

# The story of the "Uncrackable" Lockbox, and Why Hackers Need to Work Alongside Developers

Matthew Ruffell  
Chcon 2019

# Challenge #1



Posted by u/cryptocomicon 20 hours ago

32



## Hand off your digital assets, even if you are no longer around.

Over the years I've seen many people wondering how they can transfer ownership of their digital assets in the future. They don't want to give a loved one a copy of their wallet seed, but they do want to make sure that no matter what, those assets are made available at a date in the future.

TimeLock is a tool which I have developed to solve this problem, and probably many other problems which I am not aware of. The free version allows you to protect a file of up to 10KB with an un-hackable time lock, synced to the Bitcoin Network.

I'm so confident in this technology that I've created a challenge LockBox file which holds the private key to an address with 0.02 BTC.

Please give it a try.

More information at [algomachines.com](http://algomachines.com)

Link to the challenge: [challenge](#)



# Information and Scope

- We are given:
  - Password - “TimeLock”.
  - Answers to questions - “0.02”.
  - Time range – Start and end, both UTC.
- This limits scope to time lock mechanism only.



# Reconnaissance

[about](#)[download](#)[tutorial](#)[contact](#)

## about

Securely lock your data until a time you choose.

Create LockBoxes up to 10KB.

Un-crackable TimeLock synced to the Bitcoin Network.

Retain privacy. Your data stays on your computer and nowhere else.

Distribute your time locked LockBox to whomever you wish.

## technical information

Project files are encrypted using a hash of the project password and the contents of a random file generated on install.

LockBox files contain encrypted:

- Questions
- Delay and TimeLock information
- Data file

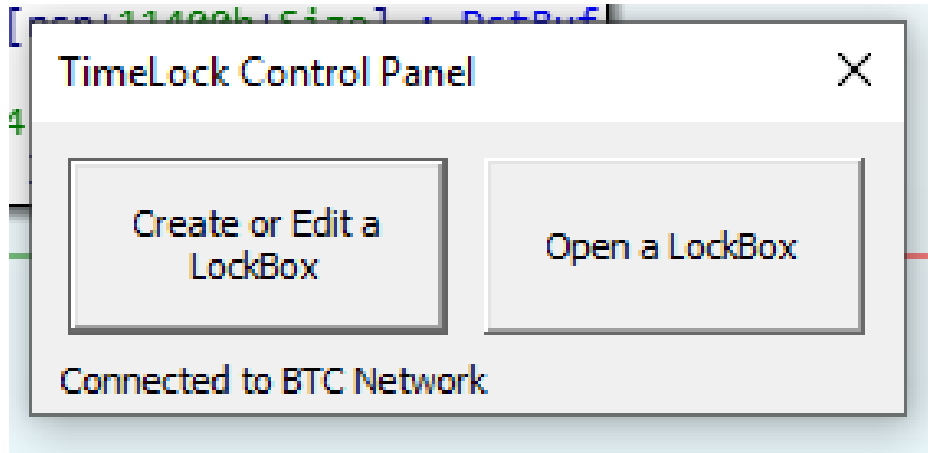
Encryption algorithm requires a 1 GB memory buffer.

CPU clock is continuously checked against BTC Network time.

## Files are encrypted with:

- Hash of password
- Random generated file
- No mention of Time?

# Running TimeLock



The screenshot shows a window titled "LockBox Creator" with a close button (X) in the top right corner. The window contains several input fields and buttons. At the top, there are three fields: "LockBox Name", "Data File", and "Password". Below these is a section titled "Questions and Answers" with five rows, each containing a question field (Q1-Q5) and an answer field (A1-A5). Below the questions is a "Delay" field with a label "minutes after asking questions." At the bottom, there are four fields for availability: "Make available on or after date", "Make available on or after time on above date", "Revoke availability on or after date", and "Revoke availability on or after time on above date". Each field has a date or time input and a format label. At the very bottom, there are five buttons: "Info", "Open", "Test", "Save", and "Generate LockBox", followed by a "Quit" button on the right.

# Unlocking a LockBox

Password


OK Cancel

Answer Question

Private key unlocks how many BTC?

Next Cancel

TimeLock

 28 days 1 hours 2529 sec until data is available

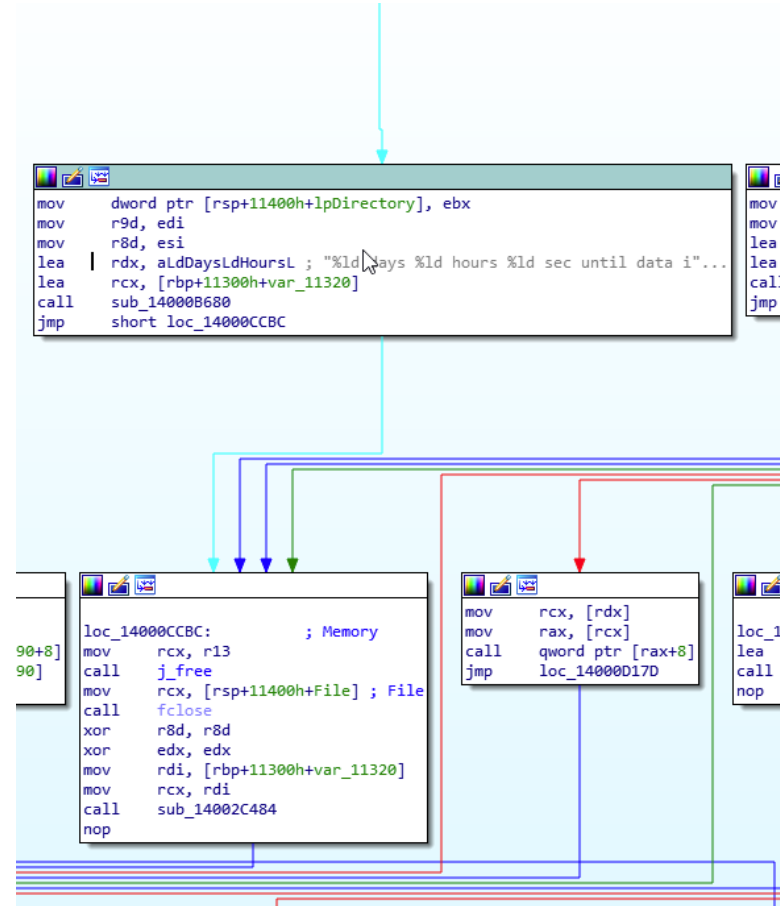
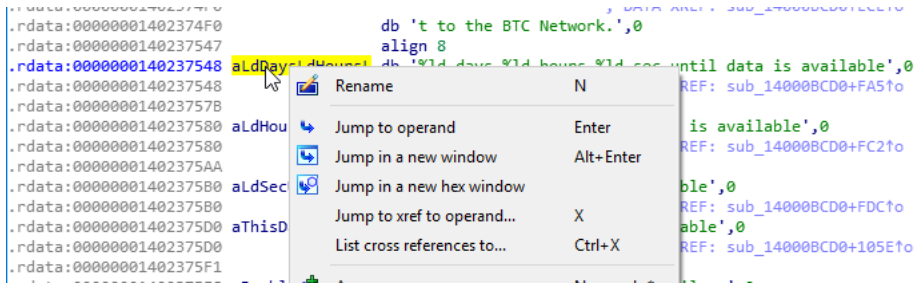
OK

# Analysing Strings

Address	Length	Type	String
.rdata:0000...	0000000D	C	ControlPanel
.rdata:0000...	00000021	C	BTC Network %ld of 8 connections
.rdata:0000...	00000057	C	Clock on this computer does not match BTC Network time, must be within +/- 10 minutes.
.rdata:0000...	00000019	C	Connected to BTC Network
.rdata:0000...	0000000A	C	LockBoxes
.rdata:0000...	0000001C	C	Select a LockBox file (*.x)
.rdata:0000...	0000001E	C	Unable to open LockBox file:
.rdata:0000...	00000022	C	Problem #1 reading LockBox file:
.rdata:0000...	0000002A	C	Incorrect password or invalid LockBox #1.
.rdata:0000...	0000002A	C	Incorrect password or invalid LockBox #2.
.rdata:0000...	00000022	C	Problem #2 reading LockBox file:
.rdata:0000...	0000001A	C	Corrupt LockBox file #1:
.rdata:0000...	0000000A	C	Canceled.
.rdata:0000...	00000022	C	Problem #3 reading LockBox file:
.rdata:0000...	00000013	C	Incorrect answers.
.rdata:0000...	0000001E	C	No connection to BTC Network.
.rdata:0000...	0000004B	C	Number of required connections to BTC Network=%ld, connections we have=%ld
.rdata:0000...	00000057	C	Clock on this computer is off by more than 10 minutes with respect to the BTC Network.
.rdata:0000...	00000033	C	%ld days %ld hours %ld sec until data is available
.rdata:0000...	0000002A	C	%ld hours %ld sec until data is available
.rdata:0000...	00000020	C	%ld sec until data is available
.rdata:0000...	00000021	C	This data is no longer available
.rdata:0000...	00000022	C	Problem #4 reading LockBox file:
.rdata:0000...	00000036	C	Can't allocate enough memory to load file to memory:
.rdata:0000...	00000022	C	Problem #5 reading LockBox file:
.rdata:0000...	00000028	C	Select folder where %s will be created.
.rdata:0000...	00000021	C	You elected not to save file: %s
.rdata:0000...	00000014	C	Can't create file:
.rdata:0000...	00000024	C	Reveal file %s in Windows Explorer?
.rdata:0000...	0000000D	C	/select,\"%s\"
.rdata:0000...	0000000D	C	explorer.exe
.rdata:0000...	00000005	C	open
.rdata:0000...	0000000C	C	DelaySecDlg
.rdata:0000...	00000008	C	%ld sec
.rdata:0000...	00000013	C	--disable-timesync
.rdata:0000...	0000000D	C	AlgoMachines
.rdata:0000...	0000001D	C	Unable to create directory:
.rdata:0000...	00000009	C	TimeLock
.rdata:0000...	00000006	C	e.bin
.rdata:0000...	0000001E	C	Can't open file for reading:
.rdata:0000...	00000012	C	File is invalid:
.rdata:0000...	00000017	C	Problem reading file:
.rdata:0000...	00000012	C	BTC_peer_list.bin

- “Incorrect answers”
- “%ld days %ld hours %ld sec until data is available”
- “Select folder where %s will be created”
- “Reveal file %s in Windows Explorer?”

# Xref Strings – Hackers Best Friend





loc\_14000CBCC: ; Time  
xor ecx, ecx  
call \_time64  
mov rbx, [rbp+11300h+var\_78]  
cmp rbx, rax  
jbe loc\_14000CD03

sub rbx, rax  
mov rax, 0C22E450672894AB7h  
mul rbx  
mov rsi, rdx  
shr rsi, 10h  
test rsi, rsi  
jz short loc\_14000CC09

imul rax, rsi, 15180h  
sub rbx, rax

xor ecx, ecx ; Time  
call \_time64  
cmp rax, [rbp+11300h+var\_70]  
jbe short loc\_14000CD3A

loc\_14000CC09:  
mov rax, 23456789ABCD0F13h  
mul rbx  
mov rdi, rbx  
sub rdi, rdx  
shr rdi, 1  
add rdi, rdx  
shr rdi, 08h  
test rdi, rdi  
jz short loc\_14000CC35

loc\_14000CD03:  
cmp [rbp+11300h+var\_70], rbx  
jbe short loc\_14000CD3A

# The “true” Path Leads to File Writing

```
loc_1400CE72:
mov     r8, [rbp+11300h+var_112D8]
lea     rdx, aSelectFolderWh ; "Select folder where %s will be created."
lea     rcx, [rbp+11300h+var_11360]
call    sub_1400B680
mov     rdx, [rbp+11300h+var_11360]
lea     rcx, [rbp+11300h+var_11370]
call    sub_14000AEA0
test     eax, eax
jnz     loc_1400CF38
```

```
lea     rcx, [rbp+11300h+var_11318] ; File
call    fopen_s
mov     r9, [rbp+11300h+var_11318] ; File
test     r9, r9
jnz     short loc_1400CFA6
```

```
loc_1400CFA6:
mov     rax, [rbp+11300h+var_112D8]
mov     ecx, r12d
test     rax, rax
jz      short loc_1400CFCB
```

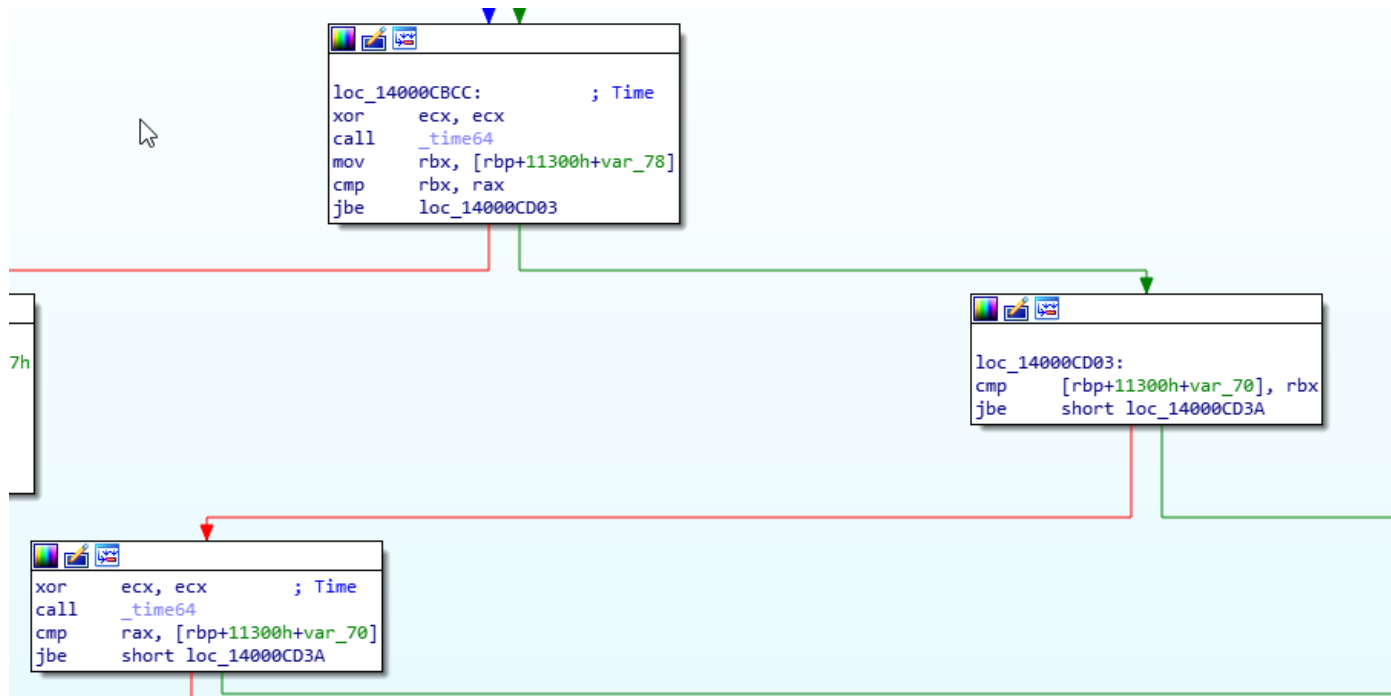
```
cmp     [rax], c1
jz      short loc_1400CFCB
```

```
db      66h, 66h
nop     word ptr [rax+rax+00000000h]
```

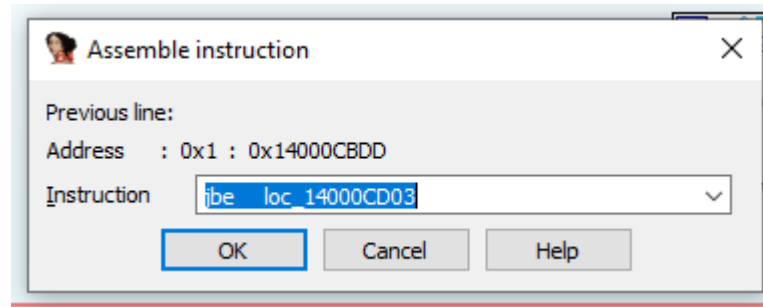
```
loc_1400CFC0:
inc     ecx
lea     rax, [rax+1]
cmp     byte ptr [rax], 0
jnz     short loc_1400CFC0
```

```
loc_1400CFCB:
inc     ecx
mov     eax, dword ptr [rsp+11400h+Size]
sub     eax, ecx
mov     edi, eax
add     rcx, rbx ; Str
mov     r8d, eax ; Count
mov     edx, 1 ; Size
call    fwrite
mov     r8d, dword ptr [rsp+11400h+Size] ; Size
```

# TimeLock Mechanism == If Statements

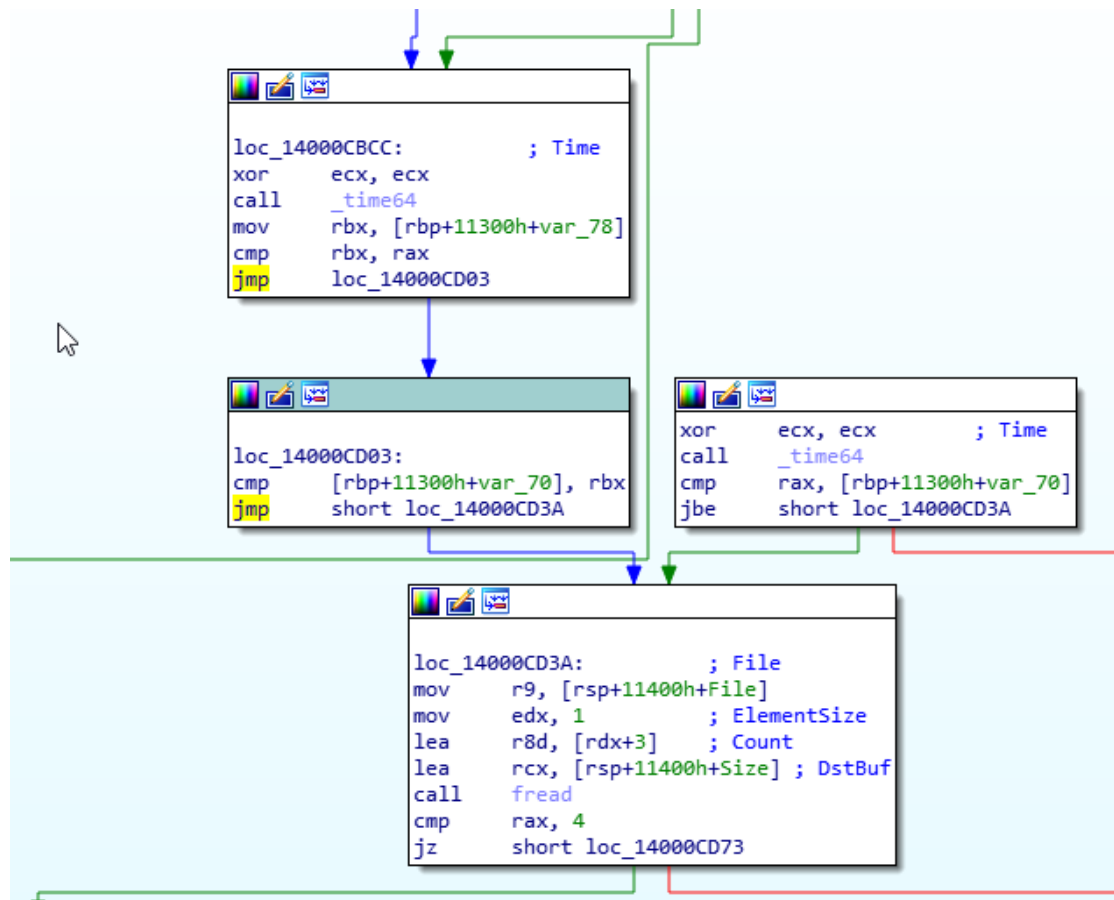


# What Happens If We Patch It Out?

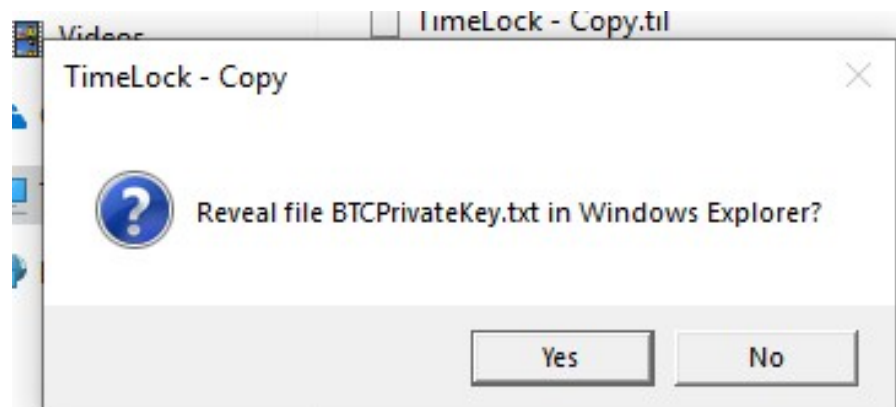
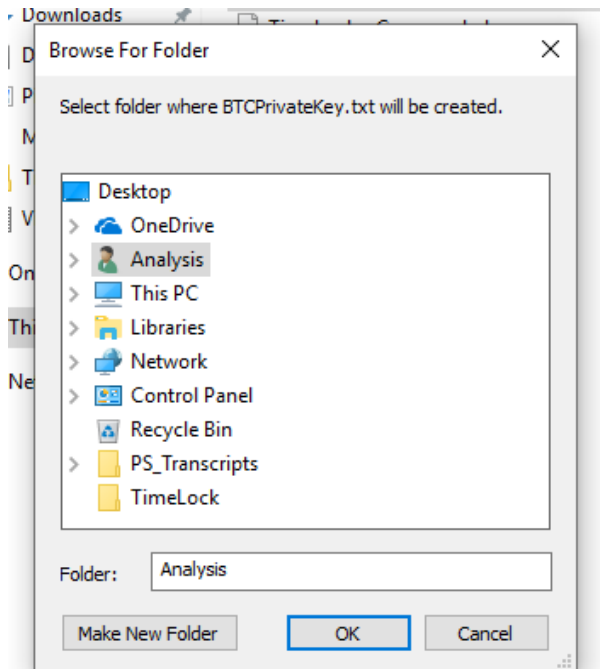


