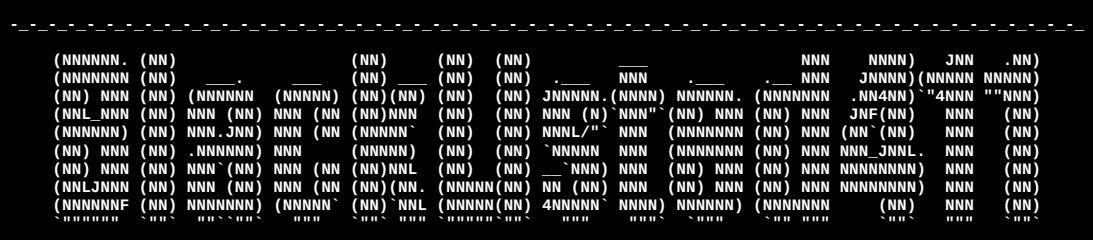


BLACKLISTED 411 .NET

Edition 6

03.19.2006

Table of Contents

[1] .NET Update

To fix the problem of .NET scheduled releases landing on inconvenient/busy days, the scheduled release date has changed to be the first Sunday/Monday after the 15th. Articles may be submitted 12 hours prior the release date.

[2] ==Edition Quote==

"Security is a process, not an end state."
Dr. Mitch Kabay – 1998

[2] ==News==

[a] Xbox 360 Copy Protection Broken

Recently, a team of six hackers (not including contributions from others in the Xbox::hacking community) released findings proving that the Xbox 360 was indeed NOT hack proof. Ironically, only four months before, Chris Satchell from Microsoft made the following statement.

> *"There are going to be levels of security in this box that the hacker community has never seen before"*

While the actual team responsible for the hack refuses to release a working hacked firmware, technical details provided give us a clear picture of Microsoft's flaw. To put it quite clearly, all Xbox media is signed by Microsoft (code signing) to prevent unauthorized code from executing. In order to prevent a 1:1 copy of the disc from playing, a mediaflag is included with the XEX (Xbox executable) on the disc (Which defines which media [cd-r, dvd-r, dvd-rw, dvdxbox360] is supposed to house the executable). Since modification of the actual mediaflag would invalidate the signature, a firmware hack was used to force the dvd drive to report the media as being "dvdxbox360" instead of simply a dvd-r, dvd-rw etc. While this hack provides users with the ability to boot from bootleg or backup copies, the final task at executing unsigned code is still at large.
http://www.xbox-scene.com/xbox1data/sep/EEuklElZyFpcibJZvz.php
http://www.xbox-scene.com/xbox1data/sep/EEukZpklFAhkcWwSgZ.php
http://www.teamxecuter.com/index.php?name=News&file=article&sid=215&CMSSESSID=2eee66ee6ed5de4475166d61dcdd4155

[b] Government receives D+ for computer security

While we are constantly being told of the clear and present danger of terrorist attacks at airports and other areas, and how the government has gone to great lengths to protect us, government agencies have neglected to properly secure their own computer infrastructure. Under the Federal Information Security Management Act (FISMA) of 2002, government agencies (twenty-four in all) receive grades based on their overall computer security. While many might overlook the lack of computer security as something that has no effect on them, let's not forget two important things. First off, in 2008, Real ID is set to debut which uses a centralized database to "securely" identify citizens. If the DOD and other "security/intelligence" related agencies are unable to secure themselves, how will they secure servers containing sensitive information pertaining to citizens? The second point is best described by the following paragraph from securityfocus.com:

> *"Sources interviewed for this article maintained that there have been SCADA (*supervisory control and data acquisition) *system attacks, but such incidents are almost never made public. Perhaps the most well-known public incident is that of an information-technology contractor who used his knowledge of control systems to release a million liters of sewage into a river basin in Australia. And U.S. authorities investigated online reconnaissance of U.S. critical infrastructure systems by attackers thought to be linked to al Qaeda in Pakistan, Saudia Arabia and Indonesia. "*

This classic example provided by security focus gives us a clear picture that compromised government security directly affects our lives. If we can rely on the government to secure their own infrastructure, why should we accept every new "security" measure they suggest? Security is only as strong as the weakest link and until the U.S. government realizes this and fixes its own security holes, no fancy million dollar system will ever provide us with true security. Clearly a D+ for government security is not acceptable.

http://www.securityfocus.com/news/11351
http://reform.house.gov/GovReform/News/DocumentSingle.aspx?DocumentID=40762
http://www.securityfocus.com/brief/167

[4] ==Articles==

[a] Hacking Cryptograms 101: Double your pleasure

Hacking Cryptograms 101
Double Your Pleasure

By Israel Torres <israel@israeltorres.org>

Welcome to the **fifth** lesson in hacking cryptograms. *If you aren't familiar with this column I highly suggest reading the first through the fourth lessons as there is cumulative knowledge being touched throughout.* In this .NET column we play with doubling your pleasure by doubling your crypto. In normal circumstances encrypting your encrypted data results in size increase due to data padding, but in this example we are using text rotation, namely ROT13 and ROT47 in tandem. (This is a basic 1:1 substitution)

In the example below you can see the benefit of alternating ROT13:ROT47 functions:

