: esi fecyl
w → CZ loc_4016A9: nov ebx, [esp+240h+var_228] nov ebx, [esp+240h+pcbBuffer] shr eax, edi nov edi, [esp+240h+pcbBuffer] shr eax, edi nov edi, [esp+240h+pcbBuffer] shr eax, edi sub [esp+240h+pcbBuffer], 61C88640h add eax, edi sub (esp+240h+pcbBuffer] add eax, edi sub (esp+240h+pcbBuffer] add eax, edi sub (esp+240h+pcbBuffer] add eax, edi sub esp+240h+pcbBuffer] add eax, edi sub esi, eax dec [esp+240h+pcbBuffer], 3910C8C0h nov [ecx+k], edx nov edi, edx shr edi, edx nov edi, edx <tr< th=""><th>nov</th><th></th></tr<>	nov	
Image: Image		[esp+240h+pcbBuffer], 0AC7BABEAh [esp+240h+var_22C], 7
<pre>loc_401649: now eax, esi shr eax, 5 now edi, esi shr edi, 08h and edi, [esp+240h+pcbBuffer] shr edi, 08h and edi, [esp+240h+pcbBuffer] sub [esp-240h+pcbBuffer], 61C88646h add eax, edi sub edx, eax now edi, [esp+240h+pcbBuffer] and edi, a shr eax, edi sub edi, eax shr eax, edi now edi, [esp+240h+pcbBuffer] and edi, a shr edi, 4 xor eax, edi sub edi, eax shr eax, edi now edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add eax, edi sub esi, eax dec [esp+240h+uar_220] now edi, edx shr eax, edi sub esi, eax dec [esp+240h+uar_220] now edi, edx shr edi, 4 xor eax, edi sub esi, eax dec [esp+240h+uar_220], 20h im == tz now edi, edx shr esi, edi now edi, [esp+240h+uar_228] now edi, edx shr esi, st now edi, [esp+240h+pcbBuffer] add edi, [esp+240h+uar_228] now edi, edx shr esi, edi shr esi, edi shr esi, edi shr esi, edi shr edi, 4 xor esi, edi shr esi, 5 now edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] and edi, 3 now edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi sub eax, esi now edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi sub eax, esi now edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi sub eax, esi now edi, [esp+240h+pcbBuffer] and edi, eax shr esi, s now edi, [esp+240h+pcbBuffer] and edi, eax shr esi, s now edi, [esp+240h+pcbBuffer] and edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, eix xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] add edi, eax xor esi, edi sub eax esi dec [esp+24</pre>		
<pre>loc_401649: now eax, esi shr eax, 5 now edi, esi shr edi, 08h and edi, [esp+240h+pcbBuffer] shr edi, 08h and edi, [esp+240h+pcbBuffer] sub [esp-240h+pcbBuffer], 61C88646h add eax, edi sub edx, eax now edi, [esp+240h+pcbBuffer] and edi, a shr eax, edi sub edi, eax shr eax, edi now edi, [esp+240h+pcbBuffer] and edi, a shr edi, 4 xor eax, edi sub edi, eax shr eax, edi now edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add eax, edi sub esi, eax dec [esp+240h+uar_220] now edi, edx shr eax, edi sub esi, eax dec [esp+240h+uar_220] now edi, edx shr edi, 4 xor eax, edi sub esi, eax dec [esp+240h+uar_220], 20h im == tz now edi, edx shr esi, edi now edi, [esp+240h+uar_228] now edi, edx shr esi, st now edi, [esp+240h+pcbBuffer] add edi, [esp+240h+uar_228] now edi, edx shr esi, edi shr esi, edi shr esi, edi shr esi, edi shr edi, 4 xor esi, edi shr esi, 5 now edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] and edi, 3 now edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi sub eax, esi now edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi sub eax, esi now edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi sub eax, esi now edi, [esp+240h+pcbBuffer] and edi, eax shr esi, s now edi, [esp+240h+pcbBuffer] and edi, eax shr esi, s now edi, [esp+240h+pcbBuffer] and edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, eix xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, eax xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] add edi, eax xor esi, edi sub eax esi dec [esp+24</pre>	1 14 5	-
<pre>bou ebs, [esp+240h+var_228] hou eas, esi shr eas, 5 hou edi, esi shr eas, edi nou edi, [esp+240h+pcbBuffer] shr edi, 08h and edi, 3 nou edi, [esp+240h+pcbBuffer] sub (esp+240h+pcbBuffer], 61088646h add eas, esi sub eds, eas hou eas, edi sub eds, eas hou eas, edi sub edi, [esp+240h+pcbBuffer] and edi, 3 nou edi, [esp+240h+pcbBuffer] add eas, esi hi edi, 4 xor eas, edi sub edi, [esp+240h+pcbBuffer] add eas, esi hou edi, [esp+240h+pcbBuffer] add eas, esi hou edi, [esp+240h+pcbBuffer] add eas, esi hou edi, [esp+240h+pcbBuffer] add eas, esi hou eas, edi sub esi, eas dec [esp+240h+uar_220] jnz short loc_A01649 bu i= iz nou eds, esi hou edi, [esp+240h+uar_220], 20h</pre>		
<pre>bov eax, esi shr eax, 5 nov edi, esi shr edi, 0Bh and edi, 3 nov edi, [esp+240h+pcbBuffer] shr edi, 0Bh and edi, 3 nov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] add eax, esi sub edx, eax nov eax, edi sub edx, eax nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] add eax, edx shr eax, edi sub esi, eax deci [esp+240h+var_220] nov [ecx+1] nov [ecx+4], edi nov [ecx+2], esi nov eax, [esp+240h+var_222], 20h</pre>		ebx, [esp+240h+var_228]