Patch jbe -> jmp

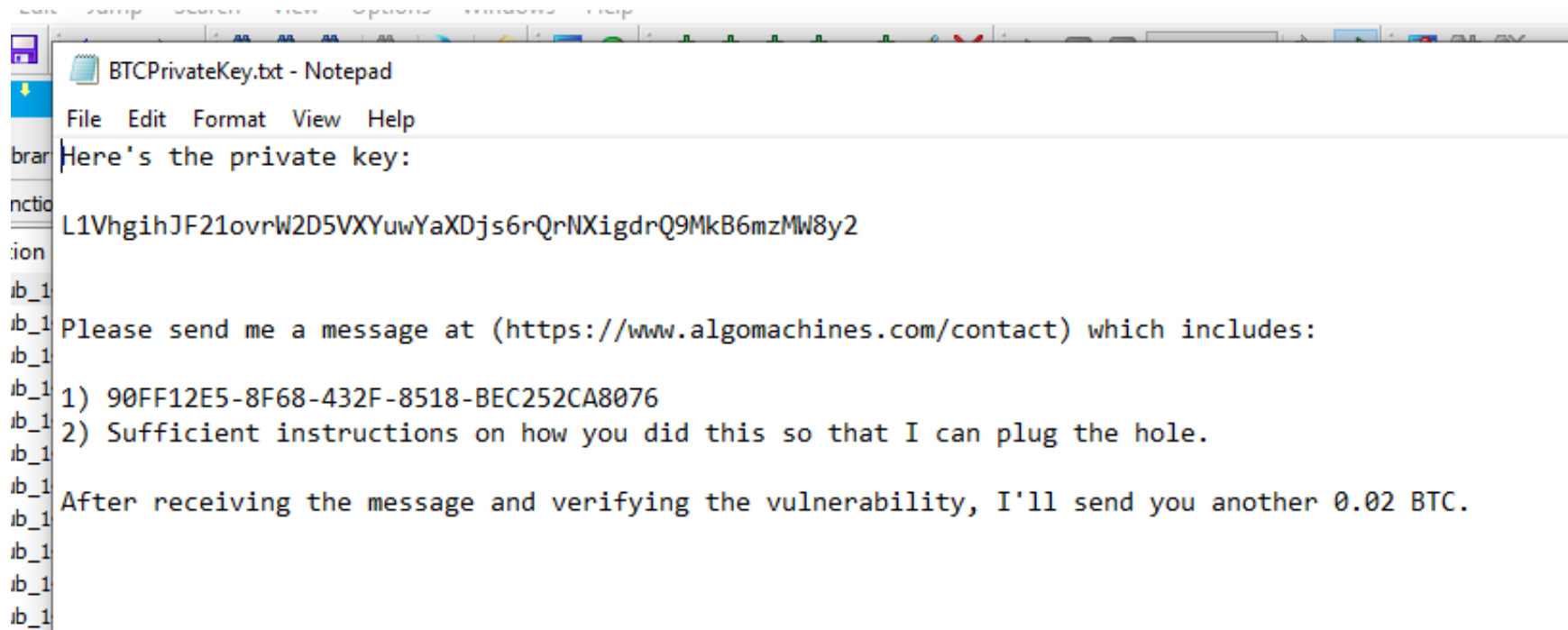
# Patched Logic



# Looks Promising...



# Loot #1



```
File Edit Format View Help
Here's the private key:

L1VhgiHJF21ovrW2D5VXYuwYaXDjs6rQrNXigdrQ9MkB6mzMw8y2

Please send me a message at (https://www.algomachines.com/contact) which includes:

1) 90FF12E5-8F68-432F-8518-BEC252CA8076
2) Sufficient instructions on how you did this so that I can plug the hole.

After receiving the message and verifying the vulnerability, I'll send you another 0.02 BTC.
```

# Lessons Learned:

Vulnerability:

- Checks placed after decryption is too late.

How to Fix:

- Time is a secret, and needs to be involved in the encryption process.
- Executables cannot be a root of trust.
- Key derivation should be handled by a third party.



# Challenge #2



Posted by u/cryptocomicon 6 days ago

2



## Hand off your digital assets, even if you are no longer around (TimeLock V1.2 challenge)

Over the years I've seen many people wondering how they can transfer ownership of their digital assets in the future. They don't want to give a loved one a copy of their wallet seed, but they do want to make sure that no matter what, those assets are made available at a date in the future.

TimeLock is a tool which I have developed to solve this problem, and probably many other problems which I am not aware of. The free version allows you to protect a file of up to 10KB with an un-hackable time lock, synced to the Bitcoin Network.

I'm so confident in this technology that I've created a challenge LockBox file which holds the private key to an address with 0.02 BTC.

Please give it a try.

NOTE: This is going to be much harder than last time.

More information at [algomachines.com](http://algomachines.com)

Link to the challenge: [challenge](#)

Here's a link to the Creator screen for this lock box: [Creator](#). This shows you the available time period for the lock box. I'm also giving you the password and the answer to the one question... much more information than you would have if you stumbled upon this file and wanted to crack it.

Here's a link to the TimeLock V1.0 challenge thread: [Challenge #1](#)

# What Changed?

↑ security\_researcher redditor for 1 week 1 point · 5 days ago  
↓ I'm looking forward to seeing what has changed. I'll give it a go tonight, and I'll let you know how I get on. Can I ask what you changed or is this meant to be a surprise?

■ Reply Share Save Edit ...

↑ cryptocomicon 1 point · 5 days ago · edited 5 days ago  
↓ This can't be cracked by stepping over some logic. The data file is encrypted with the time lock data, not just the password and answer.

The following data are required in order to crack the LockBox: Password, Answers, Available time range. However, none of these items are stored in the LockBox. (BTW I'm giving you all of this data for Challenge2.x)

The only way that the program knows that you have entered the right password and answers is if decrypted data is validated. The only way that the program knows if the BTC network time is inside of the required interval is if the data file decryption (using current BTC network time) is validated.

Data file decryption workload is programmable, can be made quite costly. Even if you have the source code for the program, the password and the answers, choosing a very high encryption cycle count for your lock box (and a narrow available time interval) can make cracking a lock box very costly.

Encryption cycle count of 10 for Challenge2.x means that about 20 seconds of CPU time will be required to decrypt.

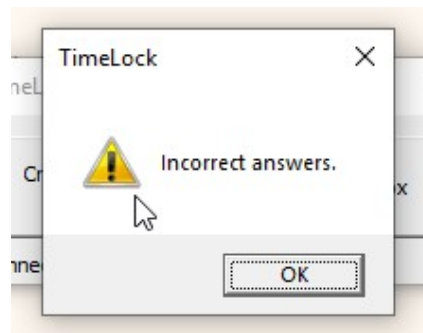
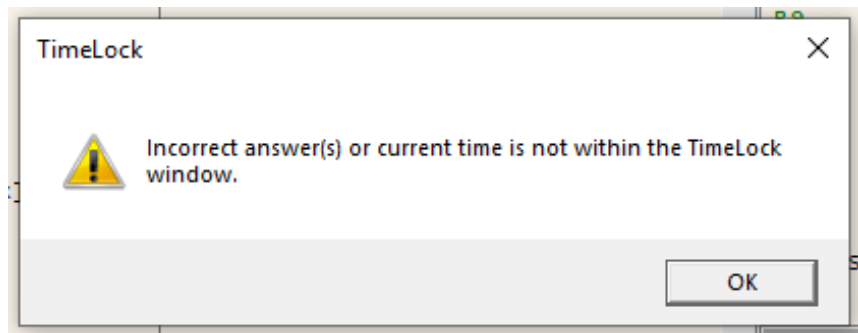
■ Reply Share Report Save Give Award

- “...encrypted with time lock data, not just password and [question] answer”
- Decrypted data must now be “validated”.
- Validation implies correct BTC network time.

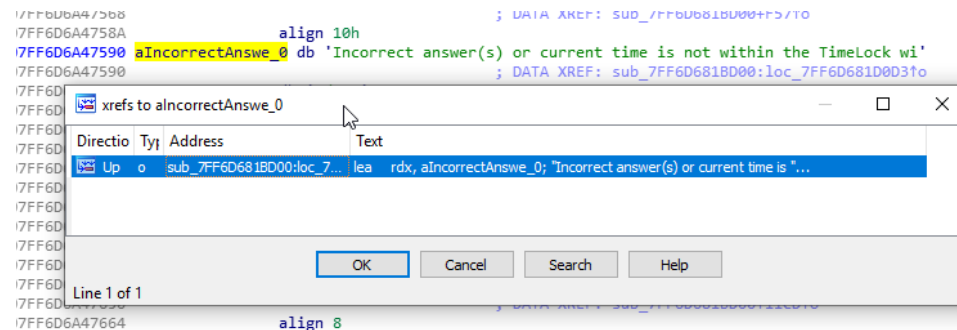
# Plan of Attack

- Locate where the time is passed into decryption function, set it to future.
- We know what the times are. Keep an eye out for:
  - 22/02/2019 00:00 UTC becomes **1550793600**. Hex: **0x5C6F3B80**
  - 23/02/2019 00:00 UTC becomes **1550880000**. Hex: **0x5C708D00**

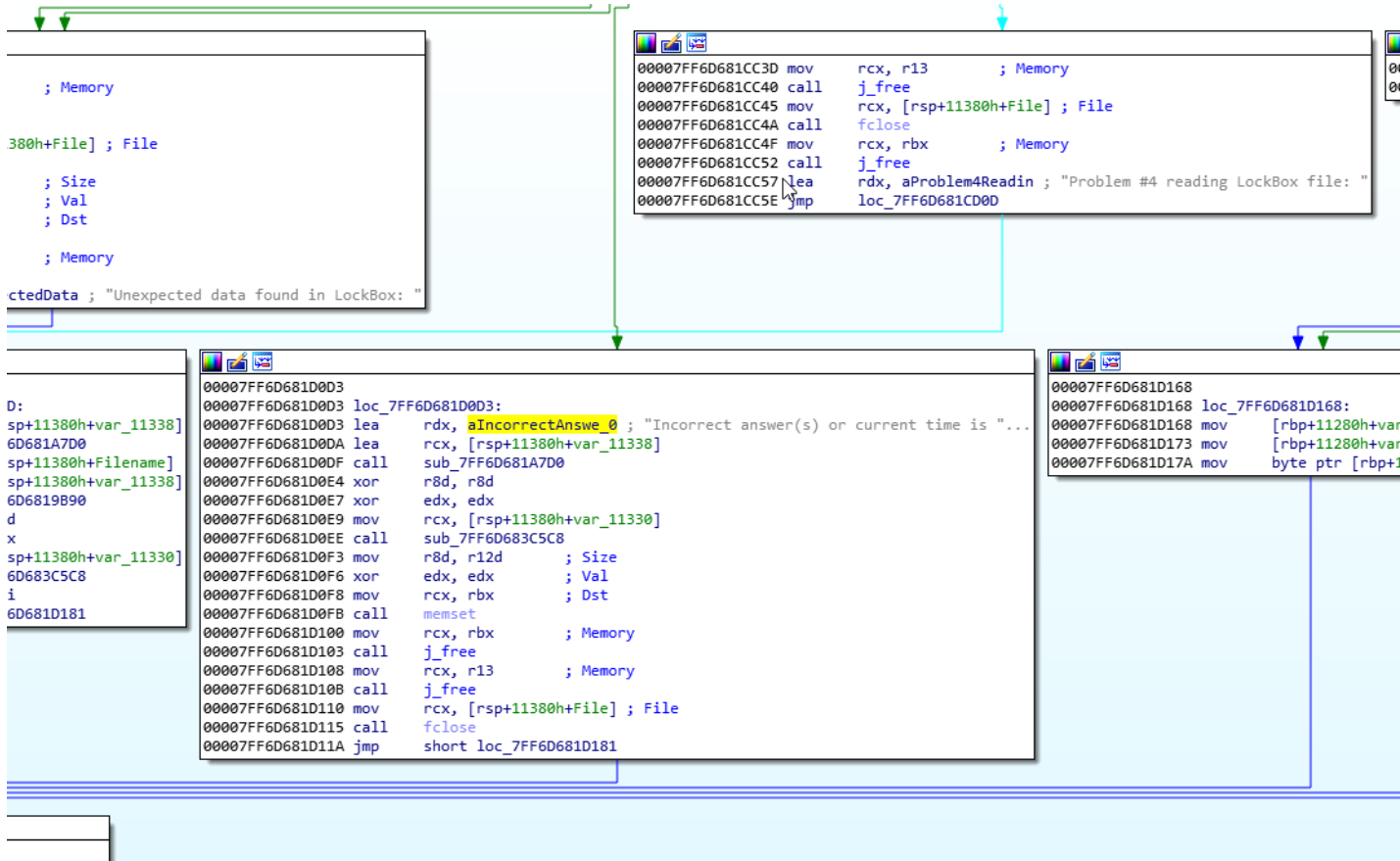
# Opening Lockboxes



Address	Length	Type	String
[s] .rdata:0000...	00000012	C	AnswerQuestionDlg
[s] .rdata:0000...	00000013	C	Incorrect answers.
[s] .rdata:0000...	00000047	C	Incorrect answer(s) or current time is not within the TimeLock window.
[s] .rdata:0000...	000000E2	C	The project may include one data file, which will be revealed when a user answers all qu...
[s] .rdata:0000...	000000A9	C	The project may include between one and five questions and answers. In order to acc...



# Jumping to String



# Looking Upwards

```
00007FF6D681CC63
00007FF6D681CC63 loc_7FF6D681CC63:
00007FF6D681CC63 mov     [rsp+11380h+var_11348], 1
00007FF6D681CC68 mov     [rsp+11380h+nShowCmd], 40000000h
00007FF6D681CC73 mov     [rsp+11380h+lpDirectory], r13
00007FF6D681CC78 mov     r9d, 100h
00007FF6D681CC7E lea     r8, [rbp+11280h+var_130]
00007FF6D681CC85 mov     edx, r12d
00007FF6D681CC88 mov     rcx, rbx
00007FF6D681CC8B call    sub_7FF6D68183A0
00007FF6D681CC90 xor     esi, esi
00007FF6D681CC92 mov     ecx, esi
00007FF6D681CC94 mov     eax, esi
```

```
00007FF6D681CC96
00007FF6D681CC96 loc_7FF6D681CC96:
00007FF6D681CC96 cmp     [rax+rbx], sil
00007FF6D681CC9A jnz     loc_7FF6D681D0D3
```

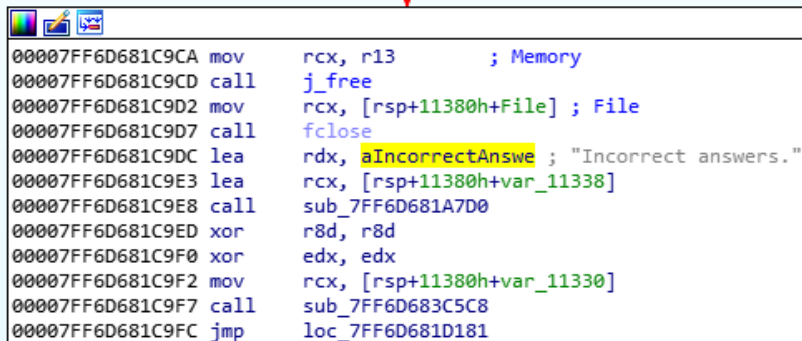
To: sub\_7FF6D681BD00:loc\_7FF6D681D0D3

```
00007FF6D681D0D3
00007FF6D681D0D3 loc_7FF6D681D0D3:
00007FF6D681D0D3 lea     rdx, aIncorrectAnsw_0 ; "Incorrect answer(s) or current time is ..."
00007FF6D681D0DA lea     rcx, [rsp+11380h+var_11338]
00007FF6D681D0DF call    sub_7FF6D681A7D0
00007FF6D681D0E4 xor     r8d, r8d
00007FF6D681D0E7 xor     edx, edx
00007FF6D681D0E9 mov     rcx, [rsp+11380h+var_11330]
00007FF6D681D0EE call    sub_7FF6D683C5C8
...
```

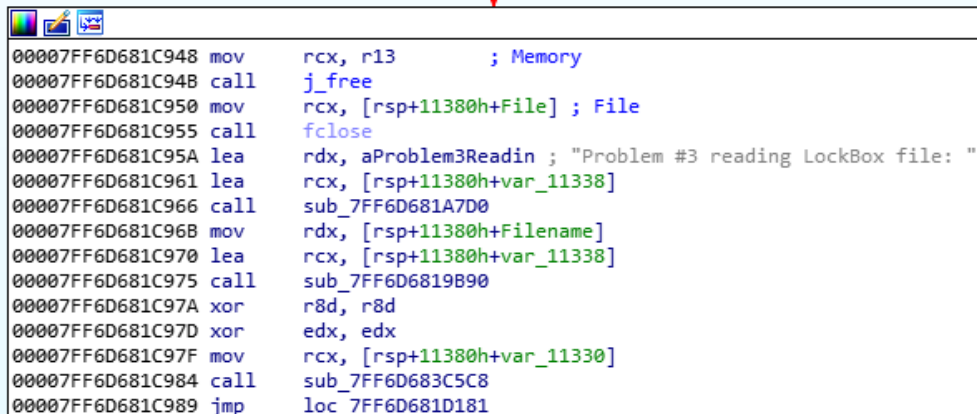
```
00007FF6D681CCAB sub     r12, 40h
00007FF6D681CCAF movsxd  rax, ecx
00007FF6D681CCB2 cmp     [rax+rbx], sil
00007FF6D681CCB6 jz      loc_7FF6D681CD3C
```

```
00007FF6D681CCBC lea     rdx, [rax+rbx]
```

# Jumping to String



```
00007FF6D681C9CA mov     rcx, r13             ; Memory
00007FF6D681C9CD call    j_free
00007FF6D681C9D2 mov     rcx, [rsp+11380h+File] ; File
00007FF6D681C9D7 call    fclose
00007FF6D681C9DC lea     rdx, aIncorrectAnswer ; "Incorrect answers."
00007FF6D681C9E3 lea     rcx, [rsp+11380h+var_11338]
00007FF6D681C9E8 call    sub_7FF6D681A7D0
00007FF6D681C9ED xor     r8d, r8d
00007FF6D681C9F0 xor     edx, edx
00007FF6D681C9F2 mov     rcx, [rsp+11380h+var_11330]
00007FF6D681C9F7 call    sub_7FF6D683C5C8
00007FF6D681C9FC jmp     loc_7FF6D681D181
```



```
00007FF6D681C948 mov     rcx, r13             ; Memory
00007FF6D681C948 call    j_free
00007FF6D681C950 mov     rcx, [rsp+11380h+File] ; File
00007FF6D681C955 call    fclose
00007FF6D681C95A lea     rdx, aProblem3Readin ; "Problem #3 reading LockBox file: "
00007FF6D681C961 lea     rcx, [rsp+11380h+var_11338]
00007FF6D681C966 call    sub_7FF6D681A7D0
00007FF6D681C96B mov     rdx, [rsp+11380h+Filename]
00007FF6D681C970 lea     rcx, [rsp+11380h+var_11338]
00007FF6D681C975 call    sub_7FF6D6819B90
00007FF6D681C97A xor     r8d, r8d
00007FF6D681C97D xor     edx, edx
00007FF6D681C97F mov     rcx, [rsp+11380h+var_11330]
00007FF6D681C984 call    sub_7FF6D683C5C8
00007FF6D681C989 jmp     loc_7FF6D681D181
```

# Looking Upwards

```
00007FF6D681C98E
00007FF6D681C98E loc_7FF6D681C98E:
00007FF6D681C98E mov     [rsp+11380h+var_11348], 1
00007FF6D681C996 mov     [rsp+11380h+nShowCmd], 40000000h
00007FF6D681C99E mov     [rsp+11380h+lpDirectory], r13
00007FF6D681C9A3 mov     r9d, 10h
00007FF6D681C9A9 lea     r8, [rbp+11280h+var_148]
00007FF6D681C9B0 lea     edx, [r9+8]
00007FF6D681C9B4 lea     rcx, [rbp+11280h+var_180]
00007FF6D681C9B8 call    sub_7FF6D68183A0
00007FF6D681C9C0 cmp     [rbp+11280h+var_180], 0
00007FF6D681C9C8 jz      short loc_7FF6D681CA01
```

```
00007FF6D681CA01
00007FF6D681CA01 loc_7FF6D681CA01:
00007FF6D681CA01 mov     [rbp+11280h+var_10ED8], 0Fh
00007FF6D681CA0C mov     [rbp+11280h+var_10EE0], rsi
00007FF6D681CA13 mov     byte ptr [rbp+11280h+var_10EF0], 0
00007FF6D681CA1A mov     r8d, 10h ; Size
00007FF6D681CA20 lea     rdx, aNoConnectionTo ; "No connection to BTC Network."
00007FF6D681CA27 lea     rcx, [rbp+11280h+var_10EF0] ; Dst
00007FF6D681CA2E call    sub_7FF6D68143E0
00007FF6D681CA33 nop
```