Macro	Seq.	Doubling with ROT(x)	Text Result
	0	Plaintext	Hello World
1	1	ROT13	**Uryyb Jbeyq**
	2	ROT47	**&CJJ3 y36JB**
2	3	ROT13	&PWW3 l36WO
	4	ROT47	U!((b =be(~
3	5	ROT13	H!((o =or(~
	6	ROT47	wPWW@ l@CWO
4	7	ROT13	jCJJ@ y@PJB
	8	ROT47	;ryyo Jo!yq
5	9	ROT13	;ellb Wb!ld

	10	ROT47	j6==3 (3P=5
6	11	ROT13	w6==3 (3C=5
	12	ROT47	Hellb Wbrld
7	13	ROT13	Uryyo Joeyq
	14	ROT47	&CJJ@ y@6JB
8	15	ROT13	&PWW@ l@6WO
	16	ROT47	U!((o =oe(~
9	17	ROT13	H!((b =br(~
	18	ROT47	wPWW3 l3CWO
10	19	ROT13	jCJJ3 y3PJB
	20	ROT47	;ryyb Jb!yq
11	21	ROT13	;ello Wo!ld
	22	ROT47	j6==@ (@P=5
12	23	ROT13	w6==@ (@C=5
	24	ROT47	Hello World

You can also see that the plaintext is eventually returned in this same alternation (*at sequence 24*). If you look carefully [highlighted in green] you can also see points in where the plaintext could be made out enough to give its sequence away. You certainly wouldn't want to use this for any type of secure environment, communications… yet many do – I recently saw it in a posting for secure web/database connections (**yikes**!).

The benefit of dual encoding your data is that someone may not instantly recognize that it is dual encoded and may give up trying to figure out a message or instruction string (*unless of course you are like me and just have to know what is going on just because [wink]*), also if there is a monitoring program searching for key words in your messages it may be prepared to recognize one or the other forms of rotation, but maybe not both in tandem (*though don't always count on it*).

In case you didn't already know one of my side projects is to open FTard Decoder Ring and make its full source available – when I get a chance I spend some time untangling things from my personal libraries and this is the perfect opportunity to show you how you can programmatically achieve dual encoding. Using macros as demonstrated above makes it a lot simpler when there is a lot of data to play with. In the following implementation of ROT13:ROT47 I created the function **frotate_c_ex();** to double your pleasure in 12 steps. Using my original project I wrote for ***Hacking Cryptograms 101****: "*Rotating letters is not secure, just annoying*"*, I have added more functionality for you to play with (I've added // [comment marks] to prior functionality that you may want to enable):