<pre>bov edd, esi sh1 edd, 4 xor eax, edd nov edd, [esp+2400+pcbBuffer] shr edd, 08h add edd, 3 nov edd, [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer], 61C88646h add eax, esi xor eax, edi sub edx, eax nov edd, [esp+2400+pcbBuffer] add edd, 3 nov edd, [esp+2400+pcbBuffer] add eax, edx sh1 edd, 4 xor eax, edi sub edd, [esp+2400+pcbBuffer] add eax, edx sor eax, edi sub esi, eax dec [esp+2400+pcbBuffer] add eax, edx sor eax, edi sub esi, eax dec [esp+2400+pcbBuffer], 3910C8C00 nov [ex,], edx nov edd, [esp+2400+pcbBuffer], 3910C8C00 nov edd, [esp+2400+pcbBuffer], 3910C8C00 nov edd, [esp+2400+pcbBuffer] add edd, [esp+2400+pcbBuffer] add edd, [esp+2400+pcbBuffer], 3910C8C00 nov edd, edd sub esi, eax dec [esp+2400+pcbBuffer], 3910C8C00 nov edd, [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] add edi, [esp+2400+pcbBuffer] sub [esp+2400+pcbBuffer] add edi, [esp+2400+pcbBuffer] a</pre>		eax, esi
<pre>xor eax, edi nov edi, [esp+240h+pcbBuffer] shr edi, 0Bh and edi, 3 nov edi, [esp+240h+pcbBuffer] sub [esp-240h+pcbBuffer], 61C88646h add eax, esi nov edi, edx shr eax, edi shr edi, 4 xor eax, edi shr edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add eax, edx shr eax, edi sub esi, eax dec [esp+240h+var_220] inz short loc_401649</pre>	nov	edi, esi
<pre>nov edd, [esp+240h+pcbBuffer] shr edd, 0Bh and edd, 3 nov edd, [ebx+edi+4] add edd, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer], 61C88646h dd eax, edx nov eax, edi sub edd, eax nov edd, edx shr eax, 5 nov edd, edx shr eax, 5 nov edd, [esp+240h+pcbBuffer] and edd, 3 nov eax, edi nov eax, edi sub edd, [esp+240h+pcbBuffer] and edd, [esp+240h+pcbBuffer] add eax, edx xor eax, edi sub eax, edi shr edi, esp+240h+var_222], 20h shr edi, 0Bh anov edi, [esp+240h+var_222], 20h shr edi, 0Bh anov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub eax, edi sub eax, esi nov edi, eax shr edi, 0Bh and edi, 3 nov edi, eax shr edi, 4 xor esi, edi sub eax, esi shu edi, 4 xor esi, edi sub eax, esi shu edi, 4 xor esi, edi sub eax, esi shu edi, 4 xor esi, edi sub eax, esi shu edi, 1 add edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] anov edi, [esp+240h+pcbBuffer] add edi, [esp+24</pre>		edi, 4 eax. edi
and edi, 3 nov edi, [ebx+edi+4] add edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer], 61C88646h dd eax, esi sub edx, eax nov eax, edi sub edx, eax nov edi, edx shr eax, 5 nov edi, edx shr eax, 6 nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] add eax, [esp+240h+pcbBuffer] add edx, [esp+240h+pcbBuffer], 3910C8C0h nov [ecx+1], edx nov edx, ecx+1] nov edx, esi nov edi, esp+240h+pcbBuffer], 3910C8C0h nov esi, edi shr esi, 5 nov edi, edx shr esi, 5 nov edi, edx shr esi, 5 nov edi, [esp+240h+pcbBuffer] shr edi, 90h and edi, 3 nov edi, edx shr esi, 5 nov edi, [esp+240h+pcbBuffer] shr edi, 90h and edi, 3 nov edi, edx shr esi, 5 nov edi, [esp+240h+pcbBuffer] shr edi, 90h and edi, 3 nov edi, [esp+240h+pcbBuffer] shr esi, 5 nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] and edi, 6 sub eax, esi nov edi, [esp+240h+pcbBuffer] and edi, 6 sub eax, esi he di, 4 xor esi, edi sub eax, esi he di, 4 xor esi, edi sub eax, esi dec [esp+240h+var_222] nov edi, [esp+240h+var_220] sub eax, esi dec [esp+240h+var_220] he di, eax he di, eax h	nov	edi, [esp+240h+pcbBuffer]
mov edi, [ebx+edi+i] add edi, [ecbx+240h+pcbBuffer], 61C88646h add eax, edi nov eax, edi sub eex, edi nov eax, edi sub edi, edx shr edi, edx shr edi, edx shr edi, edx shr edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add eax, edi sub esi, eax dec [esp+240h+pcbBuffer] add eax, edi sub esi, eax dec [esp+240h+pcbBuffer], 3910C8C0h nov [ecx+4], edx nov [ecx+4], edx nov edi, edx shr esi, edx nov [esp+240h+pcbBuffer], 3910C8C0h nov [esp+240h+pcbBuffer], 3910C8C0h nov edi, edx shr esi, edi nov edi, edx shr esi, edi nov edi, [esp+240h+pcbBuffer] shr esi, edi </th <th></th> <th>edi, ØBN edi, 3</th>		edi, ØBN edi, 3
<pre>sub [esp+240h+pcbBuffer], 61C88646h add eax, esi sub eax, eddi sub eax, eddi sub edd, eax, eddi sub edd, edx, eddi nov edd, [esp+240h+pcbBuffer] and edd, [esp+240h+pcbBuffer] add eax, eddi sub edd, [esp+240h+ucbBuffer] add eax, eddi sub esi, eax dec [esp+240h+uar_22C] jnz short loc_A01649 </pre> <pre> u == tz nov edd, [esp+240h+uar_22C] inz short loc_A01649 </pre> <pre> u == tz nov edd, [esp+240h+uar_22C] shor edd, eddi sub eax, eex+a] nov edd, esi nov edd, esi nov edd, esi nov edd, esi nov edd, [esp+240h+uar_22C], 20h </pre> <pre> u == tz nov edd, [esp+240h+uar_22C] shor edd, eddi sh edd, 4 xor esi, edd and edd, [esp+240h+uar_22C] shr edd, eddi sh edd, 4 xor esi, edd and eddi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub eax, esi nov edd, [esp+240h+pcbBuffer] sub eax, esi nov edd, [esp+240h+pcbBuffer] add eddi, [esp+240h+pcbBuffer] and eddi, a xor esi, edd sub eax, esi shr esi, 5 nov edd, [eax shr esi, 6d sub eax, esi shr esi, 5 nov edd, [esp+240h+pcbBuffer] add eddi, [esp+240h+pcbBuffer] and eddi, a xor esi, edd sub eax, esi shr esi, 5 shr esi, 6 s</pre>		edi, [ebx+edi*4]
<pre>xor eax, edi sub edx, eax mov eax, edx shr eax, 5 mov edi, edx shr eax, edi mov edi, [esp+240h+pcbBuffer] and edi, 3 mov edi, [esp+240h+pcbBuffer] add eax, edx xor eax, edi sub esi, eax dec [esp+240h+var_220] jnz short loc_A01649</pre>		[esp+240h+pcbBuffer], 61C88646h
