rcrypt packer

by Feiad Mohammed

rcrypt features

- timelock puzzle for delayed execution
- anti virtualization (vbox)
- fake imports
- supports EXE and DLL files

rcrypt purpose

rcrypt is a crypter (a type of packer) and its purpose is similar to others of its kind. To protect intellectual property. If you're familiar with commercial software, music, movies, games, etc you are familiar with this type of protection. If you create some fancy new optimization algorithm and want to profit from it you do not want someone to simply reverse engineer your binary and make use of your hard work. When commercial entities use this kind of binary protection it is called DRM (Digital Rights Management). Rcrypt offers users the ability to encrypt their binaries to make reverse engineering harder. No solution is absolute and rcrypt is no exception. It is a proof of concept and nothing more. I make no guarantee that your binary will even function after packing so your mileage may vary. Just be aware that I do not condone malicious use of rcrypt as it is designed for educational purposes only.

rcrypt features

The timelock puzzle functionality causes delay in execution to avoid virtualized scanning techniques employed by Antivirus products. In rcrypt a timelock puzzle is implemented by the brute forcing of the crypto key used to encrypt the binary with varying difficulty specified by optional parameter -complex *hexvalue*.

The encryption algorithm used in rcrypt is XXTEA [1] also known as Corrected Block TEA. I am a fan of the Tiny Encryption Algorithm as it can be implemented entirely with CPU registers with no need for stack variables. It is a very efficient algorithm. Seeing as XTEA supersedes it, followed immediately by XXTEA which supersedes both I decided to use XXTEA. It is indeed the ultimate of TEAs.

Binaries with no or very few imports are highly suspicious. rcrypt will add fake imports to packed binaries. The imports I've chosen are based on the default imports created by a hello world Visual Studio program built in release mode.

Anti virtualization is currently limited to Virtualbox. I plan to add support for others as time permits. VMWare is far too ubiquitous to have an anti for as many systems actually run on it.

Additional features not explicitly mentioned include the following:

anti debugger functionality

obviously designed to detect debuggers to make reversing more difficult.

polymorphic self destructing stub

the unpacking stub kills itself after unpacking completes. To support DLLs there is a small portion that remains after the self destruct feature.

Anti-virus Scanning Methodologies and Malware

Antivirus products have been made a fool of for a very long time. Signature based scanning, no matter how well done, cannot deal with packed executables as they completely mask the byte patterns used to detect malware. This is why sandbox scanning methods are now the main method used by the best products out there (ie: Kaspersky comes to mind). Using these emulated environments to allow binaries to execute defeats the binary obfuscation offered by packers as all packers must unpack in memory. Virtualization has become ubiquitous enough to allow isolated environments to be created for every single EXE or DLL file that an antivirus scanner might want to scan.

However these full blown virtualized environments have overhead and cannot be run throughout the entire lifetime of the target binary executable. If a user had to wait for an executable to fully execute they might be waiting a long time. Especially since most binaries are graphical user interfaces or otherwise programs awaiting user input. Imagine this happening for countless binaries at any given time and having to wait for a double clicked program to start but only after it has been virtualized in its entirety to determine whether or not it was malicious. This is why these scanning techniques can only be employed for a short amount of time. Due to the nature of malicious binaries a few seconds is generally all they require to unpack and modify/exfiltrate/beacon which will trigger detection. This makes even a few seconds of virtualized based scanning worthwhile.

Pausing for a short amount of time is a very well known method of bypassing this type of scanning technique and has been in use for a long time. Just imagine a function like the following:

```
evilFunction
{
Sleep(60000);
AllTheBadThings();
}
```

Although using Sleep() is so common it is now ignored by almost every scanner out there the point is still there. If you can cause execution to delay for a while then you can bypass such scanning methods.

Ok so are there other ways to delay execution?

Timelock Puzzle

One of the more interesting ways to produce a delay in execution is to use cryptographic functionality to somewhat predictably cause execution delay. Various cryptographic computations have been proposed and do a great job in doing pointless work to cause delay.

Pointless Computations and Avoidability

A human analyst can detect useless computations if they can determine pointless functionality and remove it from the executable code as it is not a necessary part of the binary.