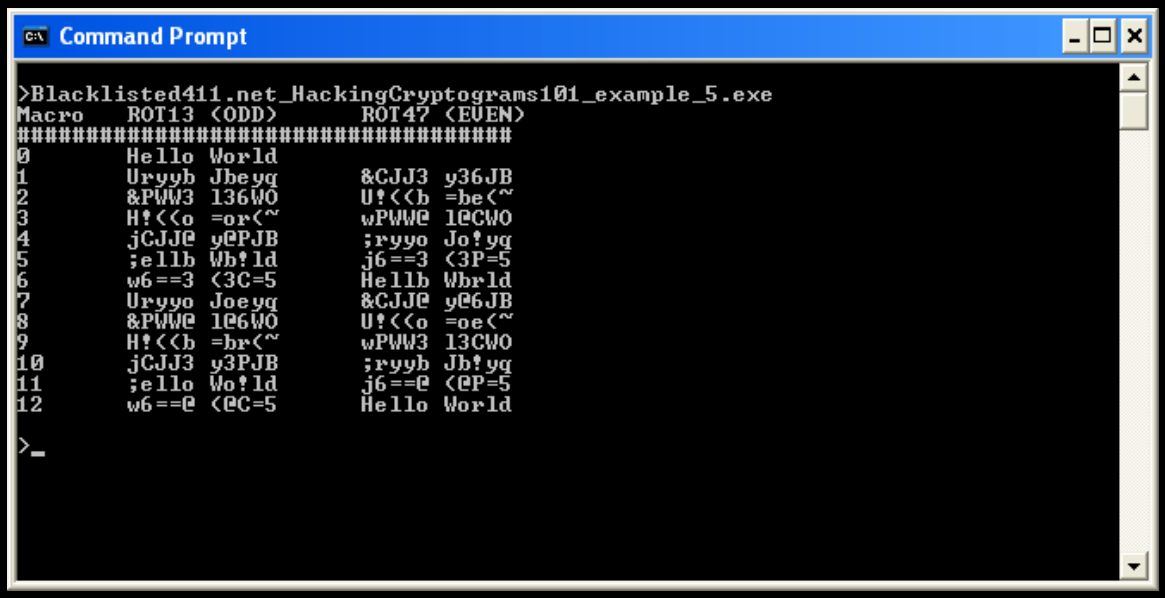
```
// Blacklisted411.net_HackingCryptograms101_example_5.cpp
// Hacking Cryptograms 101: Double Your Pleasure
// By Israel Torres < israel@israeltorres.org >
// Source: http://blacklisted411.israeltorres.org/

// This example demonstrates how to brute force ROT(x), ROT13, ROT47.
// You can play with the functions set in main:
// frotate_a_ex(), frotate_b_ex(), frotate_c_ex().
//

#include <iostream.h>
#include <string.h>

// here we declaring our rotation function prototype
void frotate_a(char szCrypto[255], int nrotate, char szCryptoOut[255]);
void frotate_a_ex(char szCrypto[255]);

void frotate_b(char szCrypto[255], int nrotate, char szCryptoOut[255]);
void frotate_b_ex(char szCrypto[255]);

void frotate_c_ex(char szCrypto[255]);
```

```
// here we are implementing our rotation function
void frotate_a(char szCrypto[255], int nrotate, char szCryptoOut[255])
{
	// here we are setting up our variables for use
	char szTemp[2]="\0";
	char szTemp2[2]="\0";
	int flag_skip = 0;

	// clear variable
	strcpy(szCryptoOut,"");

	// here we are finding out what the length of the cryptogram is
	int nCrypto = strlen(szCrypto);

	// here we are working for the entire length of the cryptogram
	for (int nloop = 0 ; nloop < nCrypto ; nloop++)
	{
		// here we are copying data into a variable for exclusive usage.
		szTemp[0] = szCrypto[nloop];

		// here we are comparing the value of the character to a set of ASCII conditions
		if ((int(szTemp[0]) >= 65) && (int(szTemp[0]) <= 90))
		{
			// we know that upper case letters are 26 values between 65 and 90
			// we then rotate the values accordinginly to make sure we are not exceeding the letter limit
			// and finally put the new data to the console
			//cout << char(65 + (int(szTemp[0]) - 65 - nrotate + 260)%26);
			szTemp2[0] = char(65 + (int(szTemp[0]) - 65 - nrotate + 260)%26);
			strcat(szCryptoOut, szTemp2);
			flag_skip = 1;
		}

		if ((int(szTemp[0]) >= 97) && (int(szTemp[0]) <= 122))
		{
			// we know that lower case letters are 26 values between 97 and 122
			// we then rotate the values accordinginly to make sure we are not exceeding the letter limit
			// and finally put the new data to the console
			//cout << char(97 + (int(szTemp[0]) - 97 - nrotate + 260)%26);
			szTemp2[0] = char(97 + (int(szTemp[0]) - 97 - nrotate + 260)%26);
			strcat(szCryptoOut, szTemp2);
			flag_skip = 1;
		}

		// here we are just passing anything that is not a letter directly to the console
		if (flag_skip == 0)
		{
			//cout << szTemp[0];
			strcat(szCryptoOut, szTemp);
		}

		// here we are resetting the flag for the next time around
		flag_skip = 0;
	}
}

void frotate_a_ex(char szCrypto[255])
{
	// here we are setting up our variables for use
	int max_rotate = 26 +1;
	char szCryptoOut[255]="\0";

	// here we are looping through our rotations
	for (int nrotate = 0; nrotate < max_rotate; nrotate++)
	{
		// here we are calling our rotation function
		// and presenting it to the console in this format
		frotate_a(szCrypto, nrotate, szCryptoOut);
		cout << nrotate << "\t" << szCryptoOut << endl;

	}
}

// here we are implementing our rotation function
void frotate_b(char szCrypto[255], int nrotate, char szCryptoOut[255])
{
```

```
	// here we are setting up our variables for use
	char szTemp[2]="\0";
	int flag_skip = 0;
	char szTemp2[2]="\0";

	// clear variable
	strcpy(szCryptoOut,"");

	// here we are finding out what the length of the cryptogram is
	int nCrypto = strlen(szCrypto);

	// here we are working for the entire length of the cryptogram
	for (int nloop = 0 ; nloop < nCrypto ; nloop++)
	{
		// here we are copying data into a variable for exclusive usage.
		szTemp[0] = szCrypto[nloop];

		// here we are comparing the value of the character to a set of ASCII conditions
		if ((int(szTemp[0]) >= 33) && (int(szTemp[0]) <= 126))
		{
			// we know that upper case letters are 26 values between 65 and 90
			// we then rotate the values accordinginly to make sure we are not exceeding the letter limit
			// and finally put the new data to the console
			//cout << char(33 + (int(szTemp[0]) - (33+47) - nrotate + 940)%94);
			szTemp2[0] = char(33 + (int(szTemp[0]) - (33+47) - nrotate + 940)%94);
			strcat(szCryptoOut, szTemp2);
			flag_skip = 1;
		}

		// here we are just passing anything that is not a letter directly to the console
		if (flag_skip == 0)
		{
			cout << szTemp[0];
			strcat(szCryptoOut, szTemp);
		}

		// here we are resetting the flag for the next time around
		flag_skip = 0;
	}
}

void frotate_b_ex(char szCrypto[255])
{
	// here we are setting up our variables for use
	int max_rotate = 47 +1;
	char szCryptoOut[255]="\0";

	// here we are looping through our rotations
	for (int nrotate = 0; nrotate < max_rotate; nrotate++)
	{
		// here we are calling our rotation function
		// and presenting it to the console in this format
		frotate_b(szCrypto, nrotate, szCryptoOut);
		cout << nrotate << "\t" << szCryptoOut << endl;
	}
}

void frotate_c_ex(char szCrypto[255])
{
	int nrotate_a = 13;
	int nrotate_b = 0;
	char szCryptoOut[255]="\0";

	cout << "Macro" << "\t" << "ROT13 (ODD)" << "\t" << " ROT47 (EVEN)" << endl;
	cout << "###################################" << endl;

	cout << "0" << "\t" << szCrypto << endl;

	for (int nmacro = 1 ; nmacro < 12 + 1; nmacro++)
	{
		cout << nmacro << "\t";
		frotate_a(szCrypto, nrotate_a, szCryptoOut);
		cout << szCryptoOut << "\t";
		frotate_b(szCryptoOut, nrotate_b, szCrypto);
		cout << szCrypto << endl;
```