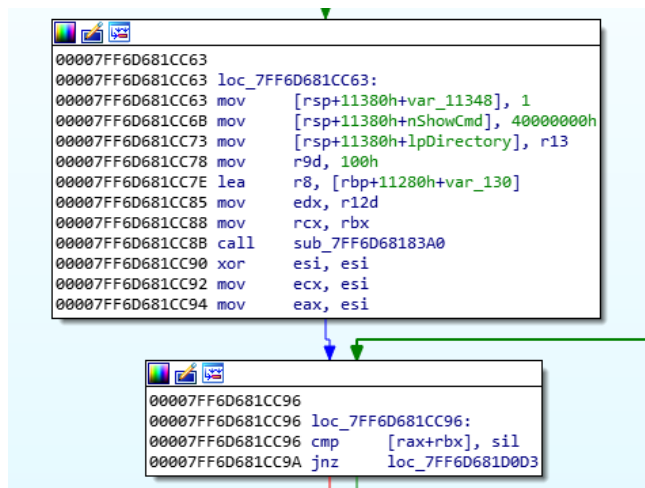
```
00007FF6D681CA34
00007FF6D681CA34 loc_7FF6D681CA34:
00007FF6D681CA34 mov     rax, cs:Memory
00007FF6D681CA3B test     rax, rax
00007FF6D681CA3E jz      loc_7FF6D681D11E
```

```
00007FF6D681CA44 mov     esi, [rax+30h]
00007FF6D681CA47 xor     ebx, ebx
00007FF6D681CA49 test     esi, esi
00007FF6D681CA4B jle     short loc_7FF6D681CA6C
```

```
To: sub_7FF6D681BD00+CCA
00007FF6D681C9CA mov     rcx, r13 ; Memory
00007FF6D681C9CD call    j_free
00007FF6D681C9D2 mov     rcx, [rsp+11380h+File] ; File
00007FF6D681C9D7 call    fclose
00007FF6D681C9DC lea     rdx, aIncorrectAnswe ; "Incorrect answers."
00007FF6D681C9E3 lea     rcx, [rsp+11380h+var_11338]
00007FF6D681C9E8 call    sub_7FF6D681A7D0
00007FF6D681C9ED xor     r8d, r8d
00007FF6D681C9F0 xor     edx, edx
...
```



# Setting Up Breakpoints



00007FF605C7CC57	48:8D15 0AA92200	lea rdx,qword ptr ds:[7FF605EA7568]	00007FF60
00007FF605C7CC5E	E9 AA000000	jmp timelock.7FF605C7CD00	
00007FF605C7CC63	C74424 38 01000000	mov dword ptr ss:[rsp+38],1	
00007FF605C7CC68	C74424 28 00000040	mov dword ptr ss:[rsp+28],40000000	
00007FF605C7CC73	4C:896C24 20	mov qword ptr ss:[rsp+20],r13	
00007FF605C7CC78	41:B9 00010000	mov r9d,100	
00007FF605C7CC7E	4C:8D85 50110100	lea r8,qword ptr ss:[rbp+11150]	
00007FF605C7CC85	41:8BD4	mov edx,r12d	
00007FF605C7CC88	48:88CB	mov rcx,rbx	
00007FF605C7CC8B	E8 1087FFFF	call timelock.7FF605C783A0	
00007FF605C7CC90	33F6	xor esi,esi	
00007FF605C7CC92	8BCE	mov ecx,esi	
00007FF605C7CC94	8BC6	mov eax,esi	
00007FF605C7CC96	40:383418	cmp byte ptr ds:[rax+rbx],sil	
00007FF605C7CC9A	0F85 33040000	jne timelock.7FF605C7D0D3	
00007FF605C7CC9B	FFC1	inc esi	

# Decryption Function Found

Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	Watch 1	[x=] Locals	Str																		
Address		Hex												ASCII											
000001C12E04DE80		AB AB AB AB AB AB AB AB AB AB AB AB AB AB												« «											

Address	Hex	ASCII
000001C12E04DE80	AB AB AB AB AB AB AB AB AB AB AB AB AB AB	<<<<<<<<<<<<<<<<<<<<<<
000001C12E04DE90	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001C12E04DEA0	EE FE EE FE EE FE EE FE B6 80 F4 3D 07 09 04 3A	ïþîþîþîþŸ.ð=...:
000001C12E04DEB0	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001C12E04DEC0	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001C12E04DED0	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001C12E04DEE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001C12E04DEF0	73 65 63 72 65 74 32 E 74 78 74 00 41 41 41 41	secret2.txt.AAAA
000001C12E04DF00	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAAAA
000001C12E04DF10	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAAAA
000001C12E04DF20	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAAAA
000001C12E04DF30	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAAAA

# Validation Loop Immediately Below

```
00007FF6D681CC85 mov     edx, r12d
00007FF6D681CC88 mov     rcx, rbx
00007FF6D681CC8B call    sub_7FF6D68183A0
00007FF6D681CC90 xor     esi, esi
00007FF6D681CC92 mov     ecx, esi
00007FF6D681CC94 mov     eax, esi
```

```
00007FF6D681CC96
00007FF6D681CC96 loc_7FF6D681CC96:
00007FF6D681CC96 cmp     [rax+rbx], sil
00007FF6D681CC9A jnz     loc_7FF6D681D0D3
```

```
00007FF6D681CCA0 inc     ecx
00007FF6D681CCA2 inc     rax
00007FF6D681CCA5 cmp     rax, 40h
00007FF6D681CCA9 jl      short loc_7FF6D681CC96
```

# Breakthrough Found

00007FF605C7CB51	48:0705 80030000 00	mov qword ptr ss:[rbp+390],0
00007FF605C7CB5C	C685 90030000 00	mov byte ptr ss:[rbp+390],0
00007FF605C7CB63	48:8B05 56302800	mov rax,qword ptr ds:[7FF605EFFBC0]
00007FF605C7CB6A	48:0305 57302800	add rax,qword ptr ds:[7FF605EFFBC8]
→ 00007FF605C7CB71	48:2B85 10110100	sub rax,qword ptr ss:[rbp+11110]
00007FF605C7CB78	33D2	xor edx,edx

0x5C4E2C15 looks familiar

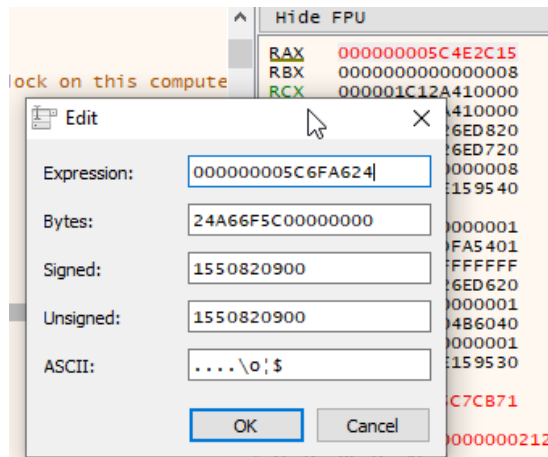
Converting to decimal: **1548626965**

**This is a Unix timestamp!**

**28/01/19 11:09:25**

RAX	000000005C4E2C15
RBX	0000000000000008
RCX	000001C12A410000
RDX	000001C12A410000
RBP	00000072D26ED820
RSP	00000072D26ED720
RSI	0000000000000008
RDI	000001C12E159540

# Modifying Timestamp



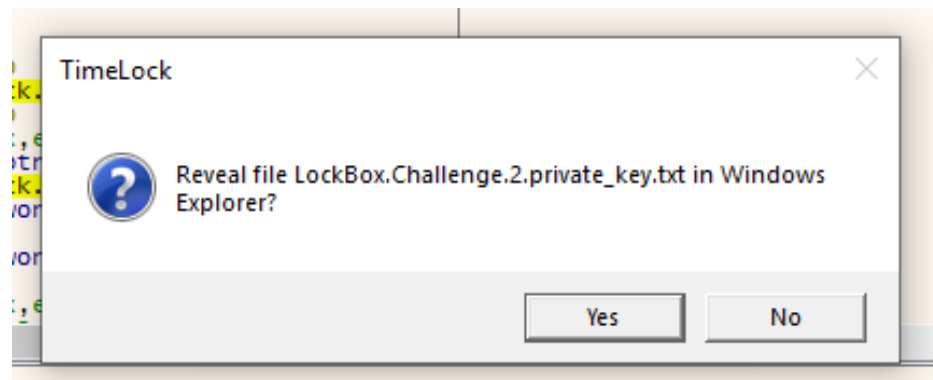
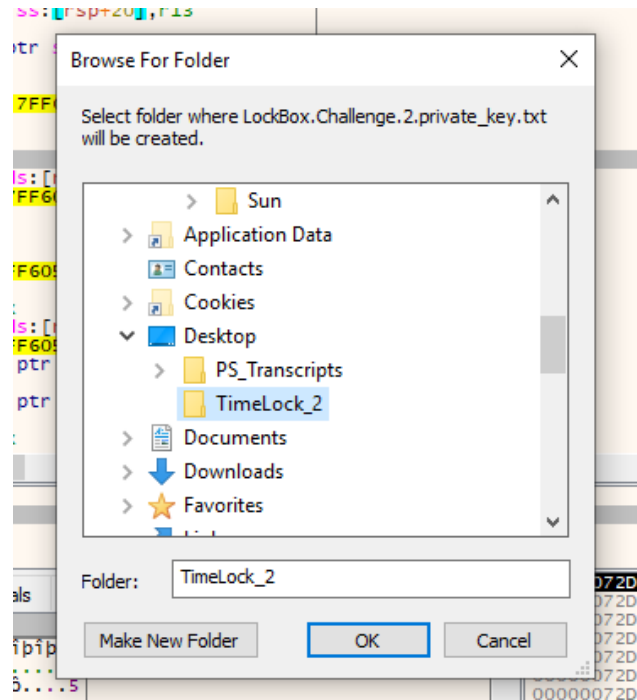
RAX	000000005C6FA624
RBX	00000000000000008
RCX	000001C12A410000
RDY	000001C12A410000
RBP	00000072D26ED820
RSP	00000072D26ED720
RSI	00000000000000008
RDI	000001C12E159540

00007FF605C7CC5E	E9 AA000000	jmp timelock.7FF605C7CD00
00007FF605C7CC63	C74424 38 01000000	mov dword ptr ss:[rsp+38],1
00007FF605C7CC68	C74424 28 00000040	mov dword ptr ss:[rsp+28],40000000
00007FF605C7CC73	4C:896C24 20	mov qword ptr ss:[rsp+20],r13
00007FF605C7CC78	41:B9 00010000	mov r9d,100
00007FF605C7CC7E	4C:8D85 50110100	lea r8,qword ptr ss:[rbp+11150]
00007FF605C7CC85	41:8BD4	mov edx,r12d
00007FF605C7CC88	48:8BCB	mov rcx,rbx
00007FF605C7CC8B	E8 10B7FFFF	call timelock.7FF605C783A0
00007FF605C7CC90	33F6	xor esi,esi
00007FF605C7CC92	8BCE	mov ecx,esi
00007FF605C7CC94	8BC6	mov eax,esi
00007FF605C7CC96	40:383418	cmp byte ptr ds:[rax+rbx],si

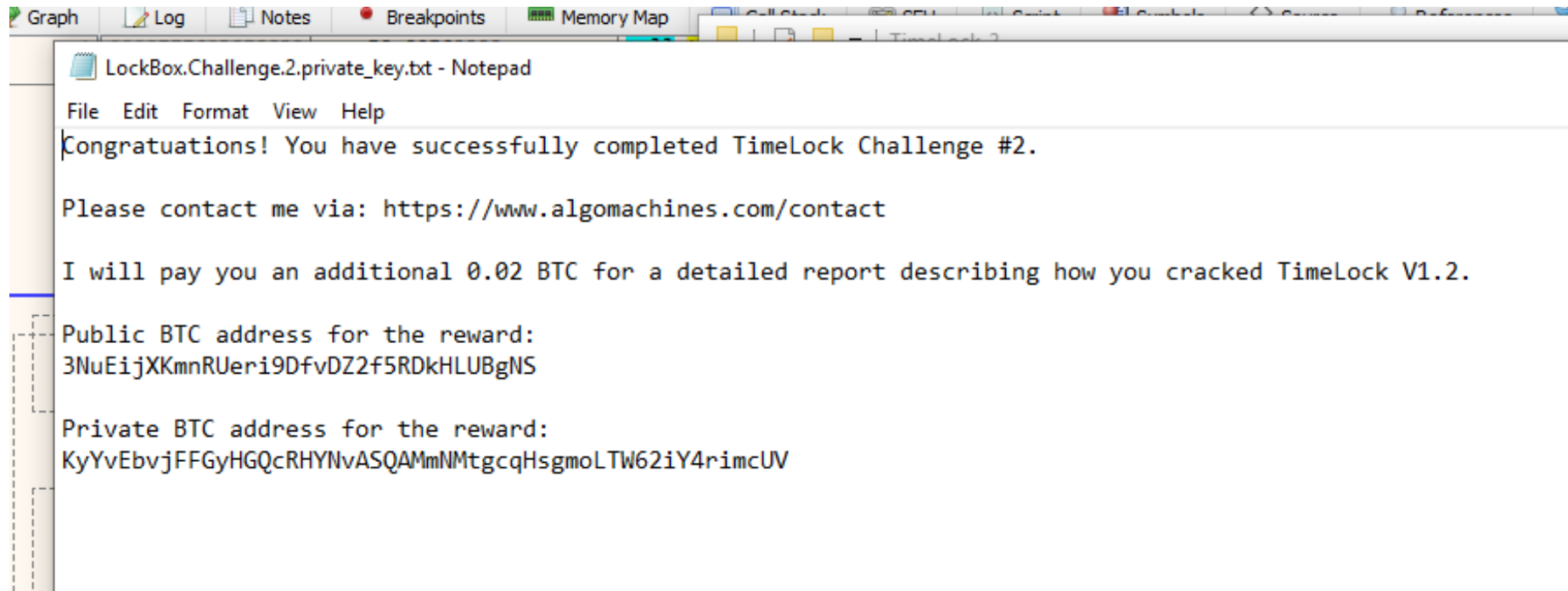
# Stepping Over Decryption

[illegible][illegible]

# Looks Promising...



# Loot #2



The image shows a screenshot of a Notepad application window titled "LockBox.Challenge.2.private\_key.txt - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text inside the Notepad is as follows:

```
File Edit Format View Help
Congratuations! You have successfully completed TimeLock Challenge #2.

Please contact me via: https://www.algomachines.com/contact

I will pay you an additional 0.02 BTC for a detailed report describing how you cracked TimeLock V1.2.

Public BTC address for the reward:
3NuEijXKmnRUeri9DfvDZ2f5RDkHLUBgNS

Private BTC address for the reward:
KyYvEbvjFFGyHGQcRHYNvASQAMmNMtgqHsgmoLTW62iY4rimcUV
```



# Lessons Learned:

## Vulnerability:

- Secret was found and replaced, with no additional validation.

## How to fix:

- Secrets should be hashed, so they are difficult / impossible to locate.
- Keys should not be generated locally, but instead supplied by a trusted third party.

# Challenge #3

Posted by u/cryptocomicon 9 days ago

## TimeLock your digital assets

Over the years I've seen many people wondering how they can transfer ownership of their digital assets in the future. They don't want to give a loved one a copy of their wallet seed, but they do want to make sure that no matter what, those assets are made available at a date in the future.

Designing an un-hackable TimeLock is challenging. This is my third version and the third challenge, with a 0.02 BTC reward.

Please give it a try.

More information at [algomachines.com](https://algomachines.com)

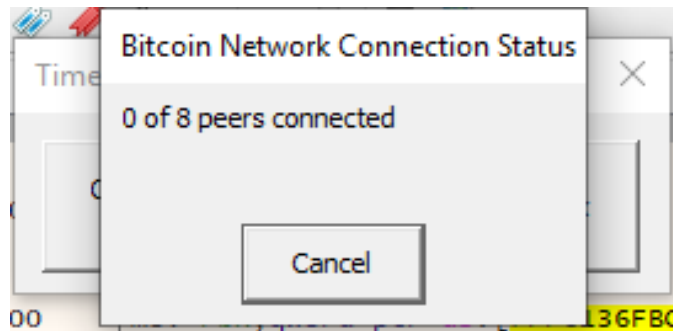
Link to the challenge: [challenge](#)

Here's a link to the Creator screen for this lock box: [Creator](#). This shows you the available time period for the lock box. I'm also giving you the password and the answer to the one question... much more information than you would have if you stumbled upon this file and wanted to crack it.

# Plan of Attack

- Launch a “Sybil” Attack
  - Introduce malicious nodes as the network.
  - Behave exactly like legitimate nodes.
  - Have time set to the future.
  - Disable internet access and force local nodes.

# TimeLock Uses DNS Seed Nodes



- seed.bitcoin.spia.be
- bitseed.xf2.org
- dnsseed.bitcoin.dashjr.org
- dnsseed.bluematt.org
- missionctrl.info

```
Domain Name System (response)
Transaction ID: 0x0002
> Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 21
Authority RRs: 0
Additional RRs: 0
v Queries
  > dnsseed.bluematt.me: type A, class IN
v Answers
  > dnsseed.bluematt.me: type A, class IN, addr 142.93.167.187
  > dnsseed.bluematt.me: type A, class IN, addr 70.120.24.242
  > dnsseed.bluematt.me: type A, class IN, addr 144.76.99.209
  > dnsseed.bluematt.me: type A, class IN, addr 24.101.67.50
  > dnsseed.bluematt.me: type A, class IN, addr 95.161.12.45
  > dnsseed.bluematt.me: type A, class IN, addr 188.193.164.196
  > dnsseed.bluematt.me: type A, class IN, addr 169.229.198.105
  > dnsseed.bluematt.me: type A, class IN, addr 91.121.97.23
  > dnsseed.bluematt.me: type A, class IN, addr 95.216.111.121
  > dnsseed.bluematt.me: type A, class IN, addr 111.90.159.213
  > dnsseed.bluematt.me: type A, class IN, addr 45.20.67.1
  > dnsseed.bluematt.me: type A, class IN, addr 86.15.59.249
  > dnsseed.bluematt.me: type A, class IN, addr 45.120.52.199
  > dnsseed.bluematt.me: type A, class IN, addr 37.59.63.56
  > dnsseed.bluematt.me: type A, class IN, addr 24.210.98.8
  > dnsseed.bluematt.me: type A, class IN, addr 129.122.222.134
  > dnsseed.bluematt.me: type A, class IN, addr 98.228.169.22
  > dnsseed.bluematt.me: type A, class IN, addr 144.76.78.214
  > dnsseed.bluematt.me: type A, class IN, addr 95.84.156.162
  > dnsseed.bluematt.me: type A, class IN, addr 71.13.92.62
```

# Rolling Our Own DNS

```
basedns.py - C:/Users/Analysis/Desktop/basedns.py (3.7.2)
File Edit Format Run Options Window Help

#!/usr/bin/env python3
# (c) 2014 Patryk Hes
import socketserver
import sys
import random

DNS_HEADER_LENGTH = 12
# TODO make some DNS database with IPs connected to regexs
IP = '127.0.0.1'

class DNSHandler(socketserver.BaseRequestHandler):
    def handle(self):
        socket = self.request[1]
        data = self.request[0].strip()

        # If request doesn't even contain full header, don't respond.
        if len(data) < DNS_HEADER_LENGTH:
            return

        # Try to read questions - if they're invalid, don't respond.
        try:
            all_questions = self.dns_extract_questions(data)
        except IndexError:
            return

        # Filter only those questions, which have QTYPE=A and QCLASS=IN
        # TODO this is very limiting, remove QTYPE filter in future, handle diff
        accepted_questions = []
        for question in all_questions:
            name = str(b'.'.join(question['name']), encoding='UTF-8')
            if question['qtype'] == b'\x00\x01' and question['qclass'] == b'\x00':
                accepted_questions.append(question)
                print('\033[32m{}\033[39m'.format(name))
            else:
                print('\033[31m{}\033[39m'.format(name))

        response = (
            self.dns_response_header(data) +
            self.dns_response_questions(accepted_questions) +
            self.dns_response_answers(accepted_questions)
        )

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help

Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Analysis/Desktop/fakedns.py =====
[36mStarted DNS server.[39m
[31m1.0.0.127.in-addr.arpa[39m
[32mhello.ss.internal[39m
[31mhello.ss.internal[39m
[32mwpad.internal[39m
[32mwpad.internal[39m
[ 32mwpad.internal[39m
[32mwpad.internal[39m
[32mwpad.internal[39m
[32mwpad.internal[39m
Fail
>>>
===== RESTART: C:/Users/Analysis/Desktop/basedns.py =====
[36mStarted DNS server.[39m
[32mwpad.internal[39m
[31m1.0.0.127.in-addr.arpa[39m
[32mhello.ss.internal[39m
[31mhello.ss.internal[39m
[31m1.0.0.127.in-addr.arpa[39m
[32mhello.com.internal[39m
[31mhello.com.internal[39m
[32mwpad.internal[39m
[32mwpad.internal[39m

Ln: 10 Col: 0
Ln: 28 Col: 0
```

# Adding Randomised DNS Entries to Lookup

```
records = b''
for question in questions:
    for i in range(1, 21):
        record = b''
        for label in question['name']:
            # Length octet
            record += bytes([len(label)])
            record += label
        # Zero length octet
        record += b'\x00'
        # TYPE - just copy QTYPE
        # TODO QTYPE values set is superset of TYPE values set, handle
        record += question['qtype']
        # CLASS - just copy QCLASS
        # TODO QCLASS values set is superset of CLASS values set, handle
        record += question['qclass']
        # TTL - 32 bit unsigned integer. Set to 0 to inform, that respo
        # should not be cached.
        record += b'\x00\x00\x00\x00'
        # RDLLENGTH - 16 bit unsigned integer, length of RDATA field.
        # In case of QTYPE=A and QCLASS=IN, RDLLENGTH=4.
        record += b'\x00\x04'
        # RDATA - in case of QTYPE=A and QCLASS=IN, it's IPv4 address.
        temp_IP = IP + '.' + str(random.randint(1, 200))
        temp_IP = temp_IP + '.' + str(random.randint(1, 200))
        temp_IP = temp_IP + '.' + str(random.randint(1, 200))
        record += b''.join(map(
            lambda x: bytes([int(x)]),
            temp_IP.split('.')
        ))
        records += record
return records
```

```
C:\Users\Analysis>nslookup hello.com
Server: UnKnown
Address: 127.0.0.1
```

```
Non-authoritative answer:
Name:      hello.com.internal
Addresses: 127.61.198.11
           127.110.170.2
           127.95.65.29
           127.195.84.141
           127.22.66.121
           127.153.66.63
           127.92.26.30
           127.89.164.167
           127.118.186.71
           127.56.36.25
           127.90.114.142
           127.49.92.175
           127.122.12.13
           127.120.48.18
           127.20.28.117
           127.50.2.168
           127.109.138.131
           127.7.57.131
           127.65.134.11
           127.78.64.2
```