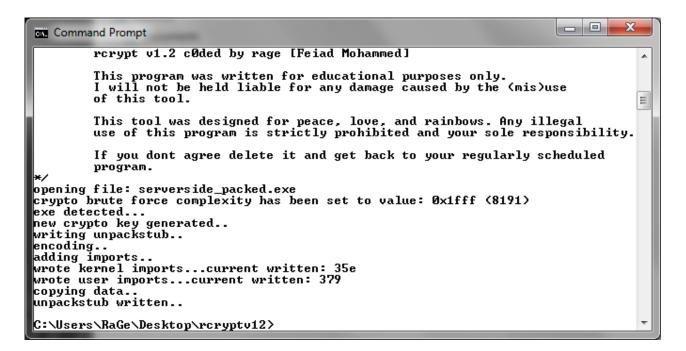
Timelock Puzzle in rcrypt

I liked the general idea behind timelock puzzles and thought that making use of cryptographic functionality while brute forcing a crypto key would be sufficient to cause delay. In rcrypt when a binary is encrypted a randomly generated key is used. Part of the packer's in memory routine is now to brute force this key which will facilitate delay and require the brute forcing as a necessary part of the process.

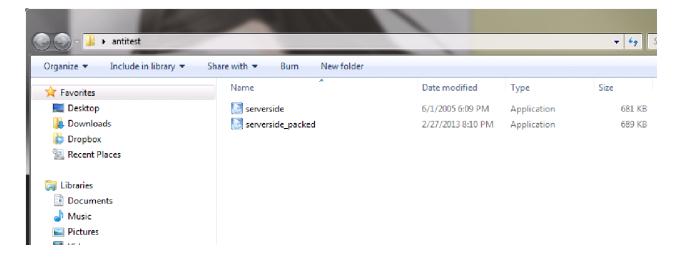
Example rcrypt packed malware detection

After scanning a known malware sample packed with rcrypt against a fully updated KAV I determined that this was sufficient to bypass a typical sandbox used by Antivirus scanning engines. Rcrypt has an optional parameter -complex 0x1234 users can use to tweak the difficulty of the brute forcing routine to increase/decrease delay.

For demonstration I will take a very well known malware released ten years ago that is detected by every Antivirus product and pack it.

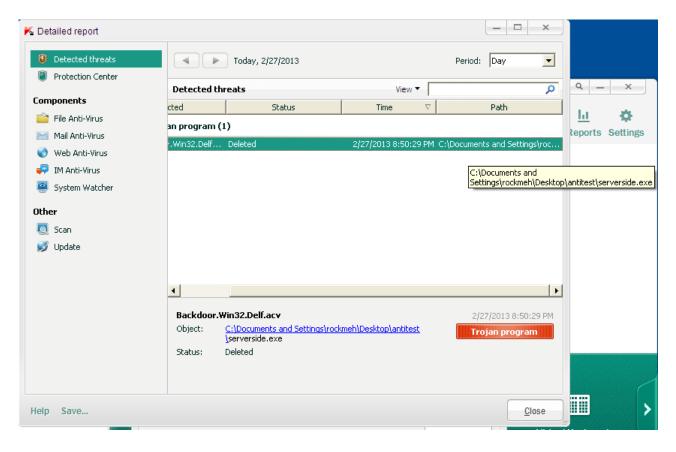


The malware sample is called Insurrection and has a client and server component. The server file is named serverside.exe and is copied to the rcrypt directory and packed.

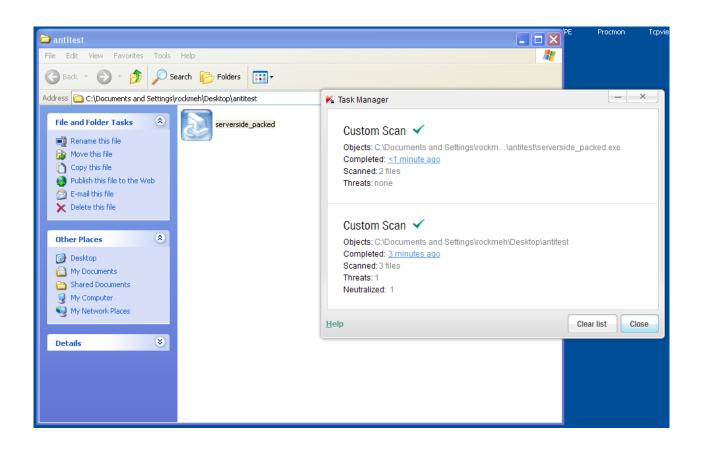


Below you can see the packed and unpacked binaries in a folder.