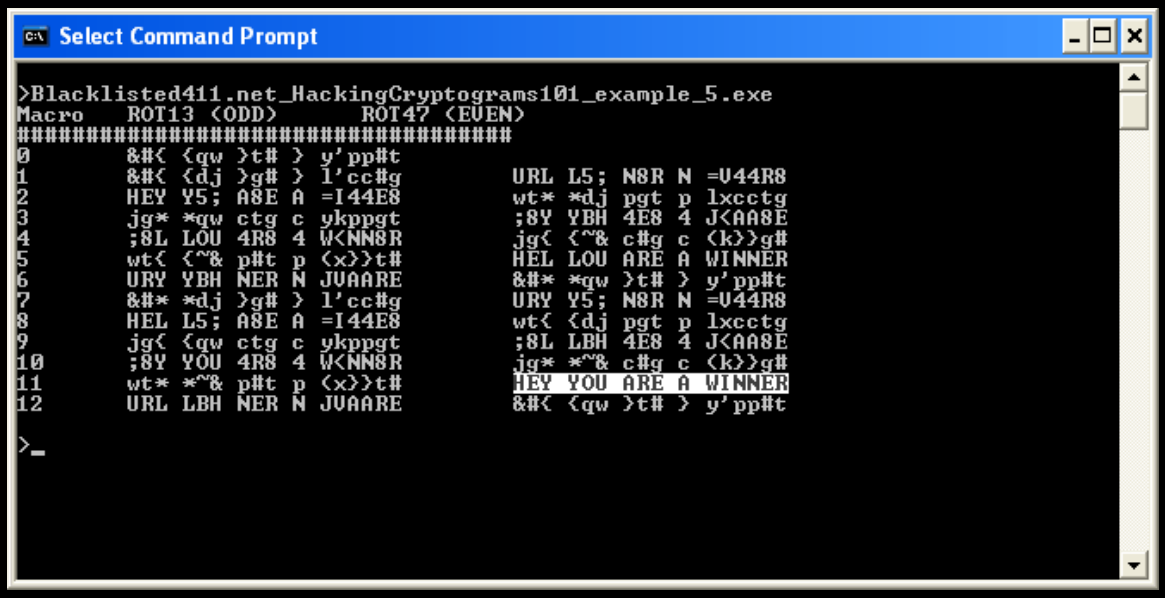
```
		}

}

void main(int argc, char* argv[])
{
		// here we are setting up our variables for use
		char szCrypto[255]="\0";
		// here we are copying our cryptogram into a variable
		strcpy(szCrypto,"Hello World");

		// rot(x) with caeser cipher A-Z, a-z including ROT13
		//frotate_a_ex(szCrypto);

		// rot(x) including ROT47
		//frotate_b_ex(szCrypto);

		frotate_c_ex(szCrypto);

}
```

After you've compiled and executed it you will see these results that match the original chart above.

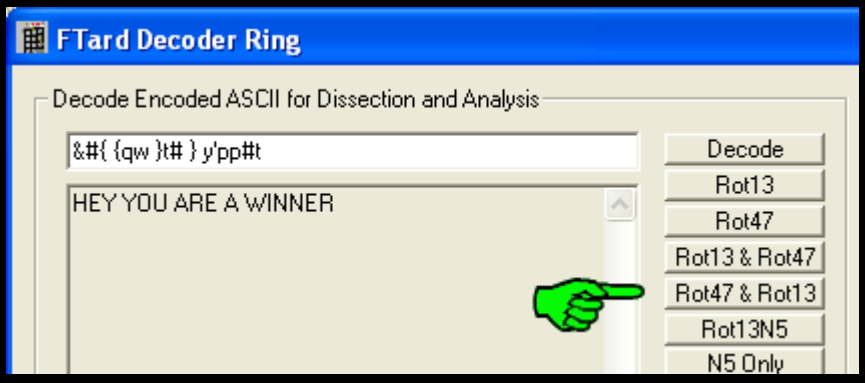


So I heard that no one submitted a valid solution to last month's cryptogram (tsk,tsk):
&#{ {qw }t# } y'pp#t

Now using this lesson it should be **really easy** to figure out ;)



Hopefully you guys will try a little harder next time in this month's cryptogram.



If you do solve it and feel more confident to play with more cryptograms check out this column's "*Fitzgerald cryptogram*" started in .NET Edition 5 below.