# Starting Our Bitcoin Node

Date & time

Date and time

7:26 PM, Saturday, 9 March 2019

Set time automatically



Set time zone automatically

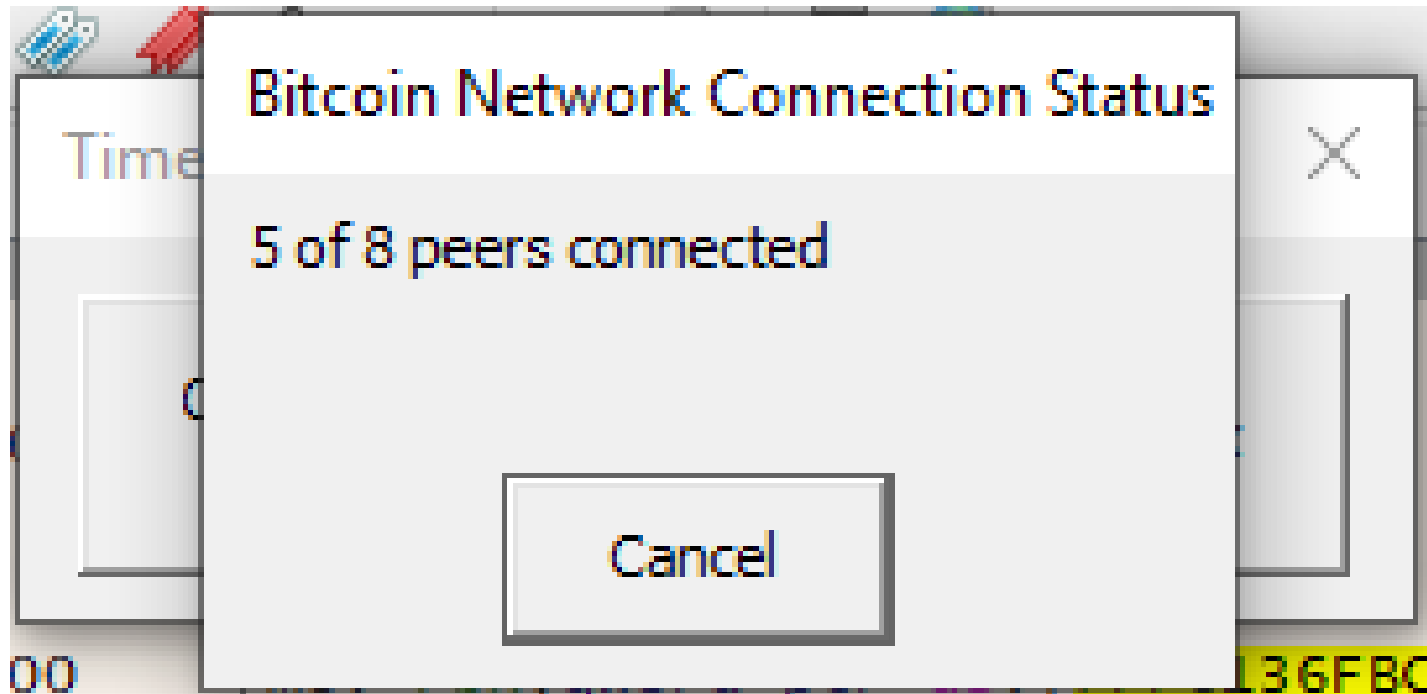


Change date and time

Change





```
Command Prompt - "C:\Program Files\Bitcoin\daemon\bitcoind.exe"
2019-03-09T06:31:11Z init message: Verifying blocks...
2019-03-09T06:31:11Z block index 262ms
2019-03-09T06:31:11Z init message: Loading wallet...
2019-03-09T06:31:11Z [default wallet] nFileVersion = 170100
2019-03-09T06:31:11Z [default wallet] Keys: 2001 plaintext, 0 encrypted, 2001 w/ metadata, 2001 total. Unknown wallet records: 1
2019-03-09T06:31:11Z [default wallet] Wallet completed loading in 168ms
2019-03-09T06:31:11Z [default wallet] setKeyPool.size() = 2000
2019-03-09T06:31:11Z [default wallet] mapWallet.size() = 0
2019-03-09T06:31:11Z [default wallet] mapAddressBook.size() = 0
2019-03-09T06:31:11Z mapBlockIndex.size() = 4001
2019-03-09T06:31:11Z Imported mempool transactions from disk: 0 succeeded, 0 failed, 0 expired, 0 already there
2019-03-09T06:31:11Z nBestHeight = 0
2019-03-09T06:31:11Z torcontrol thread start
2019-03-09T06:31:11Z Bound to [::]:8333
2019-03-09T06:31:11Z Bound to 0.0.0.0:8333
2019-03-09T06:31:11Z init message: Loading P2P addresses...
2019-03-09T06:31:11Z Loaded 1033 addresses from peers.dat 0ms
2019-03-09T06:31:11Z init message: Loading banlist...
2019-03-09T06:31:11Z init message: Starting network threads...
2019-03-09T06:31:11Z net thread start
2019-03-09T06:31:11Z dnsseed thread start
2019-03-09T06:31:11Z init message: Done loading
2019-03-09T06:31:11Z addcon thread start
2019-03-09T06:31:11Z opencon thread start
2019-03-09T06:31:11Z msghand thread start
2019-03-09T06:31:22Z Loading addresses from DNS seeds (could take a while)
2019-03-09T06:31:22Z 140 addresses found from DNS seeds
2019-03-09T06:31:22Z dnsseed thread exit
```

# Did We Hack The Thing Yet?





# When Stuck – Search Strings

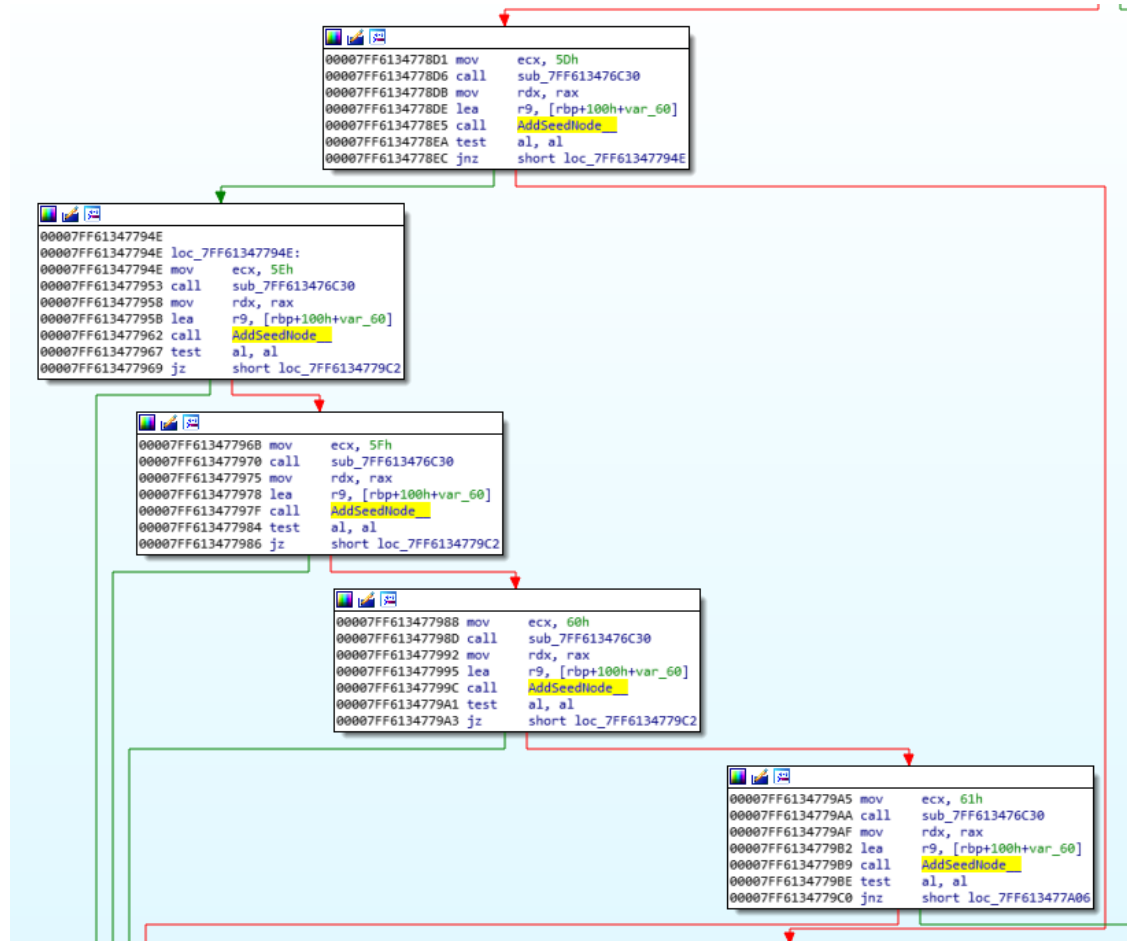
Address	Length	Type	String
 .rdata:0000...	00000036	C	AddSeedNode() : already has seed node matching url :
 .rdata:0000...	0000003B	C	AddSeedNode() : - no more than 32 seed nodes may be added.
 .rdata:0000...	000000A1	C	Connect() : number of seed nodes is zero and the size of peer_info is less than 50, must h...
 .rdata:0000...	00000045	C	Connect() : number of seed ports is not equal to number of seed urls

```
00007FF613464D98  
00007FF613464D98 loc_7FF613464D98:      ; Size  
00007FF613464D98 mov     r8d, 35h  
00007FF613464D9E lea     rdx, aAddseednodeAlr ; "AddSeedNode() : already has seed node m".  
00007FF613464DA5 mov     rcx, rsi ; Dst  
00007FF613464DA8 call    sub_7FF6134643A0  
00007FF613464DAD cmp     [rdi], bl  
00007FF613464DAF jz      short loc_7FF613464DBE
```

```
00007FF613464DB1 or     rbx, 0FFFFFFFFFFFFFFFh
```

```
00007FF613464DB5  
00007FF613464DB5 loc_7FF613464DB5:  
00007FF613464DB5 inc     rbx  
00007FF613464DB8 cmp     byte ptr [rdi+rbx], 0  
00007FF613464DBC jnz     short loc_7FF613464DB5
```

# Determining How Seed Nodes are Added



# Open Debugger; Set Breakpoint

The screenshot shows the x64dbg debugger interface for the process TimeLock.exe (PID: 3A8). The main window displays the CPU window with the instruction list. A breakpoint is set on the instruction at address 00007FF6134778D1, which is highlighted in red. The instruction is `00007FF6134778D1: 4C:8D85 A0000000 lea r8,qword ptr ss:[rbp+A0]`. The Disassembly window shows the assembly code for the current instruction, and the Register window shows the values of the registers. The Memory dump window shows the memory dump for the current instruction. The Command window shows the command `.text:00007FF613477927 timelock.exe:$17927 #16D27`.

**CPU Window:**

Address	Hex	Disassembly
00007FF6134778AD	4C:8D85 A0000000	lea r8,qword ptr ss:[rbp+A0]
00007FF6134778B4	48:8815 C5452800	mov rdx,qword ptr ds:[7FF6136FBE80]
00007FF6134778B8	E8 E0E5FEFF	call timelock.7FF613465EAD
00007FF6134778C0	48:8805 39432800	mov rax,qword ptr ds:[7FF6136FBC00]
00007FF6134778C7	8378 50 32	cmp dword ptr ds:[rax+50],32
00007FF6134778CB	0F8D 35010000	jge timelock.7FF613477A06
00007FF6134778D1	89 5D000000	mov ecx,5D
00007FF6134778D6	E8 55F3FFFF	call timelock.7FF613476C30
00007FF6134778DB	48:8BD0	mov rdx,rax
00007FF6134778DE	4C:8D8D A0000000	lea r9,qword ptr ss:[rbp+A0]
00007FF6134778E5	E8 06D4FEFF	call timelock.7FF613464CF0
00007FF6134778EA	84C0	test al,al
00007FF6134778EC	75 60	jne timelock.7FF61347794E
00007FF6134778EE	48:881D 0B432800	mov rdx,qword ptr ds:[7FF6136FBC00]
00007FF6134778F5	48:85D8	test rdx,rdx
00007FF6134778F8	74 10	jle timelock.7FF61347790A
00007FF6134778FA	48:88CB	mov rcx,rbx
00007FF6134778FD	E8 DED0FEFF	call timelock.7FF6134649E0
00007FF613477902	48:88CB	mov rcx,rbx
00007FF613477905	E8 9ED30000	call timelock.7FF613484CA8
00007FF61347790A	48:893D EF422800	mov qword ptr ds:[7FF6136FBC00],rdi
00007FF613477911	48:83BD B8000000	cmp qword ptr ss:[rbp+B8],10
00007FF613477919	72 13	jnb timelock.7FF61347792E
00007FF61347791B	48:8895 A0000000	mov rdx,qword ptr ss:[rbp+A0]
00007FF613477922	48:85D2	test rdx,rdx
00007FF613477925	75 0E	jne timelock.7FF613477935
00007FF613477927	8B C7	mov ebx,edi

**Register Window:**

Register	Value
RAX	000002AAA1530630
RBX	000002AAA1613560
RCX	000002AA9C840000
RDX	000002AA9C840000
RBP	000000C53F5AD180
RSP	000000C53F5AD080
RSI	000000C53F5AD3D0
RDI	0000000000000000
R8	0000000000000001
R9	000002AAA15F6A01
R10	00000000FFFFFFFF
R11	000000C53F5ACF50
R12	0000000000000000
R13	000002AAA2372040
R14	000000C53F5BE4C0
R15	000000C53F5AD398
RIP	00007FF6134778D1

**Memory Dump Window:**

Address	Hex	ASCII
00007FF802FD1000	CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC	iiiiiiiiiiiiiiii
00007FF802FD1010	40 53 48 83 EC 50 48 88 84 24 98 00 00 4D 88	@SH.iPH..\$....M.
00007FF802FD1020	D9 48 83 C0 07 45 8B D0 48 83 E0 F8 88 DA 66	UH.A.E.DH.aoh.UF
00007FF802FD1030	85 C9 0F 85 8C 75 0A 00 44 88 8C 24 80 00 00 00	.E...u.D.\$...
00007FF802FD1040	4D 8B C3 48 89 44 24 28 41 88 D2 48 8B 84 24 90	M.AH.D\$(A.OH...\$
00007FF802FD1050	00 00 00 48 88 CB 48 89 44 24 20 E8 0C 00 00 00	...H.EH.D\$e...
00007FF802FD1060	48 83 C4 50 5B C3 CC CC CC CC CC CC CC CC CC CC	H.AP{iiiiiiH.\\$

**Command Window:**

```
.text:00007FF613477927 timelock.exe:$17927 #16D27
```

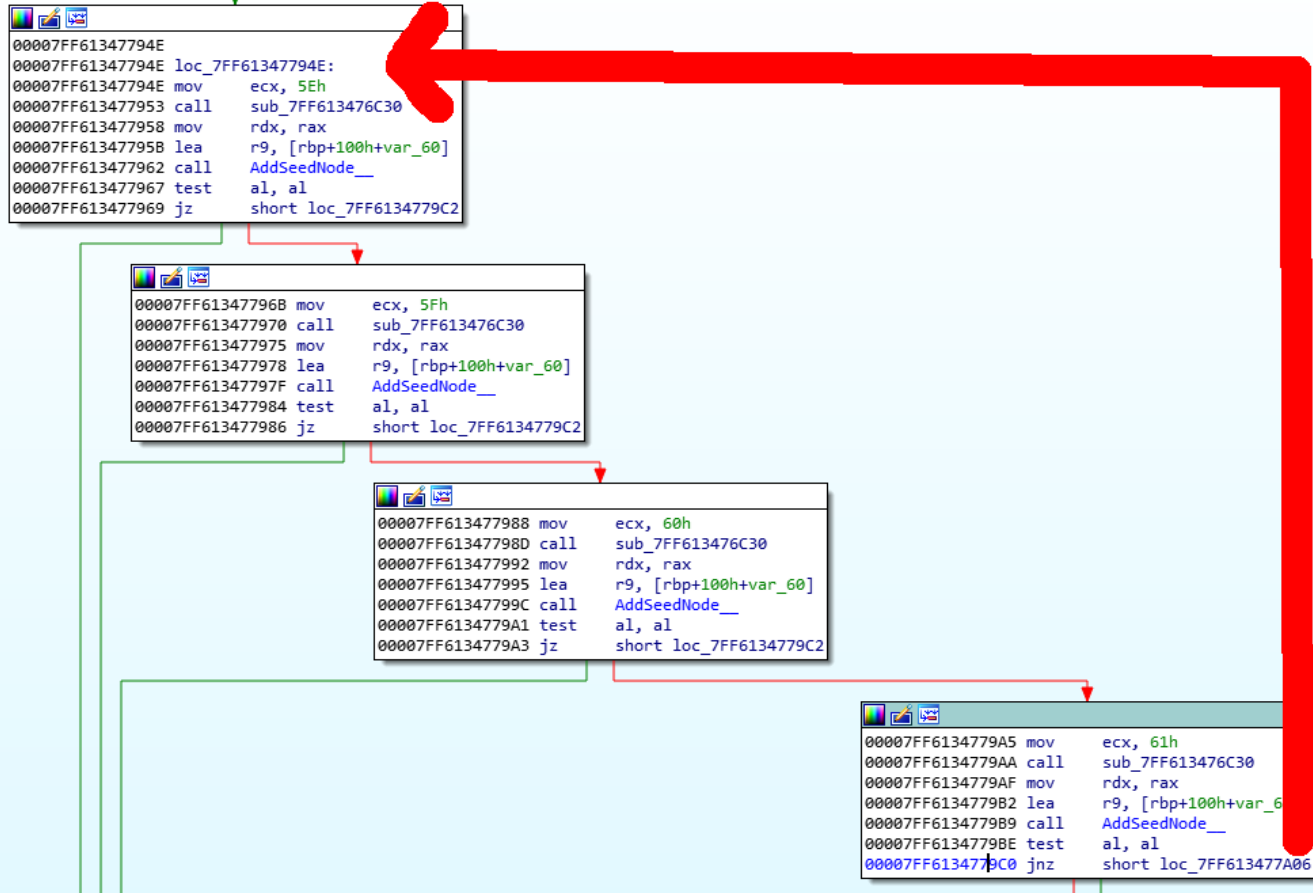
# Breakpoint Hits, Familiar DNS Entry

•	00007FF6134778AD	4C:8D85 A0000000	lea r8,qword ptr ss:[rbp+A0]
•	00007FF6134778B4	48:8815 C5452800	mov rdx,qword ptr ds:[7FF6136FB8E80]
•	00007FF6134778B8	E8 E0E5FEFF	call timelock.7FF613465EA0
•	00007FF6134778C0	48:8805 39432800	mov rax,qword ptr ds:[7FF6136FBC00]
•	00007FF6134778C7	8378 50 32	cmp dword ptr ds:[rax+50],32
•	00007FF6134778CB	✓ 0F8D 35010000	jge timelock.7FF613477A06
•	00007FF6134778D1	B9 5D000000	mov ecx,5D
•	00007FF6134778D6	E8 55F3FFFF	call timelock.7FF613476C30
•	00007FF6134778DB	48:88D0	mov rdx,rax
•	00007FF6134778DE	4C:8D8D A0000000	lea r9,qword ptr ss:[rbp+A0]
•	00007FF6134778E5	E8 06D4FEFF	call timelock.7FF613464CF0
•	00007FF6134778EA	84C0	test al,al
•	00007FF6134778EC	✓ 75 60	jne timelock.7FF61347794E
•	00007FF6134778EE	48:881D 0B432800	mov rbx,qword ptr ds:[7FF6136FBC00]
•	00007FF6134778F5	48:85D8	test rbx,rbx
•	00007FF6134778F8	✓ 74 10	je timelock.7FF61347790A
•	00007FF6134778FA	48:88CB	mov rcx,rbx

```
RAX 000001FB750E7D90
RBX 000001FB751855A0
RCX 00000000FFFFFFFF
RDX 0000000000000000
RBP 00000013528AD260
RSP 00000013528AD160
RSI 00000013528AD480
RDI 0000000000000000
```

```
"missionctrl.info"
"C:\\Users\\Analysi
```

# What Happens If We Do It Again?



# Modify DNS Entry, Avoid Conflicts

```
RAX 000001FB794DD00 "seed.bitcoin.sipa.be"  
RBX 000001FB751855A0 "C:\\Users\\Analysis\\AppData\\Roaming\\Alc  
RCX 00000000FFFFFFFF  
RDX 0000000000000000  
RBP 0000001352BAD260  
RSP 0000001352BAD160  
RSI 0000001352BAD480  
RDI 0000000000000000
```

Modify value

Expression: 69

Bytes: 69

Signed: 105

Unsigned: 105

ASCII: .....i

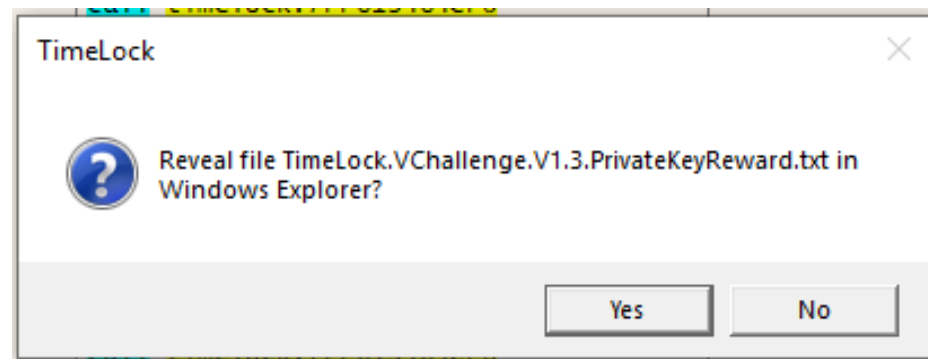
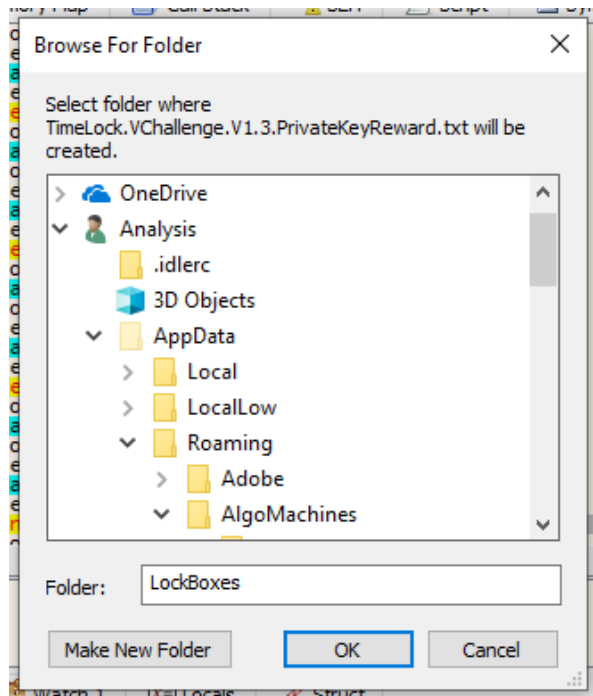
OK Cancel

Address Hex

000001FB794DDCD0	AB AB AB
000001FB794DDCE0	00 00 00
000001FB794DDCF0	EE FE EE
000001FB794DD000	73 73 73
000001FB794DD010	61 2E 65
000001FB794DD020	AB AB AB
000001FB794DD030	00 00 00

Address	Hex	ASCII
000001FB794DDCD0	AB AB AB AB AB AB AB AB	««««««««««ipip
000001FB794DDCE0	00 00 00 00 00 00 00 00	.....
000001FB794DDCF0	EE FE EE FE EE FE EE FE	ipipipip."WN..2
000001FB794DD000	73 65 65 64 2E 62 69 74	seed.bitcoin.sip
000001FB794DD010	61 2E 62 65 00 F0 AD BA	a.be.°.°.ip«
000001FB794DD020	AB AB AB AB AB AB AB AB	««««««««««ip
000001FB794DD030	00 00 00 00 00 00 00 00	.....