KAV is installed in a VM and updated and detects the unpacked binary immediately but does not detect the packed version.



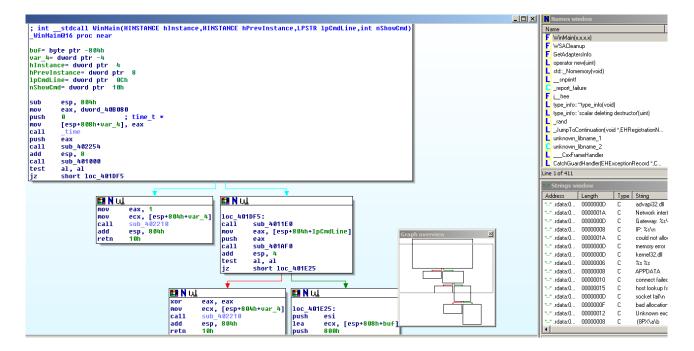
Unpacked detection.



This shows that rcrypt delays execution long enough to timeout KAV's sandbox scanning technology.

rcrypt Reversing in IDA Pro

Because the sample is packed IDA cannot determine original functionality or even identify functions or other known code. Below is an image of IDA Pro after loading the unpacked sample.



You can see IDA Pro has been able to produce a nice control flow graph. Now here is an image of IDA Pro after loading the packed version.