wb edx, eax nov edx, 5 nov edi, edx sh1 edi, edx sh1 edi, edx sh1 edi, edx sh1 edi, edx nov edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edx, edi sub esi, eax dec [esp+240h+pcbBuffer], 3910000 [inc short loc_A01649 nov [ecx+4], edx nov [ecx+4], edx nov [esp+240h+pcbBuffer], 39100000 [sp+240h+pcbBuffer], 39100000 [esp+240h+pcbBuffer], 391000000 nov edx, esi nov edx, edx shr edi, edx shr edi, esp+240h+pcbBuffer] nov edi, esp+240h+pcbBuffer] sub eax shr edi, esp+240h+pcbBuffer] sub eax shr edi, esp+240h+pcbBuffer] sub eax		eax, esi eax, edi
<pre>shr eax, 5 nov edi, edx sh1 edi, 4 xor eex, edi nov edi, [esp+240h+pcbBuffer] add edi, 3 nov edi, [eby+240h+pcbBuffer] add eax, edx xor eax, edi sub esi, eax dec [esp+240h+var_220] inz short loc.401649 u u u (ecx, esi nov [ecx+4], edx nov [ecy+240h+var_220], 3910C8C0h nov [esp+240h+var_220], 20h u u eax, [ecx+4] nov edx, esi nov edi, [esp+240h+var_220], 20h u u u (esp+240h+var_220], 20h u (esp+240h+var_220], 20h u (esp+240h+var_220], 20h u (esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub (esp+240h+pcbBuffer] sub (esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBu</pre>	sub	edx, eax
nov edi, 4 sh1 edi, 4 xor eax, edi nov edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+var_220] jnz short loc_A016A9 im im im im <td< th=""><th></th><th>eax, 5</th></td<>		eax, 5
xor eax, edi nov edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add eax, edi xor eax, edi ub eax, edi gez (esp+240h+var_222)]nz short loc_401649 ub mov [esp+240h+var_222], 20h mov [esp+240h+var_222], 20h mov [esp+240h+var_222], 20h uc_401688: mov esp+240h+var_222], 20h uc_401688: mov esp+240h+pcbBuffer], 3910C8C0h mov esp+240h+var_222], 20h uc_401688: mov esp+240h+pcbBuffer], 3910C8C0h mov esp+240h+var_222], 20h mov esp+240h+var_222], 20h nov esp+240h+pcbBuffer], 3910C8C0h mov edi, edx shr edi, edx nov edi, edx shr edi, edx shr edi, edx sor esi, edi		edi, edx
nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [ebx+edi+1] add eax, edx add eax, edx edi sub esi, eax sur eax, edi sub esi, eax edi (esp+240h+pcbBuffer] inz short loc_A01649 sur eax, edi sub esi, eax dec [esp+240h+pcbBuffer], 39100800h nov [ecx+1], edi nov eax, [ecx+4] nov eax, [ecx+4] nov eax, [ecx+4] nov edx, esi nov eds, esi nov eds, esi nov edi, esp+240h+pcbBuffer], 39100800h nov [esp+240h+pcbBuffer], 39100800h nov edi, [esp+240h+var_222], 20h shi edi, 4 xor esi, edi shi edi, 4 xor esi, edi sub eax, esi nov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] and edi, 3 nov edi, eax shi edi, 4 xor esi, edi sub eax, esi shi edi, 4 xor esi, edi sub eax, esi dec [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240	xor	eax, edi
nov edi, [ebx+edi+i] add edi, [ecy+2400+pcbBuffer] add eax, edx xor eax, edi sub esi, eax dec [esp+2400+var_220] jnz short loc_A01649 mov [ecx+i], edx nov [esy+2400+pcbBuffer], 391008000 nov [esy+2400+var_222], 200 mov [esy+2400+var_222], 200 mov edx, [esy+2400+var_228] nov edx, edx shr esi, edx shr esi, edx shl edi, edx shl< edi, 4 xor xor esi, edi nov edi, [esp+2400+pcbBuffer] sub eax shr esi, edi nov edi, [esp+2400+pcbBuffer] add edi, eax shr esi, eax		edi, [esp+240h+pcbBuffer]
add eax, edx xor eax, edi sub esi, eax dec [esp+240h+var_220] inz short loc_A01649 w w [ecx+1], edx nov [ecx+1], edx nov edx, [ecx+4] nov edx, [ecx+4] nov edx, esi nov [esp+240h+var_220], 20h w w esi, edx shr esi, 5 nov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] shr edi, 44 xor esi, edx shl edi, 4 xor esi, edi nov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer] sub eax, esi nov edi, [esp+240h+pcbBuffer] sub eax, esi nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] sub eax, esi nov edi, [esp+240h+pcbBuffer] add edi, [nov	edi, [ebx+edi*4]
xor eax, edi sub esi, eax dec [esp+2408h+var_220] jnz short loc_401649 nov [ecx+k], edx nov [esp+240h+var_220], 20h u [esp+240h+var_222], 20h u [esp+240h+var_222], 20h u [esp+240h+var_222], 20h u esp+240h+var_222], 20h u esp+240h+pcbBuffer] u esp+240h+pcb		edi, [esp+240h+pcbBuffer] eax. edx
dec [esp+240h+var_22C] jnz short loc_401649 mov [ecx], esi mov [ecx+h], edx mov [ecx], esi mov [esp+240h+pcbBuffer], 3910C8C0h mov [esp+240h+pcbBuffer], 3910C8C0h mov [esp+240h+pcbBuffer], 20h im → iz [csp+240h+pcbBuffer], 3910C8C0h mov [esp+240h+pcbBuffer], 20h im → iz [csp+240h+pcbBuffer] nov edi, edx shr edi, [esp+240h+pcbBuffer] shr edi, [esp+240h+pcbBuffer] shu [csp+240h+pcbBuffer] shu [csp+240h+pcbBuffer] shu [csp+240h+pcbBuffer] shu [csp+240h+pcbBuffer] add esi, edi sub eax shr esi, 5 mov edi, [esp+240h+pcbBuffer] add	xor	eax, edi
<pre>im → □ im → □ im</pre>		[esp+240h+var_22C]
nov [ecx+4], edx nov [ecx+4], edx nov [ecx, esi nov edx, [ecx+4] nov edx, esi nov [esp+240h+vch20uffer], 3910C8C0h nov [esp+240h+var_22C], 20h with the signed state s	jnz	short loc_401649
nov [ecx+4], edx nov [ecx+4], edx nov [ecx, esi nov edx, [ecx+4] nov edx, esi nov [esp+240h+vch20uffer], 3910C8C0h nov [esp+240h+var_22C], 20h with the signed state s		