# Looks Promising...



# Loot #3

TimeLock.VChallenge.V1.3.PrivateKeyReward - Notepad

File Edit Format View Help

Well done!

You have successfully completed the TimeLock V1.3 challenge.

I am impressed!

Please contact me via: <https://www.algomachines.com/contact>

I will happily send you at least 0.02 BTC for a detailed report describing how you cracked TimeLock V1.3.

TimeLock 1.3 challenge reward public address: 34r4PbKUM2odwf1EV2Jnxx9d3k1rWKgAzD

TimeLock 1.3 challenge reward private address: Kx4TLBeaMLG19wkeocVX6YG63BTWErKvnTvnPfVgvXf5tD1U1Mij



# Lessons Learned:

## Vulnerability:

- DNS cannot be trusted, can be easily manipulated.
- Executables cannot be trusted as attackers can easily redirect execution flow.
- No attestation performed on connected nodes.

## How to fix:

- Tunnel DNS via own dnssec validating resolver.
- Validate nodes, in this case, verify blockchain?
- No internet access should raise alarm bells.
- Do not peer with localhost.

# Challenge #4

Posted by u/cryptocomicon 23 days ago

## Quest for the unhackable TimeLock

Do you want to hand off your digital assets to your loved ones in the future without handing them off to a third party today?

I've created a software product to do that, using the decentralized Bitcoin network, and I'm perfecting it with challenges and rewards.

TimeLock 1.5 is free software. You can read about it here: <https://www.algomachines.com/>

The challenge history and latest installer / challenge is here: <https://www.algomachines.com/people>

# Plan of Attack

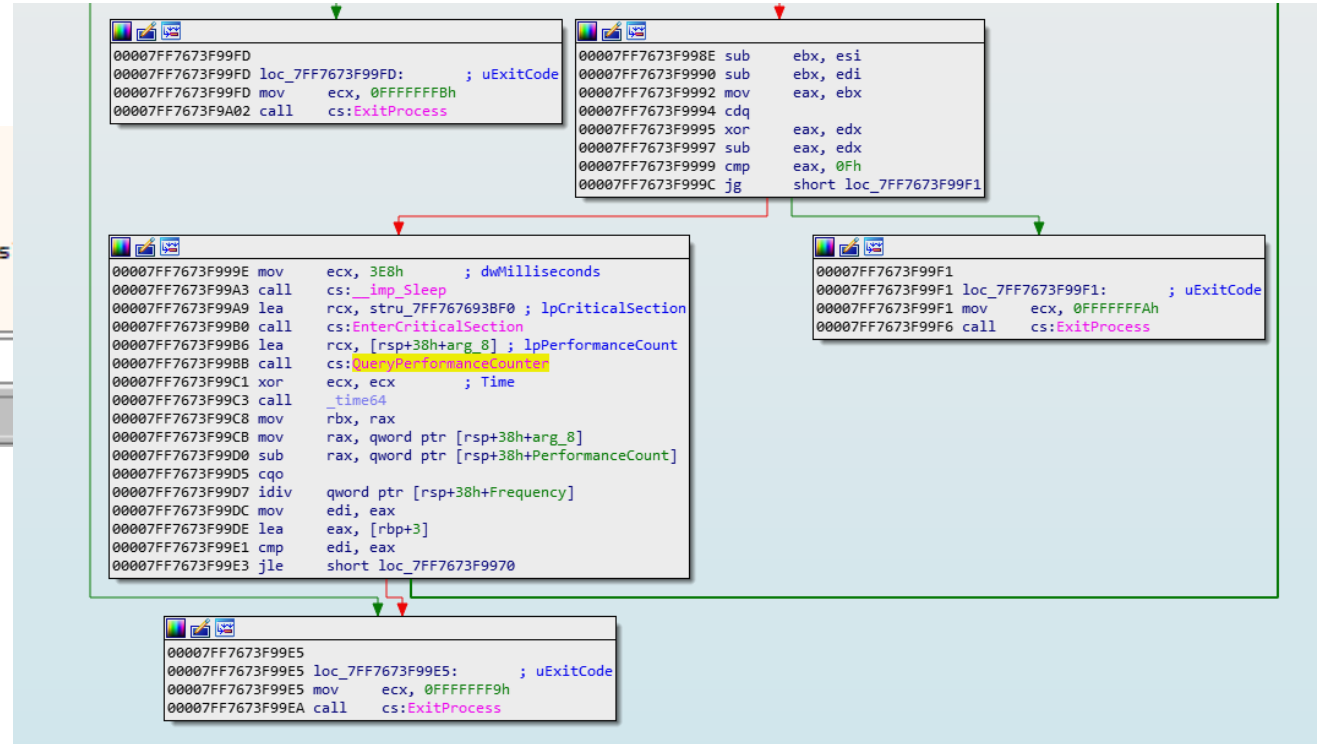
- Find encryption / decryption functions
- Review encryption:
  - Look for improper use of modes.
  - Look for weak algorithms.

# Anti Debugging Measures?

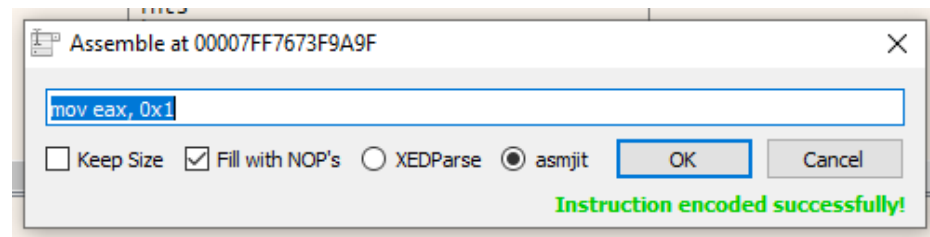
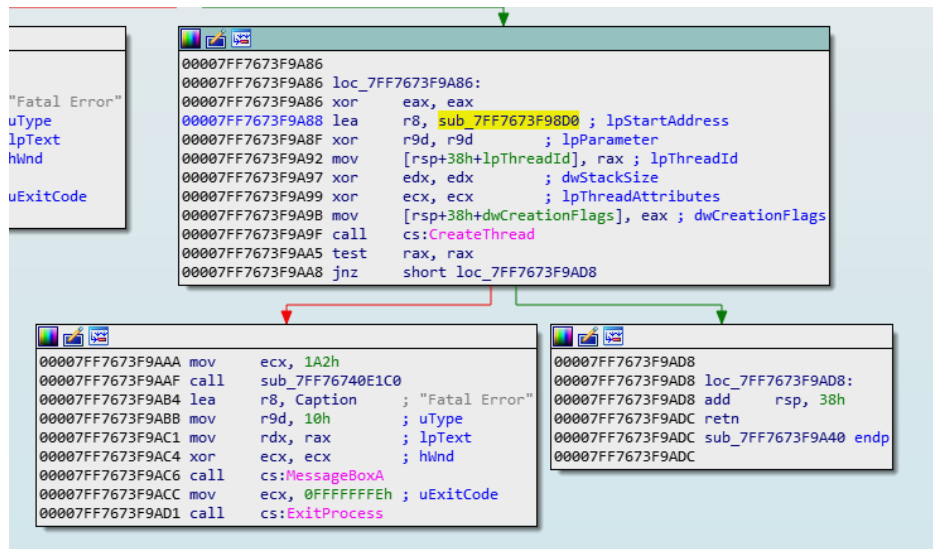
```
Thread 8CC exit
Thread 620 exit
Process stopped with exit code 0xFFFFFFFF9
Saving database to C:\Users\Analysis\Downloads
Debugging stopped!
```

Command:

**Terminated** Debugging stopped!



# Patch Out CreateThread To Defeat Anti Debugging



# Search for fread, fwrite and Encryption

```
00007FF7674180DB
00007FF7674180DB loc_7FF7674180DB:
00007FF7674180DB movsxd  rax, ecx
00007FF7674180DE lea     rcx, [rbx+41h]
00007FF7674180E2 add     rcx, rax          ; DstBuf
00007FF7674180E5 mov     r9, [rsp+530h+var_4D8] ; File
00007FF7674180EA mov     r8, rdi           ; Count
00007FF7674180ED mov     edx, 1           ; ElementSize
00007FF7674180F2 call    fread
00007FF7674180F7 cmp     rax, rdi
00007FF7674180FA jz      short loc_7FF76741813E
```

```
00007FF76741813E
00007FF76741813E loc_7FF76741813E:          ; File
00007FF76741813E mov     rcx, [rsp+530h+var_4D8]
00007FF767418143 call    fclose
00007FF767418148 mov     [rsp+530h+var_4F8], 1
00007FF767418150 mov     [rsp+530h+var_508], 40000000h
00007FF767418158 mov     [rsp+530h+var_510], r15
00007FF76741815D mov     r9d, 100h
00007FF767418163 lea     r8, [rbp+430h+var_260]
00007FF76741816A mov     edx, esi
00007FF76741816C mov     rcx, rbx
00007FF76741816F call    sub_7FF7673F9240
00007FF767418174 add     [r13+0], esi
00007FF767418178 mov     rcx, [rsp+530h+File] ; File
00007FF76741817D call    _ftelli64
00007FF767418182 mov     r9, [rsp+530h+File] ; File
00007FF767418187 mov     r8, r12           ; Count
00007FF76741818A mov     edx, 1           ; Size
00007FF76741818F mov     rcx, rbx          ; Str
00007FF767418192 call    fwrite
00007FF767418197 mov     rcx, r15          ; Memory
00007FF76741819A cmp     rax, r12
00007FF76741819D jz      short loc_7FF7674181DF
```

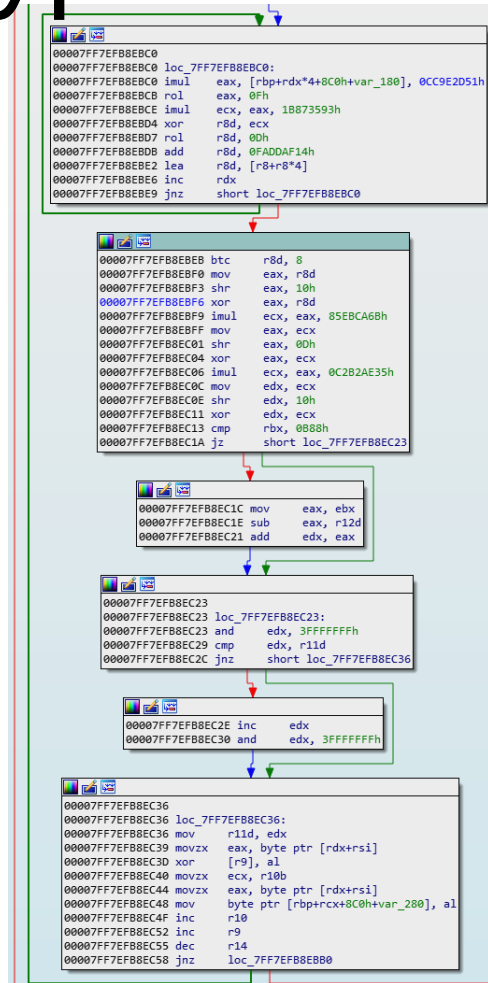
# Stepping Over Function Call

Address	Hex	ASCII
0000020B685018E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020B685018F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020B68501900	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020B68501910	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020B68501920	73 65 63 72 65 74 2E 74 78 74 00 41 41 41 41 41	secret.txt.AAAAA
0000020B68501930	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA
0000020B68501940	41 41 41 41 41 41 41 0D 0A 41 41 41 41 41 41 41	AAAAAA. AAAAAA
0000020B68501950	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA
0000020B68501960	41 41 41 41 41 0D 0A 41 41 41 41 41 41 41 41 41	AAAAA. AAAAAA
0000020B68501970	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA
0000020B68501980	41 41 41 0D 0A 41 41 41 41 41 41 41 41 41 41 41	AAA. AAAAAA
0000020B68501990	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA
0000020B685019A0	41 0D 0A 41 41 41 41 41 41 41 41 41 41 41 41 41	A. AAAAAA
0000020B685019B0	41 41 41 41 41 41 41 41 41 41 41 41 41 41 0D	AAAAAAAAAAAAAA.

Address	Hex	ASCII
0000020B685018E0	3F E6 AB C7 D1 17 0F 65 38 8A 59 00 12 34 F6 C9	æ«ÇN..e;.Y..4ôÉ
0000020B685018F0	85 2D 05 F0 A2 18 5D E6 B3 6D 37 99 F4 51 68 47	.-.ðc.]æ*m7.ðqHg
0000020B68501900	FE 5D F3 54 37 EE 63 86 D4 5C AF 38 6F 54 61 B9	p]óT7íc.ð\;oTa'
0000020B68501910	85 05 C5 E9 92 25 C1 9D 32 21 EB BF 60 A1 73 39	..Áe.%A.2!ëç'is9
0000020B68501920	26 A3 E9 25 00 AE 04 11 D0 E2 08 A5 23 9F 40 A5	&fé%.e..ðä.¥#. @¥
0000020B68501930	CC C7 EC AF 3F FA 6D 0D 4E E6 BC 3C 31 92 76 07	ìçì~?úm.Næ<1.v.
0000020B68501940	48 64 A1 8C 3F 97 9F AA 36 68 AB 61 BF CC 88 0A	Kdi.?...*6k«a;I..
0000020B68501950	24 68 9C F6 C2 8F CD 46 98 88 F5 DA 16 E0 F4 F0	\$h.ôÄ.îF..ðÜ.aôð
0000020B68501960	59 7E 6B 9A 0D A7 19 74 E2 6B 47 D5 91 C0 CA 97	Y~k..§.tâkGÖ.ÄÊ.
0000020B68501970	02 70 A0 CC 51 FC 85 75 DD 6B FB 02 81 86 41 3C	.p iQûpuYkû...A<
0000020B68501980	23 37 C3 7D 4E 9A 98 48 BC DB 14 86 3A 81 7D 80	#7A}N..H%0...}.
0000020B68501990	5B 00 BC 33 95 35 61 B5 C9 FB 64 45 DF A5 AD 07	[.%3.5apÉûDEß¥..
0000020B685019A0	E9 D0 98 F8 24 80 29 B5 DE 41 3C 75 78 17 32 B1	ëD.ø\$.)µpA<u{.2±
0000020B685019B0	8B 2D 8F FC 37 81 E0 11 CC 6C 1C 29 E0 23 17 87	.-.ü7.a.îl.)â#..

# Encryption Function

Generates single byte  
keystream



XOR 1 byte plaintext with 1  
byte keystream



# Symmetric Encryption

- Ciphertext is deterministic:
  - Same inputs create same outputs.
- Encryption is performed by xor...
  - ... so we can “decrypt” by xoring again!
- We have symmetric encryption!

# Extract Ciphertext, Create New LockBox

[illegible][illegible]

# Replace Dummy Data With Ciphertext

Address	Hex	Hex:
000001E3048732A0	AB AB AB AB AB AB AB AB	92 AA 49 29 E0 AF 58 82 42 10 93 A5 2D 65 75 75
000001E3048732B0	00 00 00 00 00 00 00 00	C0 F6 73 2D 65 B1 1F 1D 68 56 48 F1 ED 65 80 74
000001E3048732C0	00 00 00 00 00 00 00 00	3A 49 EF A5 0C DE C6 FD C4 98 F0 48 2F 76 A7 F8
000001E3048732D0	00 00 00 00 00 00 00 00	75 B0 45 B3 31 03 CC F9 11 F3 7C 58 31 78 A1 CB
000001E3048732E0	00 00 00 00 00 00 00 00	7A 4E E9 27 2B E8 30 29 70 08 0E 4E 0A 6F B8
000001E3048732F0	00 00 00 00 00 00 00 00	B6 A5 A1 02 48 64 82 6A F2 8A 27 FA 70 62 DC D6
000001E304873300	00 00 00 00 00 00 00 00	E0 70 81 B8 C7 49 26 40 01 12 F5 F8 3D 10 B6
000001E304873310	73 65 63 72 65 74 32 2E 74	62 BD 60 83 6D A7 31 D3 C0 E1 FB 74 05 DC 58 A1
000001E304873320	41 41 41 41 41 41 41 41	78 C1 50 1E 9E 7A D7 BD F2 17 9C 08 48 22 29 67
000001E304873330	41 41 41 41 41 41 41 41	4A E8 6E 4A 95 4B AD D0 F3 D8 1A 93 5A F7 F6 16
000001E304873340	41 41 41 41 41 41 41 41	6A A2 56 54 58 7F 29 46 C1 C4 81 29 1F 7C CA A2
000001E304873350	41 41 41 41 41 41 00 0A 41	54 11 00 83 D9 25 29 CA 39 51 08 E4 A3 E6 60 0C
000001E304873360	41 41 41 41 41 41 41 41	7F 77 45 0D BE 89 C9 2C 63 74 9D 79 67 63 2D EE
000001E304873370	41 41 41 41 00 0A 41 41	0E 69 9A 4B 77 F1 08 75 EE 87 79 2E 0E 57 9F D9
000001E304873380	41 41 41 41 41 41 41 41	7E 31 D8 66 32 58 3D FF 3B 2C 14 9F 56 18 83 24
000001E304873390	41 41 0D 0A 41 41 41 41	3E F7 70 A9 57 60 33 A0 0E 45 CB 3F 84 96 2D 98
000001E3048733A0	41 41 41 41 41 41 41 41	26 44 0D 30 43 55 77 38 76 B9 F0 09 A7 3A 8C D4
000001E3048733B0	0D 0A 41 41 41 41 41 41	49 0A A6 6A 9C 11 22 B2 C6 8A FC C8 9A EE 85
000001E3048733C0	41 41 41 41 41 41 41 41	78 0D 64 B8 87 63 7E 07 25 03 DD 41 41 41 41 41
000001E3048733D0	41 41 41 41 41 41 41 41	41 41 41 41 0D 0A 41 41 41 41 41 41 41 41 41
000001E3048733E0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E3048733F0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873400	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873410	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873420	41 41 41 41 41 41 0D 0A 41	41 41 41 41 41 41 41 41 41 41 41 41 0D 0A 41 41
000001E304873430	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873440	41 41 41 41 0D 0A 41 41	41 41 41 41 41 41 41 41 41 41 41 41 0D 0A 41 41
000001E304873450	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873460	41 41 0D 0A 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873470	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873480	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873490	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E3048734A0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E3048734B0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E3048734C0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E3048734D0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E3048734E0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E3048734F0	41 41 41 41 41 41 0D 0A 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873500	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873510	41 41 41 41 0D 0A 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
000001E304873520	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41