≝ View-A 🎦 Exports I	응 Imports N Names 가 Functions 는 Strings & Structures En Enums	
		■□ × Names window
40E02A	jmp 10C_40F30C	Name
40E02A ;		C start
40E02F	db 80h ; Ç	A aBaks?BI3mSB8a
40E030	dd 906142Ch, 83116000h, 74881161h, 846A2AB8h, 561D903Dh	A aTeminateproce
40E030	dd 0AF600000h, 7B16188h, 881F3062h, 5111608Eh, 15117F9Ch	A aRitefile
40E030	dd 0AB3CB163h, 0CB01184Bh, 759D916Ah, 1F173806h, 0C4600000h	
40E030	dd 10DE410Fh, 22000817h, 082782560h, 309848BCh, 96916AA0h	A aEtstdhandle
40E030	dd 8609B17Fh, OCA610EB1h, 85DE618Fh, 0B5A01F30h, 0B9780660h	A aFreeenvironmen
40E030	dd 36003700h, 088816600h, 0FD01984Bh, 3C048829Fh, 0FFFF6FDh	A aTlunwind
40E030 40E030	dd 080791180h, 08E61048Eh, 0A160848Fh, 002011854h, 0A30916Ah	A aGetcurrentthre
40E030	dd 2500201Dh, 5D08B160h, 0C6FF08B1h, 44001E88h, 0C70A0000h dd 61FFFFF1h, 0CAFF88B1h, 2C001E08h, 9C3D6000h, 357E08B1h	I CreateFileA
40E030	dd 4CB1203bh, 9E 09817Eh, 8 FBF8907h, 92608163h, 9FAFE80A8h	I VirtualProtect
40E030	dd 001406057h, 06088813Eh, 303E8881h, 0C7B12199h, 0F37ABD7Eh	I GetLocaleInfoA
40E030	dd 908817E88h, 9E99999848h, 9BA3D916Ah, 3C90881Dh, 6D988168h	I GetModuleHandleA
40E030	dd 0ED7D088B1h, 0DDB12188h, 3EFFFFE9h, 20E81089h, 41002000h	I GetStartupInfoA
40E030	dd 7AAF0000h, 0C04A806Fh, 8309B178h, 37E511B1h, 65B18000h	T GetCommandLineA
40E030	dd 0A7FFFFE5h, 987D08B1h, 0E8B12788h, 0C9FFFFE7h, 43E791B1h	GetVersionExA
40E030	dd 0D1308000h, 3E91B120h, 0D80000h, 31FFE593h, 8F985080h	T ExtProcess
4 0E 03 0	dd 41E712B1h, 71B18000h, 0B3FFFFE7h, 5D7D09B1h, 0DCB12C09h	
40E030	dd 4712B17Ch, 0A7800000h, 0F7B12C08h, 0D8FFFFE8h, 4CE711B1h	Line 1 of 58
40E030	dd 0FAB18000h, 0CDFFFFE8h, 52C8A20h, 0DEB8FA8Bh, 7000001Bh	Strings window
40E030	dd 18613D60h, 9C107B89h, 0D388B120h, 0D7C89B1h, 83277B8Bh	
40E030	dd 76F061E3h, 0F600000h, 6BB17C89h, 0D909007Dh, 0F17B8AB1h	Address Length
40E030	dd 87A07220h, 6888B13Dh, 0B1D8EC0h, 9EFFFFDBh, 2D7C08B1h	"" .rdata:0 00000005
40E030	dd 6BFFE992h, 5009A780h, 7B1970A2h, 0FB16000h, 0E408B17Dh	"" .rdata:0 00000005
40E030	dd 33248840h, 71FFE491h, 0D689B180h, 0A97D0AB1h, 97B1238Ah	"" .rdata:0 00000005
40E030	dd 0EAFFFFE9h, 397B88B1h, 5DFFE492h, 0A596E200h, 6E10B168h	"" .rdata:0 00000006
40E030	dd 1F800000h, 87807D0Ah, 0F111B132h, 0C07FFFFFh, 0FFFFE812h	"" .rdata:0 00000006
40E030	dd 820AA57Fh, 0F3E892A7h, 846E5FFFh, 0EEE8108Th, 21A57FFFh	"" .rdata:0 00000005
40E030	dd 0F988077Dh, 28B16DAEh, 49FFFFE8h, 0CAFB0981h, 0F6FFE816h	"" .rdata:0 00000007
40E030 40E030	dd 0F710818Fh, 2D1FFFFFh, 0F9FFC67Dh, 0FA10818Fh, 301FFFFFh	"" .rdata:0.,. 00000005
4 0E 03 0 4 0E 03 0	dd 0F8FFBCFDh, 2979117Fh, 0DE1A1178h, 613B1A5h, 87800000h dd 6855DE5Fh, 0D73549F8h, 0FFFFE697h, 1796BFFFh, 3A200000h	"" .rdata:0 00000005
40E030 40E030	dd 0055025711, 0073549761, 07776105711, 179087711, 37200001 dd 4DFFFGSh, 0068608814h, 87FEA19h, 21BEFB20h, 008EBF22h	
40E030	dd 4910 BEC6h, 69FFEB7bh, 2919B144h, 9E3808080h, 15D3D20h	
40E030	dd 456186060h, 9FFFFE8h, 089486C81h, 089548061h, 7810903Dh	"".rdata:0 0000005
40E030	dd 90B608080h, 9CBB12188h, 9EFFETSh, 9ED2188B1h, 12FFE199h	"" .rdata:0 00000006
40E030	dd 2192B180h, 6F0408060h, 12FFE092h, 2498B180h, 6F380000h	"" .rdata:0 00000005
40E030	dd 1CFFE391h, 75890886h, 29E41B1h, 57818800h, 0AAFFFFE4h	"" .rdata:0 00000005
	a	▼1 "" .rdata:0 00000005

As you can see you get quite a bit of garbage. Also note that the pretty control flow graphs newbie reversers tend to be over dependent upon are no longer produced by IDA. Static analysis is useless on packed binaries. Luckily we can use the built in debugger and step through code and IDA will update the view based on new information gleaned.

I did just that and produced the following results.