nov [ecx], esi nov eax, [ecx+1] nov edx, esi nov [esp+240h+pcbBuffer], 3910C8C0h nov [esp+240h+var_22C], 20h u [esp+240h+var_22C], 20h u [esp+240h+var_22C], 20h u [esp+240h+var_22C], 20h nov ebx, [esp+240h+var_222] nov ebx, [esp+240h+var_228] nov edi, edx shr edi, edx shr edi, [esp+240h+pcbBuffer] shr edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add esi, edi sub eax shr esi, edi nov edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi sub eax shr esi, edi nov edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+var_22C] jnz short l		
nov edx, esi nov [esp+240h+pcbBuffer], 3910C8C0h nov [esp+240h+var_22C], 20h imov [esp+240h+var_22C], 20h imov ebx, [esp+240h+var_22C], 20h imov ebx, [esp+240h+var_22C], 20h imov ebx, [esp+240h+var_22C] nov ebx, [esp+240h+var_22C] nov edi, edx shr edi, [ebx+edi*4] add edi, [ebx+edi*6] sub eax shr esi, edi sub< eax, esi nov nov edi, [ebx+edi*4] add edi,		[ecx+4], eax [ecx], esi
nov [esp+240h+pcbBuffer], 3910CSC0h nov [esp+240h+var_22C], 20h im		
Import Import Inc_401688: Import now ebx, [esp+240h+var_228] now est, edt now est, edt now est, edt now edt, edx shr est, edt now edt, edx shr est, edt now edt, [esp+240h+pcbBuffer] shr edt, [esp+240h+pcbBuffer] add edt, [esp+240h+pcbBuffer] sub [esp-240h+pcbBuffer], 61C88646h xor est, edt now edt, [esp+240h+pcbBuffer] add edt, eax shr est, edt now edt, [esp+240h+pcbBuffer] add edt, [esp+240h+pcbBuffer] add edt, [esp+240h+pcbBuffer] add edt, [esp+240h+pcbBuffer] add edt, [esp+240h+var_220] inow [now edt, [ecsp+240h+var_220] inow [ecx], edx now [ecx], edx inow [ecx], edx		eax, [ecx+4]
<pre>loc_401688: nov ebx, [esp+240h+var_228] nov esi, edx shr esi, 5 nov edi, edx shr edi, edx avor esi, edi nov edi, [esp+240h+pcbBuffer] shr edi, 08h and edi, 3 nov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer], 61088646h add esi, edx avor esi, edi sub eax, esi nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+var_220] sub eax, esi dec [esp+240h+var_220] inz short loc_40168</pre>	nov nov	edx, esi [esp+240h+pcbBuffer], 3910C8C0h
<pre>loc_401688: nov ebx, [esp+240h+var_228] nov esi, edx shr esi, 5 nov edi, edx shr edi, edx avor esi, edi nov edi, [esp+240h+pcbBuffer] shr edi, 08h and edi, 3 nov edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer], 61088646h add esi, edx avor esi, edi sub eax, esi nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+var_220] sub eax, esi dec [esp+240h+var_220] inz short loc_40168</pre>	nov nov	edx, esi [esp+240h+pcbBuffer], 3910C8C0h
nov ebx, [esp+240h+var_228] nov edi, edx shr esi, 5 nov edi, edx shr edi, edx shr edi, edx nov edi, edx nov edi, edx shr edi, edx nov edi, [esp+240h+pcbBuffer] shr edi, [esp+240h+cbBuffer] add edi, [esp+240h+cbBuffer] add edi, [esp+240h+cbBuffer] sub [esp+240h+cbBuffer], 61C88646h add edi, edx shr esi, edi nov edi, [eax shr esi, edi nov edi, [eax shl edi, [eax shl edi, [eax+brbcbBuffer] add edi, [eap+240h+pcbBuffer] add edi, [eap+240h+pcbBuffer] add edi, [ecp+240h+var_220] jnz sohr loc wov [ecx+1], eax add ecx, 8	nov nov nov	edx, esi [esp+240h+pcbBuffer], 3910C8C0h [esp+240h+var_22C], 20h
nov esi, edx shr esi, edx nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add esi, edx xor esi, edx sub eax, esi nov edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+var_220] jnz sohrt icz (#168	nov nov nov	edx, esi [esp+240h+pcbBuffer], 3910C8C0h [esp+240h+var_22C], 20h
nov edi, edx sh1 edi, 4 xor esi, edi nov edi, egp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add esi, edx xor esi, edx xor esi, edx sub eax, esi nov edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi nov edi, [esp+240h+pcbBuffer] and edi, 4 xor esi, edi nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edx, esi ucc [esp+240h+var_220] jnz short icz nov [ecx], edx nov [ecx], edx add ecx, 8 dcc [esp+240h+var_220]	nov nov nov	edx, esi [esp-240h+pcbBuffer], 3910C8C0h [esp+240h+var_22C], 20h z
<pre>shl edi, 4 xor eci, edi nov edi, [esp+240h+pcbBuffer] shr edi, 00h and edi, 3 nov edi, [ebx+edi*4] add edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer], 61088646h add eci, edx xor esi, edi sub eax, esi nov edi, eax shr esi, 5 nov edi, eax shl edi, 4 xor eci, edi add eci, [esp+240h+pcbBuffer] add eci, [ebx+edi*4] add eci, [eax+4] add eci, [eax xor esi, edi sub edx, esi dec [esp+240h+var_220] inz short loc_A0168 </pre>	nov nov nov	edx, esi [esp-240h+pcbBuffer], 3910C8C0h [esp+240h+var_22C], 20h z J16B8: ebx, [esp+240h+var_228] esi, edx
nov edi, [esp+240h+pcbBuffer] shr edi, [ebx+edi*4] add edi, [esp+240h+pcbBuffer] sub [esp+240h+pcbBuffer], 61C88646h add esi, edi nov esi, eax shr esi, eax shr esi, eax shr edi, eax shl edi, eax nov edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, eax sub edix, eai dec [esp+240h+var_220] jnz short loc_k0f88	nov nov nov	edx, esi [esp+240h+var_22C], 20h [esp+240h+var_22C], 20h 2 2 16088: ebx, [esp+240h+var_228] esi, edx esi, 5