☐ Keep Size

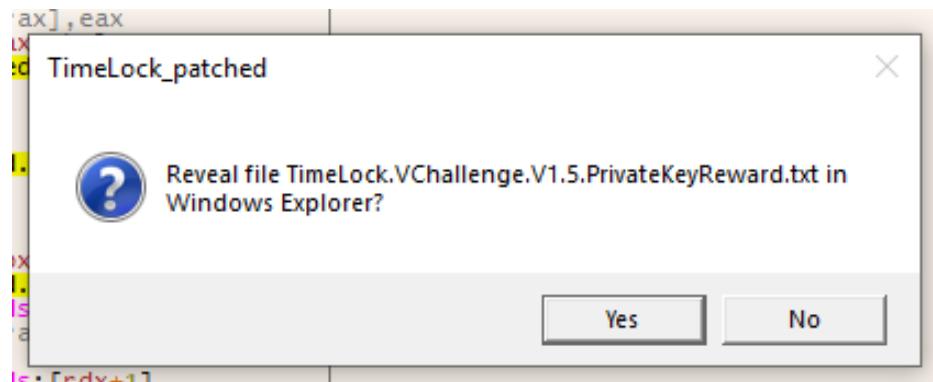
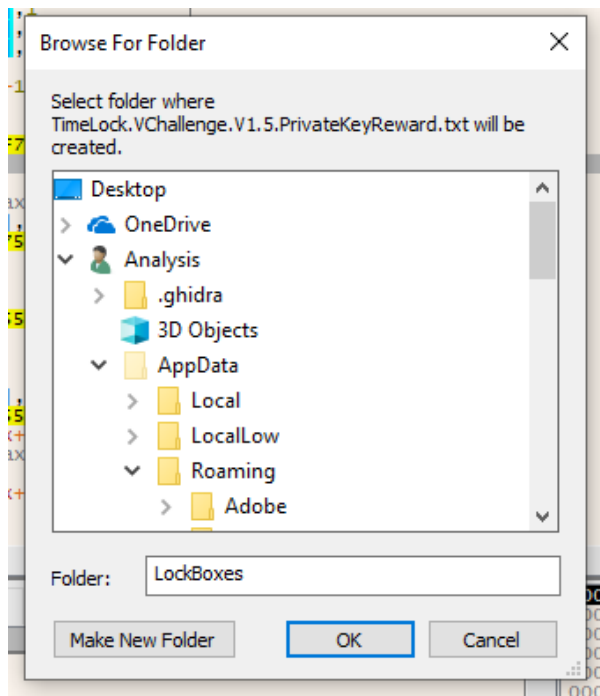
Address	Hex	ASCII
000001E3048732A0	AB AB AB AB AB AB AB AB	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048732B0	00 00 00 00 00 00 00 00	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048732C0	00 00 00 00 00 00 00 00	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048732D0	3F E6 AB C7 D1 17 0F 65 38 8A 59 00	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048732E0	85 2D 05 F0 A2 18 5D E6 B3 6D 37 99 F4	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048732F0	FE 5D 03 54 37 EE 63 86 D4 5C AF 3B 6F	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873300	85 05 C5 E9 92 25 C1 9D 32 21 E8 BF 60	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873310	01 AF E7 32 29 85 49 0E 86 C0 4B 8C 03	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873320	E3 E1 C8 C0 28 8A 02 79 21 F7 8F 14 06	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873330	41 40 99 9F 18 A1 BF D5 58 04 9E 58 8A	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873340	10 09 B5 D6 F5 AB AC 74 AC AA D7 FE 24	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873350	74 53 53 FB 2F C5 7E 45 CF 4F 72 F1 84	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873360	26 11 B5 E4 7D D8 88 5B FF 41 9A 15 F1	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873370	01 1E E3 1C 28 BE 87 6E 98 B4 58 CD 76	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873380	77 61 99 08 BD 1A 47 D4 FC D5 05 6F F0	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873390	C0 B2 E5 99 1C AE 1D D4 F8 69 19 14 53	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048733A0	C0 61 C4 ED 1A A5 C0 23 E8 0D 3E 07 CF	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048733B0	13 8E 5E 97 77 76 AF 60 CA FF 44 A5 78	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048733C0	DC FB 60 F9 41 4C AD 8D 34 F7 D4 36 70	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048733D0	FF 9B AC 81 C3 D0 F0 3A 58 95 34 7B EF	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048733E0	92 AA 49 29 E0 AF 58 82 42 10 93 A5 2D	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048733F0	C0 F6 73 2D 65 B1 1F 1D 68 56 48 F1 ED	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873400	3A 49 EF A5 0C DE C6 FD C4 98 F0 48 2F	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873410	75 B0 45 B3 31 03 CC F9 11 F3 7C 58 31	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873420	7A 4E E9 27 2B E8 30 29 70 08 0E 4E 0A	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873430	B6 A5 A1 02 48 64 82 6A F2 8A 27 FA 70	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873440	E0 70 81 B8 C7 49 26 40 01 12 F5 F8 3D	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873450	62 BD 60 83 6D A7 31 D3 C0 E1 FB 74 05	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873460	78 C1 50 1E 9E 7A D7 BD F2 17 9C 08 48	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873470	4A E8 6E 4A 95 4B AD D0 F3 D8 1A 93 5A	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873480	6A A2 56 54 58 7F 29 46 C1 C4 81 29 1F	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873490	54 11 00 83 D9 25 29 CA 39 51 08 E4 A3	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048734A0	7F 77 45 0D BE 89 C9 2C 63 74 9D 79 67	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048734B0	0E 69 9A 4B 77 F1 08 75 EE 87 79 2E 0E	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048734C0	7E 31 D8 66 32 58 3D FF 3B 2C 14 9F 56	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048734D0	3E F7 70 A9 57 60 33 A0 0E 45 CB 3F 84	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048734E0	26 44 0D 30 43 55 77 38 76 B9 F0 09 A7	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E3048734F0	49 0A A6 6A 9C 11 22 B2 C6 8A FC C8 9A	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873500	78 0D 64 B8 87 63 7E 07 25 03 DD 41 41	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873510	41 41 41 41 0D 0A 41 41	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000001E304873520	41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Command:

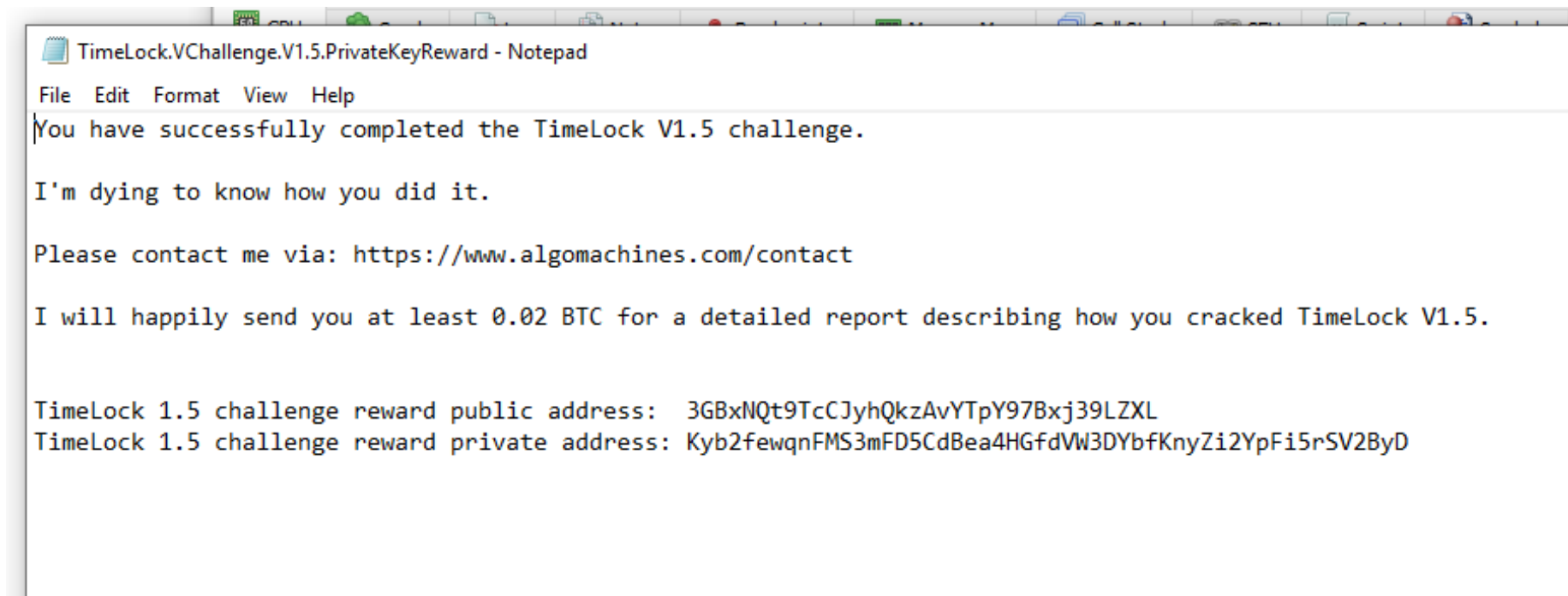
# Success! Symmetric Encryption Used!

Address	Hex	ASCII
000001E3048732A0	AB AB AB AB AB AB AB AB AB AB 00 00 00 00 00 00	««««««««««.....
000001E3048732B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001E3048732C0	00 00 00 00 00 00 00 00 00 00 A0 62 3F B2 72 48 28 32	..... b?°PK(2
000001E3048732D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001E3048732E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001E3048732F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001E304873300	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000001E304873310	54 69 6D 65 4C 6F 63 68 2E 56 43 68 61 6C 6C 65	TimeLock.VChalle
000001E304873320	6E 67 65 2E 56 31 2E 35 2E 50 72 69 76 61 74 65	nge.V1.5.Private
000001E304873330	4B 65 79 52 65 77 61 72 64 2E 74 78 74 00 59 6F	KeyReward.txt.Yo
000001E304873340	75 20 68 61 76 65 20 73 75 63 63 65 73 73 66 75	u have successfu
000001E304873350	6C 6C 79 20 63 6F 6D 70 6C 65 74 65 64 20 74 68	lly completed th
000001E304873360	65 60 54 69 6D 65 4C 6F 63 68 20 56 31 2E 35 20	e TimeLock V1.5
000001E304873370	63 68 61 6C 6C 65 6E 67 65 2E 0D 0A 0D 0A 49 27	challenge.....I'
000001E304873380	6D 20 64 79 69 6E 67 20 74 6F 20 68 6E 6F 77 20	m dying to know
000001E304873390	68 6F 77 20 79 6F 75 20 64 69 64 20 69 74 2E 0D	how you did it..
000001E3048733A0	0A 0D 0A 50 6C 65 61 73 65 20 63 6F 6E 74 61 63	...Please contac
000001E3048733B0	74 20 6D 65 20 76 69 61 3A 20 68 74 74 70 73 3A	t me via: https:
000001E3048733C0	2F 2F 77 77 77 2E 61 6C 67 6F 6D 61 63 68 69 6E	//www.algomachin
000001E3048733D0	65 73 2E 63 6F 6D 2F 63 6F 6E 74 61 63 74 0D 0A	es.com/contact..
000001E3048733E0	0D 0A 49 20 77 69 6C 6C 20 68 61 70 70 69 6C 79	...I will happily
000001E3048733F0	20 73 65 6E 64 20 79 6F 75 20 61 74 20 6C 65 61	send you at lea
000001E304873400	73 74 20 30 2E 30 32 20 42 54 43 20 66 6F 72 20	st 0.02 BTC for
000001E304873410	61 20 64 65 74 61 69 6C 65 64 20 72 65 70 6F 72	a detailed repor
000001E304873420	74 20 64 65 73 63 72 69 62 69 6E 67 20 68 6F 77	t describing how
000001E304873430	20 79 6F 75 20 63 72 61 63 68 65 64 20 54 69 6D	you cracked Tim
000001E304873440	65 4C 6F 63 6B 20 56 31 2E 35 2E 0D 0A 0D 0A 0D	eLock V1.5.....
000001E304873450	0A 54 69 6D 65 4C 6F 63 6B 20 31 2E 35 20 63 68	.TimeLock 1.5 ch
000001E304873460	61 6C 6C 65 6E 67 65 20 72 65 77 61 72 64 20 70	allenge reward p
000001E304873470	75 62 6C 69 63 20 61 64 64 72 65 73 73 3A 20 20	ublic address:
000001E304873480	73 47 42 78 4E 51 74 39 54 63 43 4A 79 68 51 68	3GBxNQt9TcCjyhQk
000001E304873490	3A 41 76 59 54 70 59 39 37 42 78 6A 33 39 4C 5A	zAvYTpY97Bxj39LZ
000001E3048734A0	58 4C 0D 0A 54 69 6D 65 4C 6F 63 68 20 31 2E 35	XL.TimeLock 1.5
000001E3048734B0	20 63 68 61 6C 6C 65 6E 67 65 20 72 65 77 61 72	challenge rewar
000001E3048734C0	64 20 70 72 69 76 61 74 65 20 61 64 64 72 65 73	d private address
000001E3048734D0	73 3A 20 48 79 62 32 66 65 77 71 6E 46 4D 53 33	s: Kyb2fewqnFMS3
000001E3048734E0	6D 46 44 35 43 64 42 65 61 34 48 47 66 64 56 57	mFD5CdBea4HGfDwV
000001E3048734F0	33 44 59 62 66 48 6E 79 5A 69 32 59 70 46 69 35	3DybfKnyZi2YpFi5
000001E304873500	72 53 56 32 42 79 44 0D 0A 0D 0A F2 49 08 9E DD	rSV2ByD.....öI..Y
000001E304873510	A2 89 2A 97 41 17 48 98 F1 95 6C 93 F2 9F 33 C9	C.*.A.H.ñ.ø.ö.3É
000001E304873520	4B 01 23 54 88 0A 5F 34 0A 95 03 28 F3 02 4F 2F	.....

# Looks Promising...



# Loot #4





# Lessons Learned:

## Vulnerability:

- Symmetric encryption means we could bypass decryption step completely.
- Ability to create lockboxes means we can use known plaintext attacks to find vulnerabilities faster.

## How to fix:

- Symmetric encryption does not provide any security if the secrets needed to make keys are available
- Public – Private key based encryption schemes should be used instead.
- A trusted third party should be the only one with access to decryption keys.

# Challenge #5

## TimeLock your digital assets (V1.7 / Challenge #5)

### RELEASE

Safely pass your digital assets on to your loved ones

- Securely lock your data until a time you choose.
- Create LockBoxes up to 10KB.
- TimeLock synced to the Bitcoin Network, using immutable block header timestamp.
- Retain privacy. Your data stays on your computer and nowhere else.
- Distribute your time locked LockBox to whomever you wish.

<https://www.algomachines.com/>

This is the seventh major version of TimeLock, and the second version anchored to the immutable timestamp of the Bitcoin network.

TimeLock has been improved via a series of challenges. You can see the reports on these challenges here:

<https://ruffell.nz/reverse-engineering/writeups/2019/01/18/timelock-analysis-and-vulnerability-writeup.html>

<https://ruffell.nz/reverse-engineering/writeups/2019/01/28/revisiting-timelock-1-2-vulnerability-writeup.html>

<https://ruffell.nz/reverse-engineering/writeups/2019/02/18/unleashing-a-sybil-attack-against-timelock-1-3-vulnerability-writeup.html>

<https://ruffell.nz/reverse-engineering/writeups/2019/03/20/double-trouble-with-symmetric-encryption-in-timelock-1-5-vulnerability-writeup.html>

The program is easy to use, and the free version supports LockBoxes of up to 10 Kbytes.



# Plan of Attack

- Review crypto again:
  - See if symmetric encryption is still used.
  - Look for weak algorithms.
  - Look for bad modes of encryption.

# Locate Encryption Functions

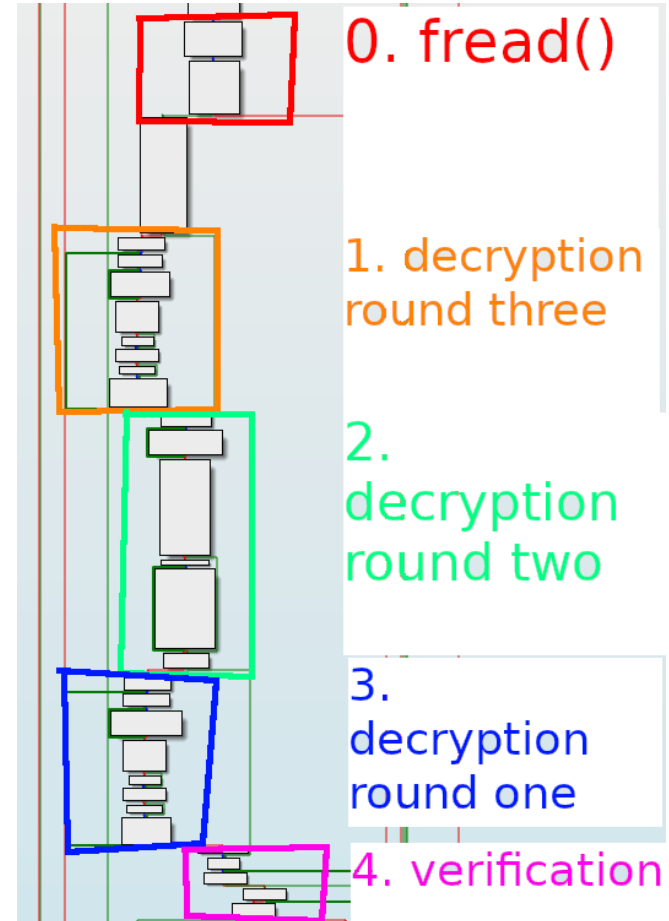
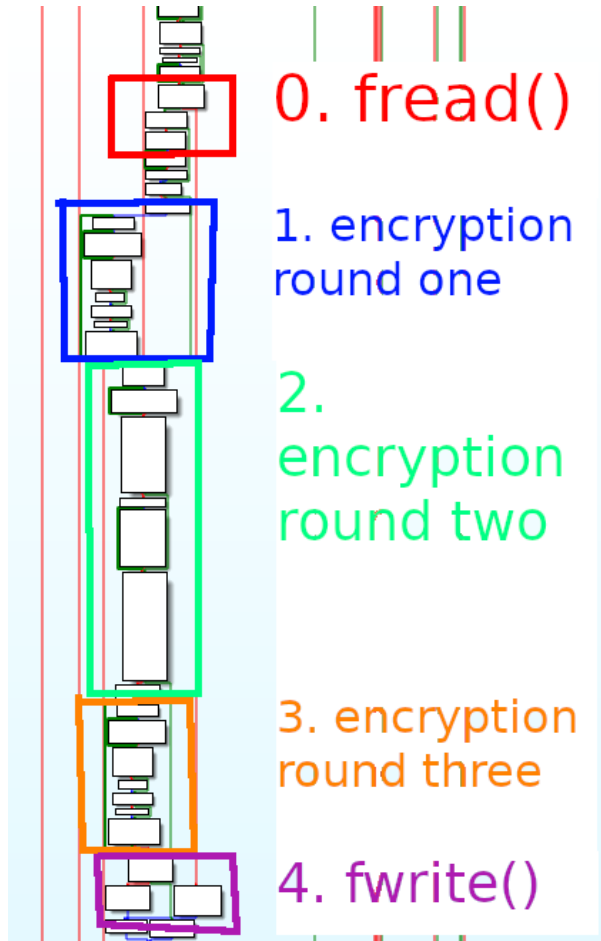
```
00007FF7EFB8EADB loc_7FF7EFB8EADB:
00007FF7EFB8EADB movsxd rcx, ecx
00007FF7EFB8EADDE add rcx, 41h
00007FF7EFB8EAE2 add rcx, r15 ; DstBuf
00007FF7EFB8EAE5 mov r9, [rsp+9C0h+var_970] ; File
00007FF7EFB8EAEA mov r8, r14 ; Count
00007FF7EFB8EAEED mov edx, 1 ; ElementSize
00007FF7EFB8EAF2 call fread
00007FF7EFB8EAF7 cmp rax, r14
00007FF7EFB8EAF7 jz short loc_7FF7EFB8EB40
```

```
00007FF7EFB8EB40 loc_7FF7EFB8EB40: ; File
00007FF7EFB8EB40 mov rcx, [rsp+9C0h+var_970]
00007FF7EFB8EB45 call fclose
00007FF7EFB8EB4A mov ebx, edi
00007FF7EFB8EB4C and ebx, 3
00007FF7EFB8EB4F jz short loc_7FF7EFB8EB86
```

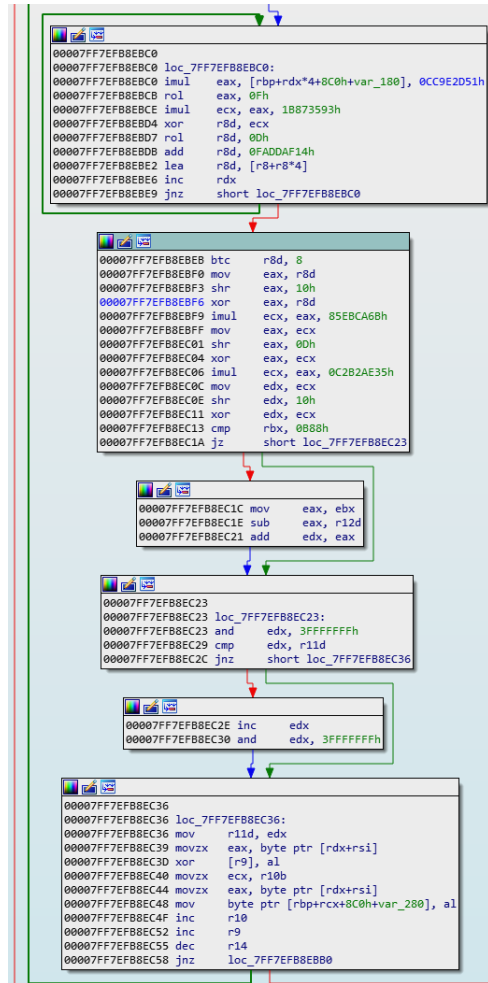
The screenshot shows a debugger window with the following components:

- Assembly View:** Displays assembly code for the function `timeLock_patched.7FF7EFD20E58`. The code includes instructions like `mov r9, qword ptr ss:[rsp+50]`, `mov r8, r14`, `mov edx, 1`, `call timeLock_patched.7FF7EFD20E58`, `cmp rax, r14`, `je timeLock_patched.7FF7EFB8EB40`, `mov rcx, r15`, `call timeLock_patched.7FF7EFB93990`, `mov rcx, r15`, `call timeLock_patched.7FF7EFB93990`, and `mov rcx, qword ptr ss:[rsp+30]`. The instruction `call timeLock_patched.7FF7EFB8EB40` is highlighted in red.
- Registers:** Shows the state of registers `rax=139` and `r14=139`.
- Memory Dump:** Displays a memory dump starting at address `0000022064EA6116`. The dump shows hex values and their corresponding ASCII representations, including `YBU.FM.`, `a.txt.`, and several lines of `AAAAAAAAAAAAAAAA`.