rage:0940F030 dd 0C383C300h, 0C6D82B2Ch, 8940E900h, 48BC318h, 0F7E8C324h, 0C3FFFFFh rage:0940F30C;		rade:0040E030
rage:0040F30C rage:0040F30C loc_40F30C: ; CODE XREF: rage:0040E02A [†] j rage:0040F31C du 7504h rage:0040F311 add ds:4740000h, al rage:0040F311 i,	; CODE XREF: <mark>rage</mark> :0040E02A [†] j	
rage:8040F30C loc_40F30C: ; CODE XREF: rage:8040E02A [†] j rage:0040F311 add ds:47400000h, al rage:0040F311 add ds:47400000h, al rage:0040F311 dw 7504h rage:0040F318 db 2 rage:0040F318 db 2 rage:0040F310 db 0C2h, 0C3h, 0E8h rage:0040F310 db 0CDh, 2 dup(0FFh) rage:0040F320 db 89h; ë rage:0040F321 db 85h, 70h, 0FDh rage:0040F328 db 0FFh rage:0040F328 db 0FFh rage:0040F338 db 0FFh rage:0040F331 db 2 dup(0FFh), 89h rage:0040F331 db 2 dup(0FFh), 89h rage:0040F338 db 85h; à rage:0040F338 db 0FFh rage:0040F338 db 0FFh	; CODE XREF: rage:0040E02A [↑] j	
Tage:0040F30C push ebp rage:0040F311 add ds:4740000h, al rage:0040F311 ;	; CODE XREF: rage:0040E02ATj	
<pre>rage:0040F311 add ds:4740000h, al rage:0040F311 ; rage:0040F311 dv 7504h rage:0040F318 db 2 rage:0040F319 db 0C2h, 0C3h, 0E8h rage:0040F310 db 0Ehh, 2 dup(0FFh) rage:0040F320 db 89h ; ë rage:0040F321 db 85h, 70h, 0FDh rage:0040F324 db 0FFh rage:0040F325 db 0FFh, 6Ah, 2 rage:0040F328 db 0FFh rage:0040F328 db 0FFh rage:0040F329 db 0B5h, 70h, 0FDh rage:0040F320 db 0FFh, 0E8h, 9 rage:0040F320 db 0FFh, 0E8h, 9 rage:0040F331 db 2 dup(0FFh), 89h rage:0040F334 db 85h ; à rage:0040F334 db 85h ; à rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh</pre>		
<pre>rage:0040F311 ;</pre>		
<pre>rage: 80440F316 dw 7504h rage: 80440F318 db 2 rage: 80440F319 db 0C2h, 0C3h, 0E8h rage: 80440F310 db 0Fh rage: 80440F310 db 0Fh, 2 dup(0FFh) rage: 80440F321 db 85h, 2 dup(0FFh) rage: 80440F324 db 0FFh rage: 80440F325 db 0FFh, 6Ah, 2 rage: 80440F325 db 0FFh, 6Ah, 2 rage: 80440F326 db 0FFh rage: 80440F326 db 0FFh rage: 80440F327 db 0FFh rage: 80440F320 db 0FFh, 9 rage: 80440F330 db 0FFh, 9 rage: 80440F331 db 2 dup(0FFh), 89h rage: 80440F331 db 2 dup(0FFh) rage: 80440F338 db 0FFh rage: 80440F339 db 0E8h, 0ABh, 0FFh rage: 80440F336 db 8FFh rage: 8048F346FFh rage: 80440F336 db 8FFh rage: 8048F346FFh</pre>		
rage:0040F318 db 2 rage:0040F319 db 0C2h, 0C3h, 0E8h rage:0040F31C db 0Fh rage:0040F320 db 89h; ë rage:0040F320 db 89h; ë rage:0040F321 db 85h, 70h, 0FDh rage:0040F325 db 0FFh rage:0040F325 db 0FFh, 6Ah, 2 rage:0040F325 db 0FFh rage:0040F32C db 0FFh rage:0040F32C db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F336 db 0FFh		
rage:0040F319 db 0C2h, 0C3h, 0E8h rage:0040F31C db 0Fh rage:0040F31D db 0EDh, 2 dup(0FFh) rage:0040F320 db 89h ; ë rage:0040F321 db 85h, 70h, 0FDh rage:0040F324 db 0FFh rage:0040F325 db 0FFh, 6Ah, 2 rage:0040F328 db 0FFh rage:0040F320 db 085h, 70h, 0FDh rage:0040F320 db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh ; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F334 db 85h ; à rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh		
rage:0040F31C db 0Fh rage:0040F31D db 0EDh, 2 dup(0FFh) rage:0040F320 db 89h ; ë rage:0040F321 db 85h, 70h, 0FDh rage:0040F325 db 0FFh rage:0040F325 db 0FFh rage:0040F329 db 085h, 70h, 0FDh rage:0040F320 db 0FFh, 0E8h, 9 rage:0040F320 db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh ; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh		