<pre>shr edi, 0Bh and edi, 3 nov edi, [ebx+edi*4] add edi, [esp-2400+pcbBuffer] sub [esp-2400+pcbBuffer], 61C88646h add esi, edx xor esi, edi sub eax, esi nov edi, eax shr esi, 5 nov edi, eax shl edi, 4 xor eci, edi nov edi, [esp+2400+pcbBuffer] add edi, [esp+2400+pcbBuffer] add edi, [esp+2400+pcbBuffer] add edi, eax xor esi, edi sub edx, esi add esi, eax xor esi, edi sub edx, esi sub edx, esi add exi, eax xor esi, edi sub edx, esi add exi, eax xor esi, edi xor esi, edi xor exi, ed</pre>	nov nov nov nov loc_46 nov nov shr nov shl	edx, esi [esp+240h+var_22C], 20h [esp+240h+var_22C], 20h 2 1668: ebx, [esp+240h+var_228] esi, edx esi, 5 edi, edx edi, 4
<pre>nov edi, [ebx*edi*4] add edi, [esp*240h*pcbBuffer] sub [esp*240h*pcbBuffer], 61C88646h add esi, edi sub eax, esi nov esi, edi sub eax, esi nov edi, eax shr esi, 5 nov edi, [esp*240h*pcbBuffer] and edi, 3 nov edi, [ebx*edi*4] add esi, eax xor esi, edi add esi, eax xor esi, edi sub edx, esi sub edx,</pre>	nov nov nov loc_44 nov shr nov shr nov sh1 xor	edx, esi [esp-240h+pcbBuffer], 3910C8C0h [esp+240h+var_22C], 20h
<pre>sub [esp+240h+pcDBuffer], 61C88646h add esi, edx xor esi, edi sub eax, esi nou esi, eax shr esi, 5 nou edi, eax shl edi, 4 xor esi, edi nou edi, [esp+240h+pcDBuffer] add edi, [esp+240h+pcbBuffer] add esi, eax xor esi, edi sub edx, esi dec [esp+240h+var_222] jnz shrt loc_40168 </pre>	nov nov nov nov nov shr nov shl xor nov shl	edx, esi [esp-240h+pcbBuffer], 3910CSCOh [esp-240h+var_22C], 20h z 11688: ebx, [esp+240h+var_228] esi, edx esi, 5 edi, edx edi, 4 esi, edi edi, [esp-240h+pcbBuffer] edi, 08h
add esi, edx xor esi, edi sub eax, esi nov eci, eax shr esi, 5 nov edi, eax shl edi, 4 xor esi, edi nov edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+var_22C] jnz short loc_401688	nov nov loc_44 nov shr nov shr nov shr nov shr nov	edx, esi [csp-240h+pcbBuffer], 3910CSC0h [csp+240h+var_22C], 20h
<pre>sub eax, esi nov eesi, eax shr esi, 5 nov edi, eax shl edi, 4 xor eesi, edi nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [ebx+edi*4] add edi, [esp+240h+pcbBuffer] add edi, eax xor esi, edi sub edx, esi dec [esp+240h+var_220] jnz short loc 401688</pre>	Nov Nov Rov	edx, esi [csp-240h+var_22C], 20h [csp-240h+var_22C], 20h z 1668: ebx, [csp+240h+var_228] esi, edx esi, 5 edi, edx edi, edx edi, edx edi, [csp-240h+pcbBuffer] edi, 3 edi, [cbx+edi+4] edi, [cbx+240h+pcbBuffer], 61C88646h
<pre>nov esi, eax shr esi, 5 nov edi, eax shl edi, 4 xor esi, edi nov edi, [esp+248h+pcbBuffer] and edi, 3 nov edi, [ebx+edi+4] add esi, eex xor esi, edi sub edx, esi dec [esp+248h+var_220] jnz short loc_481688</pre>	nov nov nov loc_44 nov shr nov sh1 xor nov sh1 xor add sub	edx, esi [esp-240h+pcbBuffer], 3910CSCOh [esp+240h+var_22C], 20h
nov edi, eax shl edi, 4 xor esi, edi nov edi, [esp+240h+pcbBuffer] and edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add edi, [esp+240h+pcbBuffer] add esi, eax xor esi, eax you edi, esi dec [esp+240h+var_22C] jnz short loc_40f688	nov nov nov	edx, esi [csp-240h+pcbBuffer], 3910CSC0h [csp-240h+var_22C], 20h z z b16B8: ebx, [csp+240h+var_228] esi, edx esi, edx edi, edx edi, 4 esi, edi edi, [csp+240h+pcbBuffer] edi, 3 edi, [csp+240h+pcbBuffer] [csp+240h+pcbBuffer] [csp+240h+pcbBuffer], 61C88646h esi, edx esi, edi
<pre>xor esi, edi nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [esp+240h+pcbBuffer] add esi, [esp+240h+pcbBuffer] add esi, eax xor esi, edi sub edx, esi dec [esp+240h+var_22C] jnz short loc_401688</pre>	nov nov nov nov loc_44 nov shr nov shr nov shr and nov str and nov str and nov	edx, esi [esp+240h+var_22C], 20h [esp+240h+var_22C], 20h ebx, [esp+240h+var_22B] esi, edx esi, 5 edi, edx edi, edx edi, edx edi, [esp+240h+vcbBuffer] edi, [esp+240h+pcbBuffer] edi, [esp+240h+pcbBuffer] edi, [esp+240h+pcbBuffer] [esp-240h+pcbBuffer], 61C88646h esi, edx eax, esi eax
<pre>nov edi, [esp+240h+pcbBuffer] and edi, 3 nov edi, [ebx+edi+4] add edi, [eax240h+pcbBuffer] add esi, eax xor esi, edi sub edx, esi dec [esp+240h+var_22C] jnz short loc.401688</pre>	nov nov nov loc_44 nov shr nov shr nov shr nov shr nov shr nov shr nov shr nov shr nov shr nov shr nov shr nov	edx, esi [esp+240h+var_22C], 20h [esp+240h+var_22C], 20h = 11688: ebx, [esp+240h+var_228] esi, edx esi, 5 edi, edx edi, edx edi, edx edi, [esp+240h+vcbBuffer] edi, [esp+240h+pcbBuffer] edi, [esp+240h+pcbBuffer] edi, [esp+240h+pcbBuffer] [esp-240h+pcbBuffer], 61C88646h esi, edx esi, edx esi, eax esi, 5 edi, eax
nov edi, [ebx+edi+4] add edi, [esp+240h+pcbBuffer] add esi, eax xor esi, edi sub edx, esi dec [esp+240h+var_22C] jnz short loc_401688 w	nov nov nov nov nov shr nov shr nov shr nov shr and add add xor sub nov shr sub shr	edx, esi [esp+240h+var_22C], 20h [esp+240h+var_22C], 20h
add edi, [esp+240h+pcbBuffer] add esi, eax xor esi, edi sub edx, esi dec [esp+240h+var_22C] jnz short loc_401688	nov nov nov nov loc_46 nov shr nov shr and sub add sub add sub shr nov shr nov add sub	edx, esi [esp-240h+pcbBuffer], 3910CSC0h [esp-240h+var_22C], 20h