This concludes Lesson 5 of ***Hacking Cryptograms 101***. If you encounter a cryptogram or what you suspect to be a secret message that interests you either because you are stuck solving it or for any other reason please send me an email. See you next edition!

Keeping it 'rael,
Israel Torres

<u>The Fitzgerald Cryptogram</u>

The test of a first-rate intelligence is the ability to hold two opposed ideas in mind at the same time and still retain the ability to function. - F. Scott Fitzgerald

04 00 00 00 00 00
16 18 21 22 24 30
01 10 17 21 24 40
60 64 89 00 00 00
10 17 33 00 00 00

Email your solution to israel@israeltorres.org

Source Code

Source code can be downloaded here:
Zip
http://blacklisted411.israeltorres.org/Blacklisted411.net_HackingCryptograms101_example_5.zip

CPP
http://blacklisted411.israeltorres.org/Blacklisted411.net_HackingCryptograms101_example_5.cpp

MD5 hash of the source code can be validated at:
http://blacklisted411.israeltorres.org/Blacklisted411.net_HackingCryptograms101_example_5.md5.txt

> **note:** you may have to copy and paste the addresses above if they wrap around!

References and Related Links

ROT13
http://en.wikipedia.org/wiki/ROT13

FTard Decoder Ring
http://tools.israeltorres.org/FTardCryptoSuite.php#FTardDecoderRing

The Hacking Cryptograms 101 Articles Collection
http://blacklisted411.israeltorres.org/

Hacking A CD Check Function

By Israel Torres <israel@israeltorres.org>

Disclaimer: *Be Responsible For Your Own Actions .This document has been censored for your protection.*

The other day I went to look for a decent Sudoku game for Windows because I no longer wanted to carry around the 2 inch thick *1001 Sudoku* book I bought the week prior. It was really inconvenient and heavy and wasn't as portable as I thought it would be. I figured something to run on my laptop would suit my portability issues a lot better. I shopped around at a few retailers without any luck around the same time I decided to get the book. It was surprising to see that the top gaming companies out there didn't have anything out to date for such a popular game – seriously most book stores have a few shelves dedicated to this game alone. So the following week I was looking in the game area of one of my favorite [undisclosed] retailers and there it was. Not really in the section I buy from but from the cheaper jewel-case-only software section (aka the *cheap shelf*). The same section I promised myself I wouldn't buy from again because last two times the games contained spyware or just plain sucked compared to their packaging artwork *(never judge a game by the pretty pictures on the box)*. With much hesitation I picked it up read it over and bought it.

The features promised on the back of the package were really good compared to most online versions of the game, or any shareware versions – it even came with a built in solver to solve offline games. I quickly read the manual to see if there were any weird warnings that weren't on the outside of the package… and what do you know:

> …requires that all programs be installed from an account with Administrator rights…

That figures. I usually never install anything from a privileged account and always feel strange when a game bought from the cheap shelf REQUIRES it. Great - this usually means I have to test it on a test machine first and check to see if there are any undocumented surprises like outbound packets leaking information, phoning home, etc.

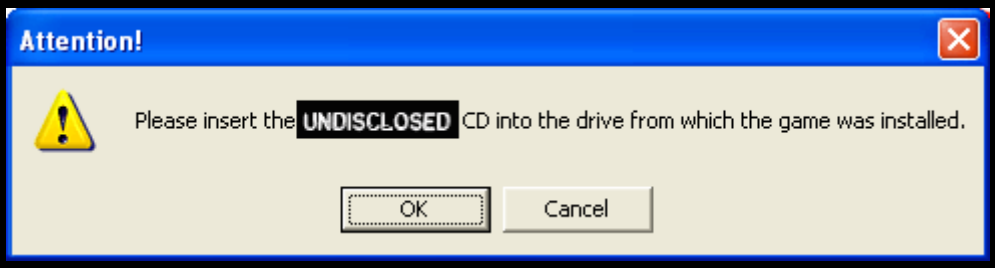
It installs fine and I fire it up, the splash screen takes a lot longer than it should and the menu is really basic and easy to follow. I monitor it for a while and don't notice anything weird trying to get out to the Internet, or suspicious activity happening in the background. I open up the installation directory and notice a bunch of VB6 DLL's I examine the main executable by throwing it into a hex editor and confirm it is written in VB6.

```
00071c00h: 00 00 00 00 2E 3D FB FC FA A0 68 10 A7 38 08 00 ; .....=ûüú h.§8..
00071c10h: 2B 33 71 B5 43 3A 5C 50 72 6F 67 72 61 6D 20 46 ; +3q┐C:\Program F
00071c20h: 69 6C 65 73 5C 4D 69 63 72 6F 73 6F 66 74 20 56 ; iles\Microsoft V
00071c30h: 69 73 75 61 6C 20 53 74 75 64 69 6F 5C 56 42 39 ; isual Studio\VB9
00071c40h: 38 5C 56 42 36 2E 4F 4C 42 00 00 00 56 42 00 00 ; 8\VB6.OLB...VB..
```

> C:\Program Files\Microsoft Visual Studio\VB98\VB

I sweep over the rest of the program quickly and don't notice anything immediately so I reboot the test system to see if there are any post-boot triggers. It comes up fine. I pick and poke a few more times and don't run across anything suspicious and conclude that running this game wouldn't put my laptop into a more vulnerable posture than it already is. Satisfied with this I remove the CD and install it on my laptop system as recommended by the instructions – of course I tried to install it as a user first without success as expected.