rage:0040F31D db 0EDh, 2 dup(0FFh) rage:0040F320 db 89h ; ë rage:0040F321 db 85h, 70h, 0FDh rage:0040F324 db 0FFh rage:0040F325 db 0FFh, 6Ah, 2 rage:0040F329 db 0FFh rage:0040F320 db 0FFh rage:0040F32D db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh ; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh		
<pre>rage:0040F320 db 89h ; ë rage:0040F321 db 85h, 70h, 0FDh rage:0040F324 db 0FFh rage:0040F325 db 0FFh, 6Ah, 2 rage:0040F328 db 0FFh rage:0040F329 db 085h, 70h, 0FDh rage:0040F320 db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh ; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh</pre>		
rage:0040F321 db 85h, 70h, 0FDh rage:0040F324 db 0FFh rage:0040F325 db 0FFh, 6Ah, 2 rage:0040F328 db 0FFh rage:0040F320 db 085h, 70h, 0FDh rage:0040F32C db 0FFh rage:0040F330 db 0EDh; f rage:0040F330 db 0EDh; f rage:0040F334 db 2 dup(0FFh), 89h rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F339 db 0E8h, 0ABh, 0FFh rage:0040F33C db 0FFh		
rage:0040F324 db 0FFh rage:0040F325 db 0FFh, 6Ah, 2 rage:0040F328 db 0FFh rage:0040F329 db 085h, 70h, 0FDh rage:0040F32C db 0FFh rage:0040F330 db 0EDh; f rage:0040F330 db 0EDh; f rage:0040F334 db 2 dup(0FFh), 89h rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F339 db 0E8h, 0ABh, 0FFh		
rage:0040F325 db 0FFh, 6Ah, 2 rage:0040F328 db 0FFh rage:0040F329 db 0B5h, 70h, 0FDh rage:0040F32C db 0FFh rage:0040F32D db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh ; f rage:0040F334 db 2 dup(0FFh), 89h rage:0040F334 db 25h ; à rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh		
rage:0040F328 db 0FFh rage:0040F329 db 0B5h, 70h, 0FDh rage:0040F32C db 0FFh rage:0040F32D db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh ; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh		
rage:0040F32C db 0FFh rage:0040F32D db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh ; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F334 db 85h ; à rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F338 db 0FFh		
rage:0040F32D db 0FFh, 0E8h, 9 rage:0040F330 db 0EDh ; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F334 db 85h ; à rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F339 db 0E8h, 0ABh, 0FFh rage:0040F33C db 0FFh		rage:0040F329
<pre>* rage:0040F330 db 0EDh ; f rage:0040F331 db 2 dup(0FFh), 89h rage:0040F334 db 85h ; à rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F339 db 0E8h, 0ABh, 0FFh rage:0040F33C db 0FFh</pre>		rage:0040F32C
rage:0040F331 db 2 dup(0FFh), 89h rage:0040F334 db 85h ; à rage:0040F335 db 18h, 0FDh, 0FFh rage:0040F338 db 0FFh rage:0040F339 db 0E8h, 0ABh, 0FFh rage:0040F33C db 0FFh		rage:0040F32D
* rage:0040F334 db 85h ; à * rage:0040F335 db 18h, 0FDh, 0FFh * rage:0040F338 db 0FFh * rage:0040F339 db 0E8h, 0ABh, 0FFh * rage:0040F33C db 0FFh		
* rage:0040F335 db 18h, 0FDh, 0FFh * rage:0040F338 db 0FFh * rage:0040F339 db 0E8h, 0ABh, 0FFh * rage:0040F33C db 0FFh		
* rage:0040F338 db 0FFh * rage:0040F339 db 0E8h, 0ABh, 0FFh * rage:0040F33C db 0FFh		
rage:0040F339 db 0E8h, 0ABh, 0FFh rage:0040F33C db 0FFh		
* rage:0040F33C db 0FFh		
rage:0040F33V align 2		
* rage:0040F33E dw 0E850h		rage:0040F33D