xor es1, ed1 sub edx, es1 dec [esp+240h+var_22C] jnz short loc 401688 mov [ecx], edx mov [ecx+1], eax add ecx, 8 dec [esp+240h+var_220]	nov nov nov nov loc_46 nov shr nov shr and add sta add sta shr sub shr sov shr nov add sta add sta shr nov add shr nov add shr nov add shr nov add add shr nov add add shr nov add add add add add add add add add ad	edx, esi [esp-240h+var_22C], 20h [esp-240h+var_22C], 20h z z z z z z z z z z z z
dec [esp+240h+var_22C] inz short loc_401688 mov [ecx], edx mov [ecx+1], eax add ecx, 8 dec [esp+240h+var_220]	nov nov nov loc_44 mov shr mov shr mov shr and add sub add sub add xor sub add xor sub add xor sub add	edx, esi [esp+240h+var_22C], 20h [esp+240h+var_22C], 20h
nov [ecx], edx nov [ecx+1], eax add ecx, 8 dec [esp+240h+var 220]	nov nov nov loc_40 mov shr nov shr and add sub add sub add sub add sub add xor shr nov shr and nov shr add add add add add	edx, esi [esp+240h+var_22C], 20h [esp+240h+var_22C], 20h z 11688: ebx, [esp+240h+var_228] esi, edx esi, edx edi, [esp+240h+var_228] edi, edx edi, edx edi, edx edi, [esp+240h+pcbBuffer] edi, [esp+240h+pcbBuffer] (esp+240h+pcbBuffer] (esp+240h+pcbBuffer] (esi, edi eax, esi esi, edi eax, esi esi, eax esi, eax esi, eax esi, eax esi, edi edi, [esp+240h+pcbBuffer] edi, [es
w → ⊂ nov [ecx], edx nov [ecx+b], eax add ecx, 8 dec [esp+240h+var 220]	nov nov nov nov loc_44 nov shr nov shr and add xor sub add xor sub shr nov add xor shr sub shi xou shr and add xor sub	edx, esi [esp-240h+vcbBuffer], 3910CSC0h [esp-240h+var_22C], 20h
nov [ecx], edx nov [ecx+4], eax add ecx, 8 dec [esp+240h+var 220]	nov nov nov loc_40 mov Shr mov Shr mov Shr mov Shr mov Shr sub add xor Sub stub add xor Shr mov add add add add add add add	edx, esi [esp-240h+var_22C], 20h [esp-240h+var_22C], 20h [esp-240h+var_22C], 20h [esp-240h+var_22B] esi, edx esi, edx esi, edx esi, edx edi, edx edi, edx edi, [esp+240h+vcbBuffer] edi, [ebx+edi=44] edi, [ebx+edi=44] edi, [ebx+edi=44] edi, [esp-240h+pcbBuffer] (esi, edx esi, edx esi, edx esi, edx esi, edx edi, [esp+240h+pcbBuffer] edi, [ebx+edi=44] edi, [ebx+edi=44] e
nov [ecx+4], eax add ecx, 8 dec [esp+240h+var_220]	nov nov nov nov nov souther shr and sub add sub add sub add sub add sub add sub shr anov shr anov add sub	edx, esi [esp+240h+vor_22C], 20h [esp+240h+vor_22C], 20h
add ecx, 8 dec [esp+240h+var_220]	nov nov nov loc_4ti nov shr nov shr and add xor sub add xor sub add xor sub add xor sub shi xou add xor sub shi xou add xor shr and add xou shi xou shi xou add xou shi xou shi xou add xou shi xou shi xou add xou shi xou shi xou add xou shi xou shi xou add xou shi xou shi xou add xou shi xou shi xou add xou shi xou xou xou shi xou xou shi xou xou xou xou xou xou xou xou xou xou	edx, esi [esp-240h+var_22C], 20h [esp-240h+var_22C], 20h = = = = = = = = = = = = =
jnz loc_401634	nov nov nov loc_40 nov shr nov shr and sub add xor sub add xor sub shr nov add add xor sub shr mov add add xor sub shr mov add add xor sub shr mov add add xor sub shr mov add add xor sub shr mov add add xor sub shr mov add add xor sub shr mov add add xor sub shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add xor shr mov add add add xor shr mov add add add xor shr mov add add add add add add add add add ad	edx, esi [esp-240h+var_22C], 20h [esp-240h+var_22C], 20h esi, edx esi, edx esi, edx esi, edx edi, edx edi, edx edi, edx edi, edx edi, edx edi, [esp+240h+pcbBuffer] edi, [ebx+edi=#4] edi, [ebx+edi=#4] edi, [esp-240h+pcbBuffer] (esi, edx esi, edx esi, edx esi, edx ex, esi esi, edx ex, esi esi, edx ex, esi esi, edx ex, esi ex, edi ex, esi ex, edi ex, esi ex, edi ex, esi ex, esi (esp-240h+pcbBuffer] ex, esi (esp-240h+pcbBuffe
	nov nov nov loc_40 nov shr nov shr and sub add xor sub add xor sub shr nov add add xor sub shr mov add add xor sub shr mov add add xor sub shr nov add add xor sub shr nov add add xor sub shr nov add add xor sub shr nov add add xor sub shr nov add add xor sub shr nov add add xor sub shr nov add add xor sub shr nov add add xor sub shr nov add add xor add add xor add add xor add add xor add add add xor sub shr nov add add xor sub shr nov add add xor sub shr nov add add xor sub shr sub shr add add xor add add add xor add add xor add add add xor add add add add add add add add add ad	edx, esi [esp-240h+vor_22C], 20h [esp-240h+vor_22C], 20h ebx, [esp+240h+vor_22C], 20h ebx, [esp+240h+vor_22B] esi, edx esi, edx edi, edx edi, edx edi, edx edi, [esp+240h+pcbBuffer] edi, [esp+240h+pcbBuffer] edi, [esp+240h+pcbBuffer] [esp-240h+pcbBuffer], 61C88646h esi, edx esi, edi eax, esi esi, exi esi, edi edi, [esp+240h+pcbBuffer] edi, [esp+240h+p
	nov nov nov loc_4f nov shr nov shr and add xor sub add xor sub add xor sub add xor sub add xor sub add xor sub add xor shr anov add add xor shr sub add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add xor shr anov add add add xor shr anov add add add xor shr anov add add add xor shr anov add add add xor shr anov add add add xor add add add add add add add add add ad	edx, esi [esp-240h+vor_22C], 20h [esp-240h+vor_22C], 20h = = = = = = = = = = = = =