# Encryption and Decryption



# Encryption Rounds One and Three



## Rounds One and Three are Unchanged, Still Symmetric

[illegible]

Address	Hex	ASCII															
0000020649A9010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....															
0000020649A9020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....															
0000020649A9030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....															
0000020649A9040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....															
0000020649A9050	61 2E 74 78 74 00 41 41 41 41 41 41 41 41 41 41	a.txt.aaaaaaaaa															
0000020649A9060	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	aaaaaaaaaaaaaaaa															
0000020649A9070	41 41 0D 0A 41 41 41 41 41 41 41 41 41 41 41 41	aa, .aaaaaaaaaaa															
0000020649A9080	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	aaaaaaaaaaaaaaaa															
0000020649A9090	0D 0A 41 41 41 41 41 41 41 41 41 41 41 41 41 41	, .aaaaaaaaaaaaa															
0000020649A90A0	41 41 41 41 41 41 41 41 41 41 41 41 41 0D 0A	aaaaaaaaaaaaaaaa															
0000020649A90B0	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	aaaaaaaaaaaaaaaa															
0000020649A90C0	41 41 41 41 41 41 41 41 41 41 41 41 0D 0A 41 41	aaaaaaaaaaaaaa, .aa															
0000020649A90D0	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	aaaaaaaaaaaaaaaa															
0000020649A90E0	41 41 41 41 41 41 41 41 41 41 41 0D 0A 41 41 41	aaaaaaaaaaaaaa, .aaa															
0000020649A90F0	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	aaaaaaaaaaaaaaaa															
0000020649A9100	41 41 41 41 41 41 41 41 41 41 0D 0A 41 41 41 41	aaaaaaaaaaaaaa, .AAAA															
0000020649A9110	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	aaaaaaaaaaaaaa, .AAAAA															
0000020649A9120	41 41 41 41 41 41 41 41 0D 0A 41 41 41 41 41 41	aaaaaaaaaaaaaa, .AAAAAA															
0000020649A9130	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAA, .AAAAAA															
0000020649A9140	41 41 41 41 0D 0A 41 41 41 41 41 41 41 41 41 41	AAAA, .AAAAAA															
0000020649A9150	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAA															
0000020649A9160	41 41 0D 0A 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAA															
0000020649A9170	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAA															
0000020649A9180	0D 0A 41 41 41 41 41 41 41 41 41 41 41 41 41 0D	AAAAAAAAAAAAAA															
0000020649A9190	00 00 AB AB AB AB AB AB AB AB AB AB AB AB AB AB	«»															
0000020649A91A0	AB AB EE FE EE FE EE FE EE FE EE FE EE FE EE FE EE	«»pibipibipibip															

The diagram illustrates the assembly code of the `nt!KeBugCheckEx` function, showing the flow of execution and the handling of various registers and memory locations. The code is organized into several sections, each with a title and a list of instructions. Arrows indicate the flow of execution between these sections.

**Section 1: Initial Setup and Register Manipulation**

```

00007FF7EFB8EAD xor     r8d, 8
00007FF7EFB8EB1 mov     eax, r8d
00007FF7EFB8EB4 shr     eax, 10h
00007FF7EFB8EB7 mov     eax, r8d
00007FF7EFB8ECA imul    ecx, eax, 85ECA6Bh
00007FF7EFB8ECB mov     ecx, ecx
00007FF7EFB8EC2 shr     eax, 00h
00007FF7EFB8EC5 mov     ecx, eax
00007FF7EFB8EC7 imul    ecx, eax, 0C282AE35h
00007FF7EFB8EC0 mov     ecx, ecx
00007FF7EFB8ECF shr     ebx, 10h
00007FF7EFB8ED2 mov     ecx, ecx
00007FF7EFB8ED4 mov     ecx, edi
00007FF7EFB8ED6 shr     rcx, 1
00007FF7EFB8ED9 mov     eax, 4
00007FF7EFB8EDE mul     rcx
00007FF7EFB8EE1 cmovb  rax, r13
00007FF7EFB8EE5 mov     rcx, rax
00007FF7EFB8EE8 call   malloc_wrapper
00007FF7EFB8EE0 mov     r13, rax
00007FF7EFB8EEF mov     [rbp+8C0h+var_918], rax
00007FF7EFB8EF4 mov     r14, rax
00007FF7EFB8EF7 mov     r8d, ebx
00007FF7EFB8EFA mov     rax, 50107E74251C8553h
00007FF7EFB8ED4 imul    r8
00007FF7EFB8ED7 sub     rdx, r8
00007FF7EFB8EDA sub     rdx, 9
00007FF7EFB8EDE mov     rcx, rdx
00007FF7EFB8ED1 shr     rcx, 3Fh
00007FF7EFB8ED5 add     rdx, rcx
00007FF7EFB8ED8 imul    rax, rdx, 2E9h
00007FF7EFB8EDF add     r8, rax
00007FF7EFB8ED2 mov     eax, 1
00007FF7EFB8ED7 cmovb  r8, rax
00007FF7EFB8ED8 xorps  xmm8, xmm8
00007FF7EFB8ED2 cvtsi2ss xmm8, r8
00007FF7EFB8ED3 divss  xmm8, cs:dword_7FF7EFDB478
00007FF7EFB8ED3 edi, 2
00007FF7EFB8ED4 test  edi, edi
00007FF7EFB8ED4 jle     short loc_7FF7EFB8EDC1

```

**Section 2: Memory Manipulation and Control Flow**

```

00007FF7EFB8ED4 mov     ebx, edi
00007FF7EFB8ED4 db     66h, 66h
00007FF7EFB8ED4 nop     word ptr [rax+rax+00000000h]

```

**Section 3: Register Manipulation and Control Flow**

```

00007FF7EFB8ED4 mov     ebx, edi
00007FF7EFB8ED4 db     66h, 66h
00007FF7EFB8ED4 nop     word ptr [rax+rax+00000000h]

```

**Section 4: Register Manipulation and Control Flow**

```

00007FF7EFB8ED50 loc_7FF7EFB8ED50:
00007FF7EFB8ED50 mov     eax, [r15]
00007FF7EFB8ED53 mov     [rbp+8C0h+var_918], eax
00007FF7EFB8ED56 mov     r8d, 4 ; Size
00007FF7EFB8ED5C lea     rdx, [rbp+8C0h+var_918] ; Src
00007FF7EFB8ED60 lea     rcx, [rsp+9C0h+var_960] ; Dst
00007FF7EFB8ED65 call   memmove
00007FF7EFB8ED6A movzx  eax, [rsp+9C0h+var_960]
00007FF7EFB8ED6F movzx  xmm7, eax
00007FF7EFB8ED73 cvtdq2ps xmm7, xmm7
00007FF7EFB8ED76 movzx  eax, [rsp+9C0h+var_95E]
00007FF7EFB8ED78 movzx  xmm6, eax
00007FF7EFB8ED7F cvtdq2ps xmm6, xmm6
00007FF7EFB8ED82 movaps  xmm0, xmm6
00007FF7EFB8ED85 mulss  xmm0, xmm6
00007FF7EFB8ED89 movaps  xmm2, xmm7
00007FF7EFB8ED8C mulss  xmm2, xmm7
00007FF7EFB8ED90 addss  xmm0, xmm2 ; X
00007FF7EFB8ED94 call   sqrtf
00007FF7EFB8ED99 movaps  dword ptr [r14], xmm0
00007FF7EFB8EDA1 movaps  xmm1, xmm7 ; X
00007FF7EFB8EDA1 movaps  xmm6, xmm6 ; Y
00007FF7EFB8EDA4 call   atan2f
00007FF7EFB8EDA9 addss  xmm0, xmm8
00007FF7EFB8EDAE movss  dword ptr [r14+4], xmm0
00007FF7EFB8EDB4 add     r14, 8
00007FF7EFB8EDB8 add     r15, 4
00007FF7EFB8EDBC dec     rbx
00007FF7EFB8EDBF jnz     short loc_7FF7EFB8ED50

```

**Section 5: Register Manipulation and Control Flow**

```

00007FF7EFB8EDC1 loc_7FF7EFB8EDC1:
00007FF7EFB8EDC1 mov     rax, [rsp+9C0h+var_958]
00007FF7EFB8EDC6 mov     [rcx], r12d
00007FF7EFB8EDC9 add     rcx, cs:qword_7FF7EFDB2D78
00007FF7EFB8EDD0 mov     r9, 87C3B911425305h
00007FF7EFB8EDD4 imul    rcx, r9
00007FF7EFB8EDDE rol     rcx, 1Fh
00007FF7EFB8EDE2 mov     rdx, 4CF5AD432745937Fh
00007FF7EFB8EDEE imul    rcx, rdx
00007FF7EFB8EDF0 mov     rcx, 48F73C39h
00007FF7EFB8EDF7 rol     rcx, 18h
00007FF7EFB8EDFB mov     rax, 1BF811446h
00007FF7EFB8EE05 add     rax, rcx
00007FF7EFB8EE08 lea     r8, [rax+rcx*4]
00007FF7EFB8EE0C mov     rax, cs:qword_7FF7EFDB2D80
00007FF7EFB8EE13 imul    rax, rdx
00007FF7EFB8EE17 rol     rax, 21h
00007FF7EFB8EE18 imul    rax, r9
00007FF7EFB8EE1F xor     rax, 48F73C39h
00007FF7EFB8EE25 rol     rax, 1Fh
00007FF7EFB8EE29 add     rax, 084DEF1Fh
00007FF7EFB8EE2F add     rax, r8
00007FF7EFB8EE32 lea     rcx, [rax+rax*4]
00007FF7EFB8EE36 xor     rcx, 10h
00007FF7EFB8EE3A xor     r8, 10h
00007FF7EFB8EE3E add     r8, rcx
00007FF7EFB8EE41 mov     rax, r8
00007FF7EFB8EE44 shr     rax, 21h
00007FF7EFB8EE48 xor     rax, r8
00007FF7EFB8EE4E mov     r9, 0FF51AFD7ED558CCDh
00007FF7EFB8EE55 imul    rax, r9
00007FF7EFB8EE59 mov     rdx, rax
00007FF7EFB8EE5C shr     rdx, 21h
00007FF7EFB8EE60 xor     rdx, rax
00007FF7EFB8EE63 mov     r10, 0C4CEB9E1A85EC53h
00007FF7EFB8EE66 imul    rdx, r10
00007FF7EFB8EE71 lea     rax, [rcx+r8]
00007FF7EFB8EE75 mov     rcx, rax
00007FF7EFB8EE78 shr     rcx, 21h
00007FF7EFB8EE7C xor     rcx, rax
00007FF7EFB8EE7F imul    rcx, r9
00007FF7EFB8EE83 mov     rax, rcx
00007FF7EFB8EE86 shr     rax, 21h
00007FF7EFB8EE8A xor     rax, rcx
00007FF7EFB8EE8D imul    rax, r10
00007FF7EFB8EE91 mov     rcx, rax
00007FF7EFB8EE94 shr     rcx, 21h
00007FF7EFB8EE98 xor     rcx, rax
00007FF7EFB8EE9B mov     rax, rdx
00007FF7EFB8EE9E shr     rax, 21h
00007FF7EFB8EEA2 xor     rax, rdx
00007FF7EFB8EEA5 add     rax, rcx
00007FF7EFB8EEA8 add     rcx, rax
00007FF7EFB8EEAB mov     [rbp+8C0h+var_280], rax
00007FF7EFB8EEB2 mov     [rbp+8C0h+var_278], rcx
00007FF7EFB8EEB9 test  r12d, r12d
00007FF7EFB8EEBC jle     loc_7FF7EFB8EEF93

```

# Decryption Round Two

```
00007FF7EFB71880
00007FF7EFB71880 loc_7FF7EFB71880:
00007FF7EFB71880 mov rcx, cs:qword_7FF7EFDB2D78
00007FF7EFB71887 mov r8, 87C37891114253D5h
00007FF7EFB71891 imul rcx, r8
00007FF7EFB71895 rol rcx, 1Fh
00007FF7EFB71899 mov rdx, 4CF5AD432745937Fh
00007FF7EFB718A3 imul rcx, rdx
00007FF7EFB718A7 xor rcx, 48F73C39h
00007FF7EFB718AE rol rcx, 18h
00007FF7EFB718B2 mov rax, 1BF811446h
00007FF7EFB718B8 add rax, rcx
00007FF7EFB718BF lea rdi, [rax+rcx*4]
00007FF7EFB718C3 mov rax, cs:qword_7FF7EFDB2D80
00007FF7EFB718CA imul rax, rdx
00007FF7EFB718CE rol rax, 21h
00007FF7EFB718D2 imul rax, r8
00007FF7EFB718D6 xor rax, 48F73C39h
00007FF7EFB718DC rol rax, 1Fh
00007FF7EFB718E0 add rax, 0B41DEF1h
00007FF7EFB718E6 add rax, rdi
00007FF7EFB718E9 lea rcx, [rax+rax*4]
00007FF7EFB718ED xor rcx, 10h
00007FF7EFB718F1 xor rdi, 10h
00007FF7EFB718F5 add rdi, rcx
00007FF7EFB718F8 mov rax, rdi
00007FF7EFB718FB shr rax, 21h
00007FF7EFB718FF xor rax, rdi
00007FF7EFB71902 mov r8, 0FF51AFD7ED558CCDh
00007FF7EFB7190C imul rax, r8
00007FF7EFB71910 mov rdx, rax
00007FF7EFB71913 shr rdx, 21h
00007FF7EFB71917 xor rdx, rax
00007FF7EFB7191A mov r9, 0C4CEB9FE1A85EC53h
00007FF7EFB71924 imul rdx, r9
00007FF7EFB71928 lea rax, [rcx+rdi]
00007FF7EFB7192C mov rcx, rax
00007FF7EFB7192F shr rcx, 21h
00007FF7EFB71933 xor rcx, rax
00007FF7EFB71936 imul rcx, r8
00007FF7EFB7193A mov rax, rcx
00007FF7EFB7193D shr rax, 21h
00007FF7EFB71941 xor rax, rcx
00007FF7EFB71944 imul rax, r9
00007FF7EFB71948 mov rcx, rax
00007FF7EFB7194B shr rcx, 21h
00007FF7EFB7194F xor rcx, rax
00007FF7EFB71952 mov rax, rdx
00007FF7EFB71955 shr rax, 21h
00007FF7EFB71959 xor rax, rdx
00007FF7EFB7195C add rax, rcx
00007FF7EFB7195F add rcx, rax
00007FF7EFB71962 mov [rbp+117C0h+var_1C8], rax
00007FF7EFB71969 mov [rbp+117C0h+var_1C0], rcx
00007FF7EFB71970 mov ebx, r14d
00007FF7EFB71973 test r15d, r15d
00007FF7EFB71976 jle loc_7FF7EFB71A38

00007FF7EFB71A77 xor edi, 8
00007FF7EFB71A7A mov eax, edi
00007FF7EFB71A7C shr eax, 10h
00007FF7EFB71A7F xor eax, edi
00007FF7EFB71A81 imul ecx, eax, 85EBCA6Bh
00007FF7EFB71A87 mov eax, ecx
00007FF7EFB71A89 shr eax, 00h
00007FF7EFB71A8C xor eax, ecx
00007FF7EFB71A8E imul ecx, eax, 0C2B2AE35h
00007FF7EFB71A94 mov edi, ecx
00007FF7EFB71A96 shr edi, 10h
00007FF7EFB71A99 xor edi, ecx
00007FF7EFB71A9B mov rax, r15
00007FF7EFB71A9E cqv
00007FF7EFB71AA0 sub rax, rdx
00007FF7EFB71AA3 sar rax, 1
00007FF7EFB71AA6 mov r14, rax
00007FF7EFB71AA9 mov [rsp+118C0h+lpParameters], r14
00007FF7EFB71AAE mov rcx, rax
00007FF7EFB71AB1 call malloc_wrapper
00007FF7EFB71AB6 mov rbx, rax
00007FF7EFB71AB9 mov rsi, r12
00007FF7EFB71ABC mov rax, 50107E74251C8553h
00007FF7EFB71AC6 imul rdi
00007FF7EFB71AC9 sub rdx, rdi
00007FF7EFB71ACC sar rdx, 9
00007FF7EFB71AD0 mov rcx, rdx
00007FF7EFB71AD3 shr rcx, 3Fh
00007FF7EFB71AD7 add rdx, rcx
00007FF7EFB71ADA imul rax, rdx, 2E9h
00007FF7EFB71AE1 add rdi, rax
00007FF7EFB71AE4 mov eax, 1
00007FF7EFB71AE9 rdi, rax
00007FF7EFB71AE0 xorps xmm0, xmm0
00007FF7EFB71AF0 cvtsi2ss xmm0, rdi
00007FF7EFB71AF5 divss xmm0, cs:dword_7FF7EFDB4D78
00007FF7EFB71AFD movss xmm10, cs:dword_7FF7EFDB4D64
00007FF7EFB71B06 subss xmm10, xmm0
00007FF7EFB71B08 mov edi, edi
00007FF7EFB71B00 mov rax, r15
00007FF7EFB71B10 cqv
00007FF7EFB71B10 and edx, 7
00007FF7EFB71B15 add rax, rdx
00007FF7EFB71B18 sar rax, 3
00007FF7EFB71B1C mov r15, rax
00007FF7EFB71B1F test eax, eax
00007FF7EFB71B21 jle loc_7FF7EFB71BCF

00007FF7EFB71B30
00007FF7EFB71B30 loc_7FF7EFB71B30:
00007FF7EFB71B30 movss xmm8, dword ptr [rsi]
00007FF7EFB71B35 movss dword ptr [rbp+117C0h+var_117C8], xmm8
00007FF7EFB71B38 movss xmm7, dword ptr [rsi+4]
00007FF7EFB71B40 add rsi, 8
00007FF7EFB71B44 addss xmm7, xmm10
00007FF7EFB71B49 movss dword ptr [rbp+117C0h+var_117C8+4], xmm7
00007FF7EFB71B4E movaps xmm0, xmm7 ; X
00007FF7EFB71B51 call cosf
00007FF7EFB71B56 movaps xmm6, xmm0
00007FF7EFB71B59 movaps xmm0, xmm7 ; X
00007FF7EFB71B5C call sinf
00007FF7EFB71B61 movaps xmm7, xmm0
00007FF7EFB71B64 mulss xmm6, xmm8
00007FF7EFB71B69 xorps xmm1, xmm1
00007FF7EFB71B6C cvtss2sd xmm1, xmm6
00007FF7EFB71B70 andps xmm1, xmm9
00007FF7EFB71B74 cvtpd2ps xmm0, xmm1
00007FF7EFB71B78 call roundf
00007FF7EFB71B7D movaps xmm6, xmm0
00007FF7EFB71B80 mulss xmm7, xmm8
00007FF7EFB71B85 xorps xmm1, xmm1
00007FF7EFB71B88 cvtss2sd xmm1, xmm7
00007FF7EFB71B8C andps xmm1, xmm9
00007FF7EFB71B90 cvtpd2ps xmm0, xmm1
00007FF7EFB71B94 call roundf
00007FF7EFB71B99 cvtss2si rax, xmm6
00007FF7EFB71B9E mov word ptr [rsp+118C0h+var_11868], rax
00007FF7EFB71BA3 cvtss2si rax, xmm0
00007FF7EFB71BA8 mov word ptr [rsp+118C0h+var_11868+2], rax
00007FF7EFB71BAD movss rax, edi
00007FF7EFB71BB0 lea rcx, [rbx+rax*4] ; Dst
00007FF7EFB71BB4 mov r8d, 4 ; Size
00007FF7EFB71BB8 lea rdx, [rsp+118C0h+var_11868] ; Src
00007FF7EFB71BBF call memmove
00007FF7EFB71BC4 inc edi
00007FF7EFB71BC6 cmp edi, r15d
00007FF7EFB71BC9 jl loc_7FF7EFB71B30
```



# Encryption Round Two is Not Symmetric

Address	Hex	ASCII
000002206499A6F0	B3 02 5D 47 3D 85 10 40 ED A4 52 47 78 7D 12 40	■.]G=..@iRgX}.@
000002206499A700	8A FD 95 47 9E 50 2D 40 4B 51 5F 47 2F 42 1F 40	.ý.G.P-@KQ_G/B.@
000002206499A710	90 49 18 47 20 AE 2C 40 F0 1D 60 47 22 EF 38 40	.I.G °,@ð. G"i8@
000002206499A720	E4 A1 68 47 30 F6 50 40 5E 93 7D 47 0A D9 56 40	äihG0öP@^.]G.ÚV@
000002206499A730	22 F7 90 47 F4 80 3F 40 39 0E 98 47 1F 6B 33 40	"÷.Gô.?@9..G.k3@
000002206499A740	8D B1 5D 47 A1 E1 40 40 D5 7C 32 47 B9 E3 3D 40	.±]Giá@ëÖ 2G'ã=@
000002206499A750	3C 53 67 47 92 10 3F 40 D9 6C 79 47 21 B5 1C 40	<SgG..?@ÜlyG!µ.@
000002206499A760	03 A3 91 47 3F B0 39 40 84 1B 9C 47 30 DF 34 40	.f.G?°9@...G0ß4@
000002206499A770	0B E2 46 47 F4 87 10 40 34 23 83 47 E2 B7 49 40	.âFGô..@4#.Gâ.I@
000002206499A780	28 65 2E 47 D8 23 49 40 4A 18 B6 46 FA EC 42 40	(e.G0#I@J.¶FûiB@
000002206499A790	33 94 9B 47 BE E7 3F 40 15 4A 9C 47 33 FB 37 40	3..G%ç?@.J.G3û7@
000002206499A7A0	44 34 F2 46 92 D2 12 40 3A 27 9F 47 50 DC 30 40	D4òF.Ô.@:'.GPÜ0@
000002206499A7B0	42 2D AD 46 93 00 3A 40 76 EE 0E 47 4F F3 1F 40	B-.F...@vî.G0ó.@
000002206499A7C0	9C E1 83 46 B5 CF 12 40 DF 47 82 46 60 B2 2A 40	.â.Fµİ.@ßG.F`=*@
000002206499A7D0	BF 17 64 47 B0 49 37 40 E6 EF 23 47 7B 9D 0E 40	¿.dG°I7@æi#G{..@
000002206499A7E0	2F F4 13 47 56 21 55 40 55 A9 AB 46 B2 78 43 40	/ô.GV!U@U@«F=xC@
000002206499A7F0	AD 8E 21 47 AA F3 38 40 67 2B 27 47 A3 0C 19 40	..!G°ô8@ag+'Gf..@
000002206499A800	31 85 66 47 8A AB 63 40 65 B4 C6 46 4A A2 65 40	1.fG.«c@e'ÆFJc@e
000002206499A810	B4 63 7E 47 F3 27 2D 40 F9 72 3A 47 C5 C1 47 40	'c~Gó'~@ür:GÁÁ@
000002206499A820	2B F8 51 47 60 93 63 40 78 A3 3E 47 1B 65 48 40	+øQG`.c@xf>G.eH@
000002206499A830	8F 33 0A 47 60 75 0C 40 EB 12 4E 46 39 B6 0D 40	.3.G`u.@ë.NF9¶.@
000002206499A840	3E 2F 18 47 F1 E1 0E 40 CE 8F 0F 47 75 32 32 40	>/.Gñá.@İ..Gu22@
000002206499A850	5B 5E 69 47 7D 81 0D 40 DF 23 7B 47 56 CA 60 40	[^iG]..@ß#{GVÊ`@
000002206499A860	82 07 85 47 0B 8F 24 40 81 4F EE 46 73 32 21 40	...G...\$@.0íFs2!@
000002206499A870	CB 75 57 47 9B 0F 14 40 56 82 9F 47 8E 2B 35 40	ËuWG...@V..G.+5@
000002206499A880	13 3E 1B 47 C9 4C 38 40 D5 F2 6A 47 12 D2 4E 40	.>.GÉL8@ÖòjG.ÔN@



# Game Plan

- Peel off one layer of encryption at a time:
  - “encrypt” third round via symmetric encryption.
  - “decrypt” second round.
    - Hope for hardcoded constants, no time related secrets.
  - “encrypt” first round via symmetric encryption.