Here I am continuing to step through code and there is still no graph or easily identifiable code constructs.

And then finally....

<pre>rage:0840F344 db 80h; i rage:0840F345 db 90h, 18h, 0FDh rage:0840F348 db 9FFh rage:0840F348 db 9FFh rage:0840F348 db 9FFh rage:0840F348 db 0FFh rage:0840F348 db 0FFh rage:0840F348 db 0FFh rage:0840F348 db 0FFh rage:0840F348 db 0FFh rage:0840F351 db 0FFh rage:0840F352 db 0FFh</pre>
* rage: 0040F348 db 0FFh * rage: 0040F340 db 0FFh * rage: 0040F340 ebx * rage: 0040F340 b0 FFh * rage: 0040F340 b0 6Fh * rage: 0040F340 b0 6Fh * rage: 0040F340 b0 6Fh * rage: 0040F351 b0 6Fh * rage: 0040F355 b0 FFh * rage: 0040F355 b1 6Fh * rage: 0040F356 b1 6Fh * rage: 0040F365 b1 6Fh * rage: 0040F364 D0 7Th
<pre>* rage: 0040F340 db 0FFh rage: 0040F34A ; rage: 0040F34A ; rage: 0040F34A ; rage: 0040F34A db 0ESh ; rage: 0040F34A db 0ESh ; rage: 0040F34A db 0FFh rage: 0040F34C db 0FFh rage: 0040F35A db rage: 0040F3A db rage: 0040</pre>
rage:0040F34A ; rage:0040F34A push else else rage:0040F34A push else else rage:0040F34A push else else rage:0040F34A push else else rage:0040F34B db push else rage:0040F34B db push else rage:0040F35A db of 0040F35A db rage:0040F35A db rage:0040F36A db rage:
<pre>* rage: 004/0F34A push ebx rage: 004/0F34A db 0E8h ; F rage: 004/0F34A db 0E8h ; F rage: 004/0F34A db 0B9h rage: 004/0F35A db 0FDh, 2 dup(0FFh) rage: 004/0F35A db 0FDh rage: 004/0F35A db 0FFh rage: 004/0F3A db rage: 0</pre>
rage:0040F34A • rage:0040F34A 00 0FBh; rage:0040F34A 00 0FBh; rage:0040F34A 00 0FFh) rage:0040F35A 00 0FFh) rage:0040F35A 00 0FFh) rage:0040F35A 00 0FFh) rage:0040F35A 00 0FFh
* rage: 084.0F 34.0 db 0F.Bh ; F * rage: 084.0F 34.0 db 0F.Dh ; rage: 084.0F 35.0 db 0F.Fh * rage: 084.0F 35.5 db 0F.Fh ; rage: 084.0F 35.5 db 0F.Fh * rage: 084.0F 35.5 db 0F.Fh ; rage: 084.0F 35.5 db 0F.Fh * rage: 084.0F 35.5 db 0F.Fh ; rage: 084.0F 35.5 db 0F.Fh * rage: 084.0F 35.5 db 0F.Fh ; rage: 084.0F 35.5 db 0F.Fh * rage: 084.0F 35.5 db 0F.Fh ; rage: 084.0F 35.5 db 0F.Fh * rage: 084.0F 35.5 db 0F.Fh ; rage: 084.0F 35.5 db malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) * rage: 084.0F 35.6 db image: malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) * rage: 084.0F 35.6 db image: 084.0F 36.6 db * rage: 084.0F 36.6 db image: 084.0F 36.6 db * rage: 084.0F 36.6 db image: 084.0F 36.6 db
* rage: 0040F340c 0b 0FFh * rage: 0040F3340c 0b 0FFh * rage: 0040F3530c 0b 0FFh * rage: 0040F3550c 0b 0FFh * rage: 0040F350c 0b 0b * rage: 0040F350c 0b 0bb * rage: 0040F350c 0b 0bb * rage: 0040F350c 0b 0bb * rage: 0040F363c 0b 0bb * rage: 0040F363c 0b 0bb * rage: 0040F363c 0b
<pre>rage:0040F330 db 0FDh, 2 dup(0FFh) rage:0040F330 db 0FDh, 2 dup(0FFh) rage:0040F335 db 0FFh rage:0040F355 db 0FFh rage:0040F355</pre>
r age: 0040F355 db 0FFh rage: 0040F355 db 09Fh rage: 0040F355 db 09Fh rage: 0040F355 db 09Fh rage: 0040F355 dt rage: 0040F355 dt rage: 0040F355 dt rage: 0040F355 dt rage: 0040F355 dt rage: 0040F356 dt rage: 0040F366 dt rage: 0040
* rage: 004/0F351 db 005D, 180, 0F5D * rage: 004/0F354 db 0FFL * rage: 004/0F355 db 0FFL * rage: 004/0F355 db 0FFL * rage: 004/0F354 db 0FFL * rage: 004/0F354 db 0FFL * rage: 004/0F355 db 0FE * rage: 004/0F355 db 0FE * rage: 004/0F356 db 0FE * rage: 004/0F356 db 0FE * rage: 004/0F364 db 0FE * rage: 004/0F364 db 0K * rage: 004/0F364 db 0K * rage: 004/0F364 db 0K
<pre>rage:0040F354 db 0FFh rage:0040F355 db 0FFh rage:0040F355 db 0FFh rage:0040F355 db affFh affFh</pre>
* rage: 0040F355 dt 0FBF 52h 52h 0FBF 52h <
rage: 804.0F358 Umaining X rage: 804.0F359 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804.0F351 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804.0F351 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804.0F351 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804.0F351 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804.0F351 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804.0F354 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804.0F364 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. rage: 804.0F364 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC. rage: 804.0F364 malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFEFE3EC.
rage: 804.0F355 dl malware_sample_packed.exe: The instruction at 0x40F077 referenced memory at 0xFEFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804.0F355 dl rage: 804.0F366 dl rage: 804.0F366 dl rage: 804.0F364 dl rage:
rage: 804/0F35C dl A malware_sample_packed exe: The instruction at 0x40F077 referenced memory at 0xFFFE3EC. The memory could not be read (0x0040F077 > FFEFE3EC) rage: 804/0F361 dl rage
* rage: 8040F350 dl / Marate_sande_pasted exe the instruction activity of resterict memory data for CFLCCCC. The memory data for CFLCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
rage:0940F360 dl DK rage:0940F361 dl Don't display this message again rage:0940F364 dl rage:0940F365 db 9Eh, 0FDh, 0FFh
* rage: 80440F361 dl DDnt display this message again * rage: 80440F364 dl * rage: 80440F365 dD 9Eh, 0FDh, 0FFh
* rage:0040F364 dl * rage:0040F365 db 9Eh, 0FDh, 0FFh
° rage:0040F365 db 9Eh, 0FDh, 0FFh
1 Hages 891 85249 db 8556
raye.0040roud up orri
* rage:0040F369 db 74h, 8, 75h
* rage:0040F36C db 6
* rage:0040F36D db 0A2h, 0C3h, 0E8h
* rage:0040F370 db 34h ; 4
* rage:0040F371 db 0E9h, 44h, 68h
* rage:0040F374 db 1Fh
* rage:0040F375 db 08Ah, 25h, 99h
° rage:0040F378 db 68h ; h
* rage:0040F379 db 1Fh, 0BAh, 25h
* rage:0040F37C db 99h ; Ö
° rage:0040F37D db 0FFh, 0B5h, 18h