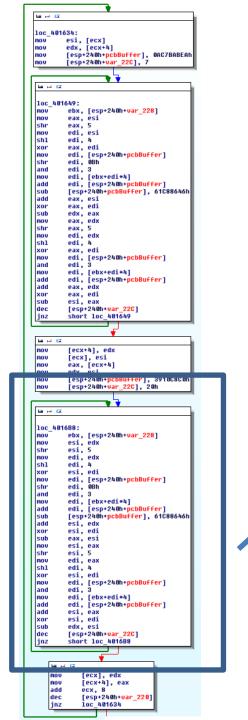
mov	[esp+240h+var_22C], 7	
🔛 🖂 🗵	2	
1oc_40	1649 :	
mov	ebx, [esp+240h+var 228]	
mov	eax, esi	
shr	eax, 5	
mov	edi, esi	
shl	edi, 4	
xor	eax, edi	
mov	edi, [esp+240h+pcbBuffer]	
shr	edi, OBh	
and	edi, 3	
mov	edi, [ebx+edi*4]	
add	edi, [esp+240h+pcbBuffer]	
sub	[esp+240h+pcbBuffer], 61C88646h	
add	eax, esi	
xor	eax, edi	
sub	edx, eax	
mov	eax, edx	
shr	eax, 5	
mov	edi, edx	
shl	edi, 4	
xor	eax, edi	
MOV	edi, [esp+240h+pcbBuffer]	
and	edi, 3	
MOV	edi, [ebx+edi*4]	
add	edi, [esp+240h+pcbBuffer]	
add	eax, edx	
xor	eax, edi	
sub	esi, eax	
dec	[esp+240h+var_22C]	
jnz	short loc_401649	