I put the CD away and forgot about the game until later that evening. I thought about playing Sudoku and fired it up by clicking the icon in my quick start toolbar. It came up as expected (after it dropped my volume down to start its soundtrack) and it prompted me to choose a player identity and click one of the menu game items. I clicked on the simple version and immediately was surprised to see the [undisclosed] "CD Check" message box pop up:




```
---------------------------
Attention!
---------------------------
Please insert the [UNDISCLOSED] CD into the drive from which the game was
installed.
---------------------------
OK   Cancel
---------------------------
```

I thought to myself, geeze are really people still doing this – especially on a cheap game?

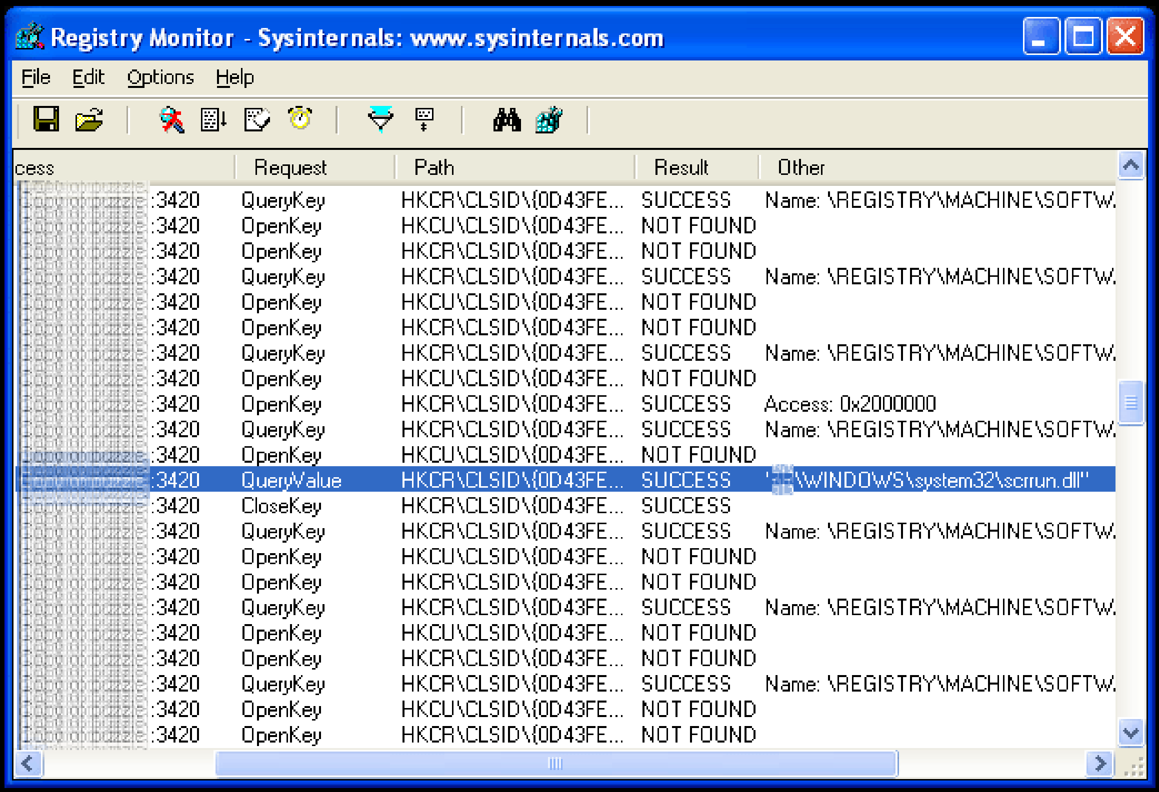
Instinctively my mind switched from wanting-to-play-Sudoku mode to wanting-to-hack-Sudoku-application mode *almost* instantly.

I clicked on OK just to see if that would shut it up. No luck there. I clicked Cancel and it terminated the whole game. As far as I was concerned this was completely unacceptable. There was no way I could return this game due to store policy and putting any more effort into going about getting my money back was already into the red. I don't have the luxury of carrying a spare CD-ROM compatible drive whenever I want to play, nor should I have to since the entire game installs itself on my hard disk. Someone at this company decided some time along the way that putting in a CD-check for this game would do their part in piracy prevention at the everyday user's inconvenience (*namely mine*). Things were about to change as far as I was concerned…