IDA Pro crashes as it is unable to continue execution. As you can see this is what IDA Pro looks like when stepping through rcrypt packed executable code. There are many things here designed to trip up analysis and frustrate reverse engineering.

Detecting Packed Malware

To detect packed malware Antivirus products employ unpacking functionality for each supported packer to allow scanning of the original binary. One would simply need to be created for rcrypt. Of course this kind of approach does not scale as there are many packers out there.

Full Blown Virtualization

Automated analysis tools are a dime a dozen these days and will allow a user to upload binaries and get reports back from their virtualized analysis engines. These virtual environments run executables for arbitrary amounts of time so the delay in execution does not deter such analysis. Starting with rcrypt 1.3 there is an optional parameter "*-trick1*" that will attempt to detect the most typical sandbox environments and cause early termination. This is an option that shows that overreliance on simple automated solutions can be very misleading. Never rely solely on your tools as any tool can be fooled if someone wants to do so.

Conclusion

It is clear that the arms race between detecting malware and evading detection systems has produced some interesting new products and techniques. However no single solution solves everything and a layered security model is needed. There are always ways around detection techniques and evading techniques and all one can do is raise the bar as high as possible to lower the chance of falling prey to a successful attack.

My website: http://www.0xrage.com

References

[1] http://en.wikipedia.org/wiki/XXTEA