# Extracting Ciphertext, Third Round of Encryption

[illegible]

**Disassembly**

Address	Disassembly	Comment
00000000	mov byte ptr ds:[r9],al	
00000001	mov rcx,r10	
00000002	and ecx,f	
00000003	movzx eax,byte ptr ds:[rdx+rsi]	
00000004	mov byte ptr ss:[rbp+rcx+640],al	
00000005	inc r10	
00000006		
00000007		
00000008		
00000009		
0000000A		
0000000B		
0000000C		
0000000D		
0000000E		
0000000F		
00000010		
00000011		
00000012		
00000013		
00000014		
00000015		
00000016		
00000017		
00000018		
00000019		
0000001A		
0000001B		
0000001C		
0000001D		
0000001E		
0000001F		
00000020		
00000021		
00000022		
00000023		
00000024		
00000025		
00000026		
00000027		
00000028		
00000029		
0000002A		
0000002B		
0000002C		
0000002D		
0000002E		
0000002F		
00000030		
00000031		
00000032		
00000033		
00000034		
00000035		
00000036		
00000037		
00000038		
00000039		
0000003A		
0000003B		
0000003C		
0000003D		
0000003E		
0000003F		
00000040		
00000041		
00000042		
00000043		
00000044		
00000045		
00000046		
00000047		
00000048		
00000049		
0000004A		
0000004B		
0000004C		
0000004D		
0000004E		
0000004F		
00000050		
00000051		
00000052		
00000053		
00000054		
00000055		
00000056		
00000057		
00000058		
00000059		
0000005A		
0000005B		
0000005C		
0000005D		
0000005E		
0000005F		
00000060		
00000061		
00000062		
00000063		
00000064		
00000065		
00000066		
00000067		
00000068		
00000069		
0000006A		
0000006B		
0000006C		
0000006D		
0000006E		
0000006F		
00000070		
00000071		
00000072		
00000073		
00000074		
00000075		
00000076		
00000077		
00000078		
00000079		
0000007A		
0000007B		
0000007C		
0000007D		
0000007E		
0000007F		
00000080		
00000081		
00000082		
00000083		
00000084		
00000085		
00000086		
00000087		
00000088		
00000089		
0000008A		
0000008B		
0000008C		
0000008D		
0000008E		
0000008F		
00000090		
00000091		
00000092		
00000093		
00000094		
00000095		
00000096		

# Extracting Ciphertext, Second Round Decryption

Address	Hex																ASCII
000002377830FDF0	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	««««««««««««««««
000002377830FE00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000002377830FE10	EE	EE	EE	EE	EE	EE	EE	EE	4F	5C	72	81	9F	59	00	3C	ipipipipoAr...Y.
000002377830FE20	83	02	5D	47	3D	85	10	40	ED	A4	52	47	78	70	12	40	*,.jG.,.e[PRGX].
000002377830FE30	8A	FD	95	47	9E	50	2D	40	48	51	5F	47	2F	42	1F	40	Y.G.,P-«KQ_G/B.e.
000002377830FE40	90	49	18	47	20	AE	2C	40	F0	1D	60	47	22	EF	38	40	.I.G.e,e0.,.g8
000002377830FE50	E4	A1	68	47	30	F6	50	40	5E	93	7D	47	0A	D9	56	40	ajhGO0P«e.,.g8.UV0
000002377830FE60	22	F7	90	47	F4	80	3F	40	39	0E	98	47	1F	68	33	40	"-.G0.,?e9.,.G.k30
000002377830FE70	8D	B1	5D	47	A1	E1	40	40	D5	7C	32	47	B9	E3	3D	40	.zj G ae02G'«=0
000002377830FE80	3C	53	67	47	92	10	3F	40	D9	6C	79	47	21	B5	1C	40	<Sg.,.?uUlyG u.e.
000002377830FE90	03	A3	91	47	3F	B0	39	40	84	18	9C	47	30	DF	34	40	.E.f?79e.,.G0Bz40
000002377830FEA0	21	12	03	47	89	D5	08	40	D8	78	97	47	3E	02	34	40	!.,.G0.,e0j.f.G.40
000002377830FEB0	F1	C3	37	47	A8	98	58	40	7A	93	DD	46	6A	0A	5E	40	«A7G'[,eZ.YF]«
000002377830FEC0	67	06	9F	47	D2	16	32	40	0C	BE	AE	47	68	8F	37	40	g.,.G0.,2e.Gh.,.G
000002377830FED0	A4	D1	CA	46	10	8A	1C	40	E6	71	8D	47	48	2A	46	40	=NEF.,.e0j.GH?F0
000002377830FEE0	3C	D2	AA	46	F6	F7	60	40	D4	AE	3C	47	D4	04	25	40	=0F0«.,.e0.G0.%0
000002377830FEF0	20	9C	0E	46	35	23	1E	40	43	01	D5	45	15	6E	1C	40	.,.aF5#.,@C.OE.n.e.
000002377830FF00	15	76	68	47	DA	8A	43	40	7D	4F	1B	47	81	CA	1F	40	.vHgU.C«jo.G.E.e.
000002377830FF10	2C	68	6A	47	37	26	65	40	86	77	10	47	D6	52	45	40	,k,jG7«e.w.G0RE0
000002377830FF20	36	7E	EC	45	95	6F	34	40	77	91	62	47	7C	92	20	40	6-7E.0«w.bGj.
000002377830FF30	1A	FE	3D	47	F2	92	4C	40	2F	39	85	46	D5	53	6A	40	=p-G0.,LE.F.0F0jG
000002377830FF40	60	11	71	47	E2	77	36	40	90	CB	3E	47	E0	17	4F	40	.,qGaw6E.E«G«0.
000002377830FF50	D0	7C	38	47	EA	13	56	40	F7	3C	4F	47	AC	A4	53	40	Dj ;G0.,v«e«G«=50
000002377830FF60	5F	78	3A	47	83	4F	1A	40	B9	AB	8D	46	1F	53	28	40	«X:G«0.,e«%F.F0
000002377830FF70	FC	FD	58	47	E4	CA	29	40	44	C4	AF	46	2D	32	36	40	ux G40j«DA.F-260
000002377830FF80	1E	11	4D	47	63	10	16	40	9A	8C	19	47	80	3F	5F	40	.,Mgc.,e...G?«0

[illegible]

# Extracting Ciphertext and First Round of Encryption

Assembly view of a program patch showing instructions and registers. The instructions are in x86 assembly, and the registers are in the right column. The patch is for the function `timeLock_patched.exe`.

Instructions and registers:

- `xor byte ptr ds:[r9],al` (41:3001)
- `movzx ecx,r10b` (41:0F8CA)
- `movzx eax,byte ptr ds:[rdx+rsi]` (0F860432)
- `mov byte ptr ss:[rbp+rcx+640],al` (888400 40060000)
- `inc r10` (49:FFC2)
- `inc r9` (49:FFC1)
- `inc r9` (49:FFC0)

Registers:

- `41:3001`
- `41:0F8CA`
- `0F860432`
- `888400 40060000`
- `49:FFC2`
- `49:FFC1`
- `49:FFC0`

Hex dump of the patch data:

Address	Hex
000002377D0E0760	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E0770	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E0780	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E0790	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E07A0	43 68 61 6C 6C 65 6E 67 65 35 52 65 77 61 72 64
000002377D0E07B0	2E 74 78 74 00 59 6F 75 20 68 61 76 65 20 63 6F
000002377D0E07C0	6D 70 6C 65 74 65 64 20 54 69 6D 65 4C 6F 63 68
000002377D0E07D0	20 63 68 61 6C 6C 65 6E 67 65 20 23 35 20 75 73
000002377D0E07E0	69 6E 67 20 56 31 2E 37 2E 0D 0A 0D 0A 49 20 61
000002377D0E07F0	6D 20 69 6D 70 72 65 73 73 65 64 20 77 69 74 68
000002377D0E0800	20 79 6F 75 72 20 73 68 69 6C 6C 20 61 6E 64 20
000002377D0E0810	61 6E 78 69 6F 75 73 20 74 6F 20 75 6E 64 65 72
000002377D0E0820	73 74 61 6E 64 20 68 6F 77 20 79 6F 75 20 61 63
000002377D0E0830	63 6F 6D 70 6C 69 73 68 65 64 20 74 68 69 73 21
000002377D0E0840	0D 0A 0D 0A 43 68 61 6C 6C 65 6E 67 65 20 52 65
000002377D0E0850	77 61 72 64 3A 20 0D 0A 0D 0A 42 54 43 20 50 75
000002377D0E0860	62 6C 69 63 20 41 64 64 72 65 73 73 3A 20 0D 0A
000002377D0E0870	33 4E 68 68 42 65 4C 39 7A 37 66 62 72 48 76 47
000002377D0E0880	4C 58 51 79 7A 52 43 45 76 43 66 44 57 35 6E 51
000002377D0E0890	54 51 0D 0A 0D 0A 42 54 43 20 50 72 69 76 61 74
000002377D0E08A0	65 20 41 64 64 72 65 73 73 3A 20 0D 0A 4C 32 73
000002377D0E08B0	62 47 42 33 4C 68 46 64 67 43 34 48 34 4A 55 54
000002377D0E08C0	44 39 44 57 74 34 6F 53 7A 52 78 50 62 6A 59 7A
000002377D0E08D0	6A 35 75 70 4B 47 68 68 54 4D 69 79 71 34 71 36
000002377D0E08E0	3D 47 AB AB AB AB AB AB AB AB AB AB AB AB AB AB
000002377D0E08F0	AB AB EE FE EE FE EE FE EE FE EE FE EE FE EE FE
000002377D0E0900	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E0910	EE FE EE FE EE FE EE FE 73 C5 72 BD A7 59 10 30
000002377D0E0920	50 A8 91 C9 F8 7F 00 00 70 4A 15 7D 37 02 00 00
000002377D0E0930	70 4A 15 7D 37 02 00 00 10 4A 15 7D 37 02 00 00

String dump of the patch data:

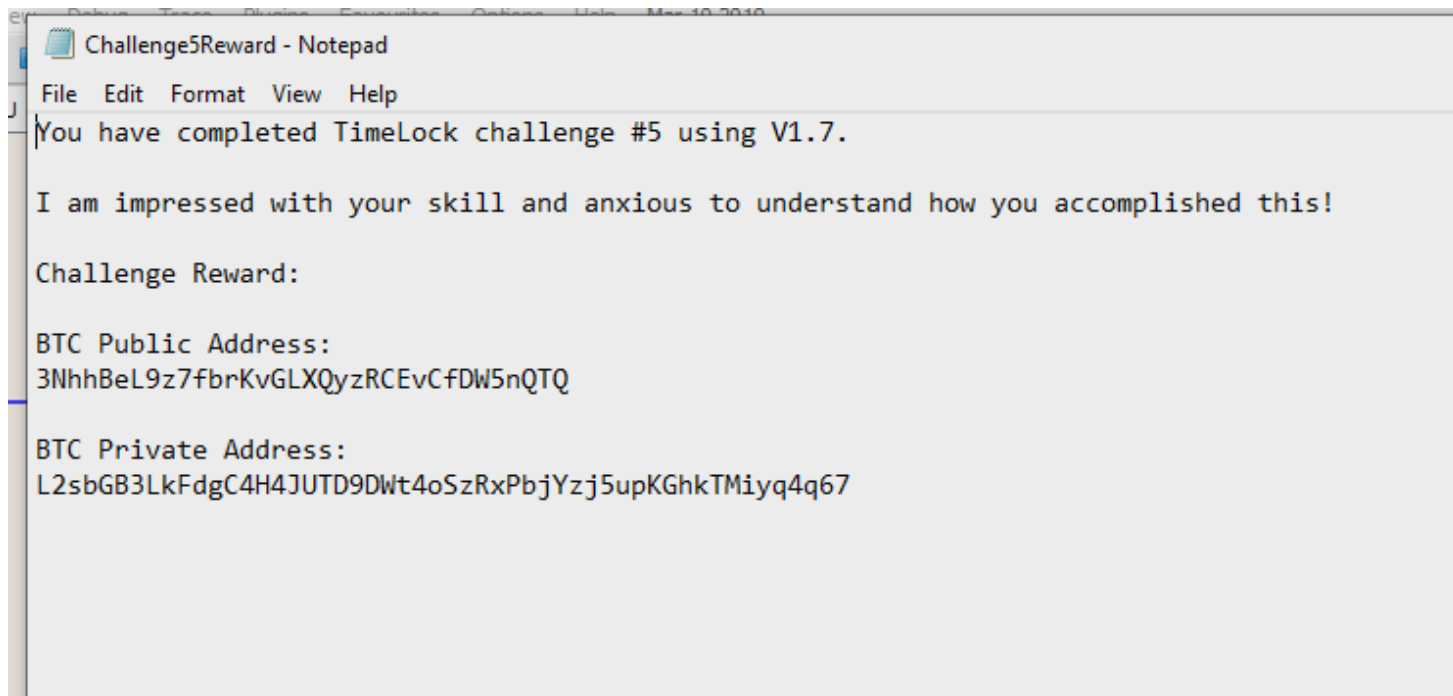
Address	String
000002377D0E0760	000002377D0E0760
000002377D0E0770	000002377D0E0770
000002377D0E0780	000002377D0E0780
000002377D0E0790	000002377D0E0790
000002377D0E07A0	000002377D0E07A0
000002377D0E07B0	000002377D0E07B0
000002377D0E07C0	000002377D0E07C0
000002377D0E07D0	000002377D0E07D0
000002377D0E07E0	000002377D0E07E0
000002377D0E07F0	000002377D0E07F0
000002377D0E0800	000002377D0E0800
000002377D0E0810	000002377D0E0810
000002377D0E0820	000002377D0E0820
000002377D0E0830	000002377D0E0830
000002377D0E0840	000002377D0E0840
000002377D0E0850	000002377D0E0850
000002377D0E0860	000002377D0E0860
000002377D0E0870	000002377D0E0870
000002377D0E0880	000002377D0E0880
000002377D0E0890	000002377D0E0890
000002377D0E08A0	000002377D0E08A0
000002377D0E08B0	000002377D0E08B0
000002377D0E08C0	000002377D0E08C0
000002377D0E08D0	000002377D0E08D0
000002377D0E08E0	000002377D0E08E0
000002377D0E08F0	000002377D0E08F0
000002377D0E0900	000002377D0E0900
000002377D0E0910	000002377D0E0910

Hex dump of the patch data, showing the first round of encryption. The data is organized into columns, with the first column containing the address and the subsequent columns containing the hex values.

Hex dump:

Address	Hex
000002377D0E0760	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E0770	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E0780	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E0790	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E07A0	43 68 61 6C 6C 65 6E 67 65 35 52 65 77 61 72 64
000002377D0E07B0	2E 74 78 74 00 59 6F 75 20 68 61 76 65 20 63 6F
000002377D0E07C0	6D 70 6C 65 74 65 64 20 54 69 6D 65 4C 6F 63 68
000002377D0E07D0	20 63 68 61 6C 6C 65 6E 67 65 20 23 35 20 75 73
000002377D0E07E0	69 6E 67 20 56 31 2E 37 2E 0D 0A 0D 0A 49 20 61
000002377D0E07F0	6D 20 69 6D 70 72 65 73 73 65 64 20 77 69 74 68
000002377D0E0800	20 79 6F 75 72 20 73 68 69 6C 6C 20 61 6E 64 20
000002377D0E0810	61 6E 78 69 6F 75 73 20 74 6F 20 75 6E 64 65 72
000002377D0E0820	73 74 61 6E 64 20 68 6F 77 20 79 6F 75 20 61 63
000002377D0E0830	63 6F 6D 70 6C 69 73 68 65 64 20 74 68 69 73 21
000002377D0E0840	0D 0A 0D 0A 43 68 61 6C 6C 65 6E 67 65 20 52 65
000002377D0E0850	77 61 72 64 3A 20 0D 0A 0D 0A 42 54 43 20 50 75
000002377D0E0860	62 6C 69 63 20 41 64 64 72 65 73 73 3A 20 0D 0A
000002377D0E0870	33 4E 68 68 42 65 4C 39 7A 37 66 62 72 48 76 47
000002377D0E0880	4C 58 51 79 7A 52 43 45 76 43 66 44 57 35 6E 51
000002377D0E0890	54 51 0D 0A 0D 0A 42 54 43 20 50 72 69 76 61 74
000002377D0E08A0	65 20 41 64 64 72 65 73 73 3A 20 0D 0A 4C 32 73
000002377D0E08B0	62 47 42 33 4C 68 46 64 67 43 34 48 34 4A 55 54
000002377D0E08C0	44 39 44 57 74 34 6F 53 7A 52 78 50 62 6A 59 7A
000002377D0E08D0	6A 35 75 70 4B 47 68 68 54 4D 69 79 71 34 71 36
000002377D0E08E0	3D 47 AB AB AB AB AB AB AB AB AB AB AB AB AB AB
000002377D0E08F0	AB AB EE FE EE FE EE FE EE FE EE FE EE FE EE FE
000002377D0E0900	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002377D0E0910	EE FE EE FE EE FE EE FE 73 C5 72 BD A7 59 10 30
000002377D0E0920	50 A8 91 C9 F8 7F 00 00 70 4A 15 7D 37 02 00 00
000002377D0E0930	70 4A 15 7D 37 02 00 00 10 4A 15 7D 37 02 00 00

# Loot #5



# Lessons Learned

Vulnerability found:

- Flawed encryption rounds may as well not be present at all.
- Crypto with hardcoded constants is easy to get around, since there are no variable secrets like time.
- Nesting crypto with different properties to make up for their shortcomings doesn't work when they are each critically flawed.

How to fix:

- Adopt a public – private key encryption scheme, and use a trusted third party to store keys and only provide access to authorised users.

# The Case For Hackers In Dev Teams

- Auditing at end of development, right before production is not productive.
  - Its irresponsible and costly.
  - Deep flaws unlikely to be fixed before production.
- Security needs to be present at every step:
  - Early design / architecture planning.
  - Development.
  - Testing.
  - Deployment.
  - Post production.
- Hackers are specifically trained to consider the “big picture” when it comes to complex system interactions and spot small nuisances which are easily overlooked.

# What Would a Robust TimeLock Look Like?

- Client and Server model.
- Secrets stored on trusted server.
- Data never leaves the client.
- Communication with public / private encryption systems.
  - Server.priv stored on the server.
  - Server.pub distributed to all clients.



# Encryption

- Client wishes to make Lockbox:
  - Client sends `Server.pub(hash(password, answers), start, stop)` to Server.
  - Server stores `hash(password, answers), start, stop`.
  - Server generates new `lockbox.priv` and `lockbox.pub`, sends `Server.priv(lockbox.pub)` to client.
  - Client makes `lockbox.pub(data) + enc(answers)`.

# Decryption

- Client wishes to decrypt:
  - Client collects and sends  
Server.pub(lockbox.pub(hash(password, answers))).
  - Server decrypts. Verifies hash and checks time window.
  - If correct, server sends  
Server.priv(lockbox.priv(lockbox.priv)).
  - Client decrypts with lockbox.priv(encdata).

# Greetz

- u/cryptocomicon, for the interesting challenges.
- #kiwicon users for the weekday banter.
  - I am captianype, btw.
- NSA for making Ghidra public and FOSS.
  - Ghidra is the biggest thing to happen in reverse engineering for quite some time...

# About Me

- Sustaining Engineer at Canonical.
  - Fixing Linux kernel gremlins in Ubuntu kernels.
  - Come find me and chat about Linux or reversing =)
- Read my blog:

<https://ruffell.nz>

[matthew@ruffell.nz](mailto:matthew@ruffell.nz)