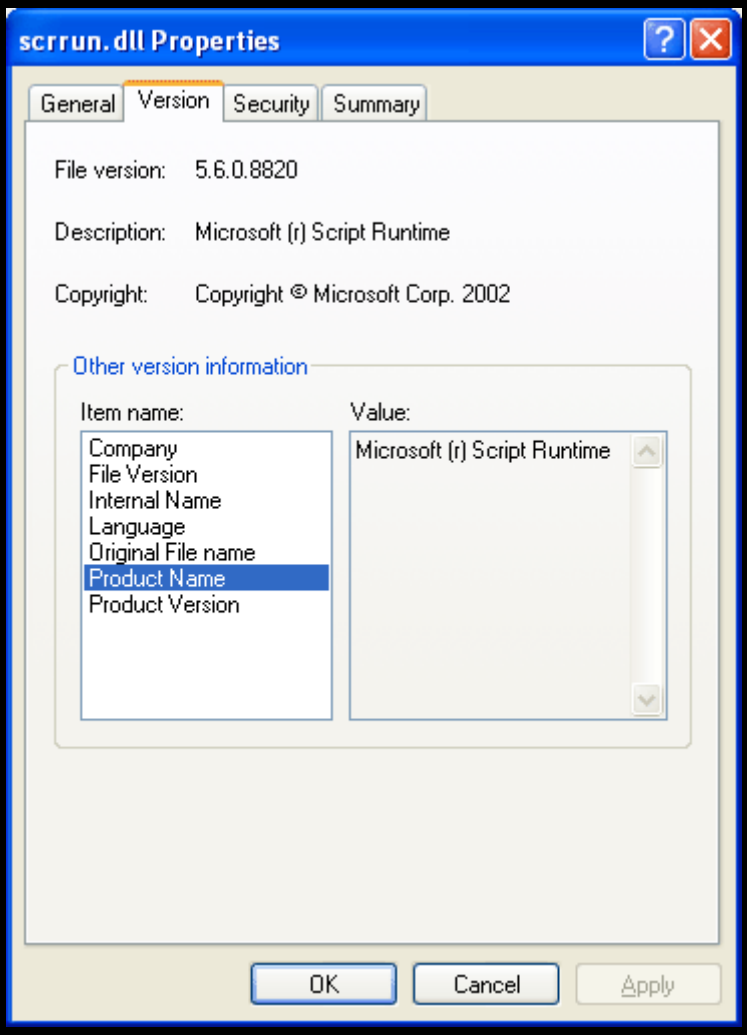
I took the simplest route (to reinvention prevention) and knocked on Google to see if there was a patch for this game that would make it "friendlier". I had my doubts since it was from the cheap shelf and I probably had one of 40 copies worldwide. My doubts were correct, no one had heard of this game. I also figured that installing a patch would put my machine at more risk than it was worth even if I had found one. Some of the patches out there have some creepy stuff in them. If anything I would check it out to see what the patch was doing and do it myself since it was just for my single system. Just to be super safe I

read over the document and EULA to make sure there wasn't anything about patching their software, and there wasn't. Good for me and good for you.

I took a breath then created a temporary directory, filled it with a handful of useful tools for finding out what an application is doing (Filemon, Regmon, Depends, Spy++ just to name a few). Then began the slow process of finding out what all was involved with this CD-check. There wasn't an INI file present in the installation directory so more than likely there were settings in the registry that it was initializing itself with upon invocation. I fired up Regmon, filtered by filename, got to the point where it was going to ask for the CD used ALT-TAB to switch tasks to start and stop Regmon and examined the resulting log:



I noticed one file that was being queried more than once and it is SCRRUN.DLL. I searched the system for the file and checked the properties on it:



Its description is **Microsoft ® Script Runtime** -the keyword being **script**. From this point it is easy enough to deduce that they are using Visual Basic Script / **VBScript** around the same time they are doing the CD-Check. Now to verify and see if we can find this somewhere in the executable file.

I drag the executable to my hex editor (*hex-fu time*) and search for the hexadecimal representation of the word “Script”. Depending on which editor you are using you may need to convert the word correctly so that it is found if it is there:

Conversion of the word Script	
Script	1. The word Script
S c r i p t	2. With spaces between each character
53 63 72 69 70 74	3. Converted to hexadecimal values
53 00 63 00 72 00 69 00 70 00 74	4. With 00h in between each character.