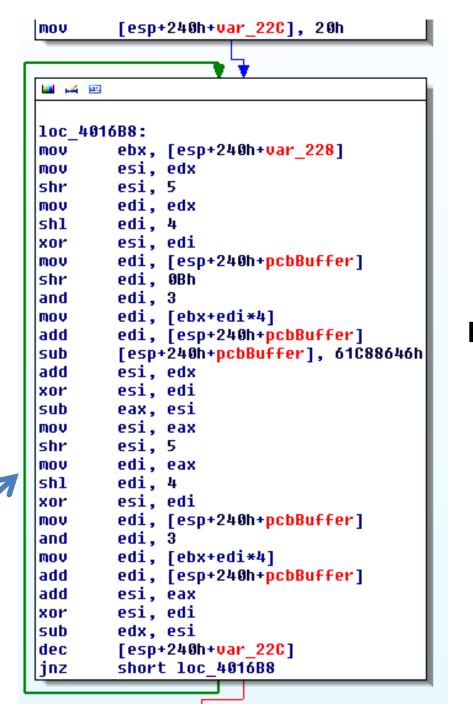
LOOP 1



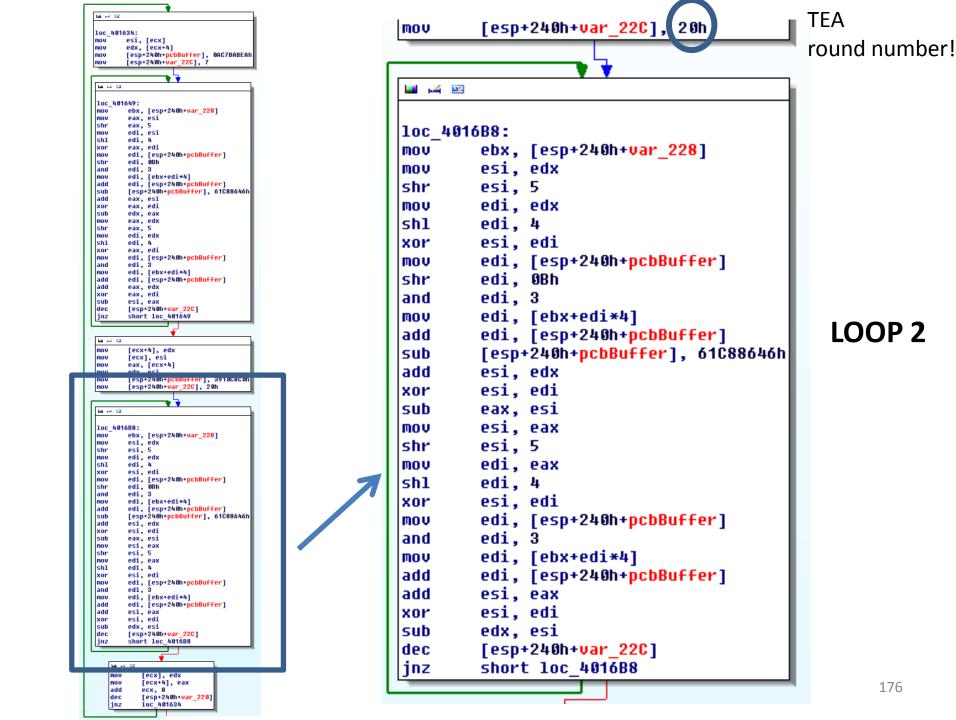


	•
₩ •4 ⊠	
	1634:
nov	esi, [ecx] edx, [ecx+4]
nov	[esp+240h+pcbBuffer], 0AC7BABEAh
nov	[esp+240h+var_22C], 7
₩ µ4 E	*
	1649:
nov	ebx, [esp+240h+var_228] eax, esi
shr	eax, 5
nov shl	edi, esi edi, 4
xor	eax, edi
nov shr	edi, [esp+240h+ <mark>pcbBuffer</mark>] edi, 0Bh
and nov	edi, 3 edi, [ebx+edi*4]
add	edi, [esp+240h+pcbBuffer]
sub add	[esp+240h+pcbBuffer], 61C88646h eax, esi
xor	eax, edi
sub nov	edx, eax eax, edx
shr	eax, 5
nov shl	edi, edx edi, 4
xor	eax, edi
and	edi, 3
nov add	edi, [ebx+edi*4] edi, [esp+240h+ <mark>pcbBuffer</mark>]
add	eax, edx
xor sub	eax, edi esi, eax
dec	[esp+240h+var_22C]
jnz	short loc_401049
	z [ecx+4], edx
nov	[ecx+4], edx [ecx], esi eax, [ecx+4]
nov	eax, [ecx+4] edx, esi
nov	[esp+240h+pcbBuffer], 3910C8C0h
nov	[esp+240h+var_22C], 20h
	* *
■ µ4 E	·
	1688:
nov	ebx, [esp+240h+ <mark>var_228</mark>] esi, edx
shr	esi, 5
nov shl	edi, 4
xor nov	esi, edi edi, [esp+240h+ <mark>pcbBuffer</mark>]
shr	edi, OBh
and nov	edi, 3 edi, [ebx+edi*4]
add	edi, [esp+240h+pcbBuffer]
sub add	[esp+240h+pcbBuffer], 61C88646h esi, edx
xor sub	esi, edi eax, esi
nov	esi, eax
shr nov	esi, 5 edi, eax
shl	edi, 4
xor	esi, edi edi, [esp+240h+ <mark>pcbBuffer</mark>]
and	edi, 3
add	edi, [ebx+edi*4] edi, [esp+240h+ <mark>pcbBuffer</mark>]
add xor	esi, eax esi, edi
sub	edx, esi
dec jnz	[esp+240h+var_22C] short loc_4016B8
no	v [ecx], eux
no	v [ecx+4], eax
no no ad	v [ecx+4], eax d ecx, 8 c [esp+240h+var 220]
no no ad	v [ecx+4], eax d ecx, 8 c [esp+240h+var_220]

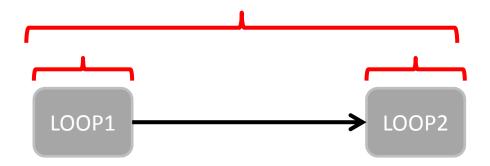




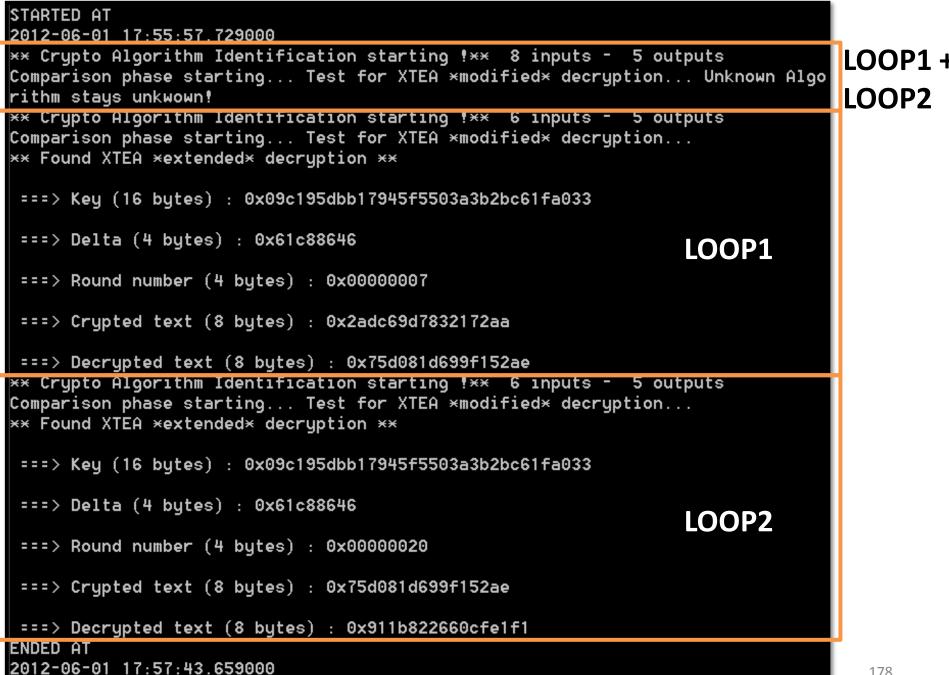
LOOP 2



Loop Data Flow Graph



3 possible cryptographic algorithms



Method Recap

- 1. We collect an execution trace.
- 2. We extract possible cryptographic algorithms with their parameter values.
- 3. We compare the input-output relationship with known algorithms.

We prove that a program behaves like a known crypto algorithm during one particular execution path.

Conclusion (1)

- Interesting alternative to pure syntacticidentification for crypto algorithms:
 - Resistance against usual obfuscation techniques.
 - Gives the exact parameters.
- As any dynamic technique, you have to know how to exhibit interesting execution paths.
- It is easy to bypass, like any program analysis technique ☺

Conclusion (2)

- The identification process itself is generic:
 - Collect the execution trace
 - Extract the type of code you are looking for (here is the magic)
 - Get I/O values
 - Compare with reference implementations

Conclusion (3)

 Nice work: Felix Gröbert "Automatic Identification of Cryptographic Primitives in Software", 27th CCC

http://code.google.com/p/kerckhoffs/

What's Next ? (1)

• That's only the beginning! Just wanted to show that it is feasible and useful.

 What about more complex algorithms ? What about hash functions ? Compression algorithms ?

• What about proprietary algorithms ?

What's Next ? (2)

• Make a real tool. This one is just a PoC.

 How to use the analyst knowledge ? In practice he often knows where the crypto is, analyzing a complete execution trace is more an academic hobby.

Thank you for your attention ;-)

j04n.calvet@gmail.com