Doing this **53 00 63 00 72 00 69 00 70 00 74** is the hexadecimal representation for the word Script. Type this in and search for it.

```
00075100h: 30 00 00 00 06 00 00 00 20 00 76 00 39 00 00 00 ; 0....... .v.9...
00075110h: 34 00 00 00 53 00 63 00 72 00 69 00 70 00 74 00 ; 4...S.c.r.i.p.t.
00075120h: 69 00 6E 00 67 00 2E 00 46 00 69 00 6C 00 65 00 ; i.n.g...F.i.l.e.
00075130h: 53 00 79 00 73 00 74 00 65 00 6D 00 4F 00 62 00 ; S.y.s.t.e.m.O.b.
00075140h: 6A 00 65 00 63 00 74 00 00 00 00 00 44 00 72 00 ; j.e.c.t.....D.r.
00075150h: 69 00 76 00 65 00 73 00 00 00 00 00 04 00 00 00 ; i.v.e.s.........
00075160h: 41 00 3A 00 00 00 00 00 49 00 73 00 52 00 65 00 ; A.:.....I.s.R.e.
00075170h: 61 00 64 00 79 00 00 00 56 00 6F 00 6C 00 75 00 ; a.d.y...V.o.l.u.
00075180h: 6D 00 65 00 4E 00 61 00 6D 00 65 00 00 00 00 00 ; m.e.N.a.m.e.....
00075190h: 0C 00 00 00 53 00 55 00 44 00 4F 00 4B 00 55 00 ; ....S.U.D.O.K.U.
000751a0h: 00 00 00 00 14 00 00 00 41 00 74 00 74 00 65 00 ; ........A.t.t.e.
000751b0h: 6E 00 74 00 69 00 6F 00 6E 00 21 00 00 00 00 00 ; n.t.i.o.n.!.....
000751c0h: 24 00 00 00 50 00 6C 00 65 00 61 00 73 00 65 00 ; $...P.l.e.a.s.e.
000751d0h: 20 00 69 00 6E 00 73 00 65 00 72 00 74 00 20 00 ;  .i.n.s.e.r.t. .
000751e0h: 74 00 68 00 65 00 20 00 00 00 00 00 6A 00 00 00 ; t.h.e. .....j...
000751f0h: 20 00 43 00 44 00 20 00 69 00 6E 00 74 00 6F 00 ;  .C.D. .i.n.t.o.
00075200h: 20 00 74 00 68 00 65 00 20 00 64 00 72 00 69 00 ;  .t.h.e. .d.r.i.
00075210h: 76 00 65 00 20 00 66 00 72 00 6F 00 6D 00 20 00 ; v.e. .f.r.o.m. .
00075220h: 77 00 68 00 69 00 63 00 68 00 20 00 74 00 68 00 ; w.h.i.c.h. .t.h.
00075230h: 65 00 20 00 67 00 61 00 6D 00 65 00 20 00 77 00 ; e. .g.a.m.e. .w.
00075240h: 61 00 73 00 20 00 69 00 6E 00 73 00 74 00 61 00 ; a.s. .i.n.s.t.a.
00075250h: 6C 00 6C 00 65 00 64 00 2E 00 00 00 1E 00 00 00 ; l.l.e.d.........
```

… and I immediately hit **pay dirt**! Quickly I can see the and extra the text:

4
Scripting FileSystemObject
Drives
A:
IsReady
VolumeName
SUDOKU
Attention!
$
Please Insert the
j
CD into the drive from which the game was installed

I see the message box (MsgBox) message that is being thrown up when the conditional isn't being met, and I can edit it to say whatever I'd like – but what I'd really like to do is not have it come up at all. After making a backup I first try and remove it by overwriting it with 00's.

```
00075100h: 30 00 00 00 06 00 00 00 20 00 76 00 39 00 00 00 ; 0....... .v.9...
00075110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00075120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00075130h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00075140h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00075150h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00075160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00075170h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00075180h: 00 00 00 00 00 00 00 00 00 00 05 00 00 00 00 00 ; ................
00075190h: 0C 00 00 00 53 00 55 00 44 00 4F 00 4B 00 55 00 ; ....S.U.D.O.K.U.
000751a0h: 00 00 00 00 14 00 00 00 41 00 74 00 74 00 65 00 ; ........A.t.t.e.
000751b0h: 6E 00 74 00 69 00 6F 00 6E 00 21 00 00 00 00 00 ; n.t.i.o.n.!.....
000751c0h: 24 00 00 00 50 00 6C 00 65 00 61 00 73 00 65 00 ; $...P.l.e.a.s.e.
000751d0h: 20 00 69 00 6E 00 73 00 65 00 72 00 74 00 20 00 ;  .i.n.s.e.r.t. .
000751e0h: 74 00 68 00 65 00 20 00 00 00 00 00 6A 00 00 00 ; t.h.e. .....j...
000751f0h: 20 00 43 00 44 00 20 00 69 00 6E 00 74 00 6F 00 ;  .C.D. .i.n.t.o.
00075200h: 20 00 74 00 68 00 65 00 20 00 64 00 72 00 69 00 ;  .t.h.e. .d.r.i.
00075210h: 76 00 65 00 20 00 66 00 72 00 6F 00 6D 00 20 00 ; v.e. .f.r.o.m. .
00075220h: 77 00 68 00 69 00 63 00 68 00 20 00 74 00 68 00 ; w.h.i.c.h. .t.h.
00075230h: 65 00 20 00 67 00 61 00 6D 00 65 00 20 00 77 00 ; e. .g.a.m.e. .w.
00075240h: 61 00 73 00 20 00 69 00 6E 00 73 00 74 00 61 00 ; a.s. .i.n.s.t.a.
00075250h: 6C 00 6C 00 65 00 64 00 2E 00 00 00 1E 00 00 00 ; l.l.e.d.........
```

This doesn't work and I get script errors within the program that the object couldn't be created correctly.

Sudoku

Run-time error '429':

ActiveX component can't create object

OK

After running into this error message I try and think of how the logic is being set up and think of the pseudo-code:

‘ VBScript Pseudo-Code for checking the drives for a CD-ROM with the volume name SUDOKU
‘
Create **Scripting FileSystemObject**

Check the **Drives** for an inserted (**IsReady**) CD (4) with the **VolumeName** SUDOKU
‘

Ok, that looks like what is happening, I check out Microsoft’s Visual Basic and VBScript references on these functions and write up the following VBScript code following the pseudo-code above:

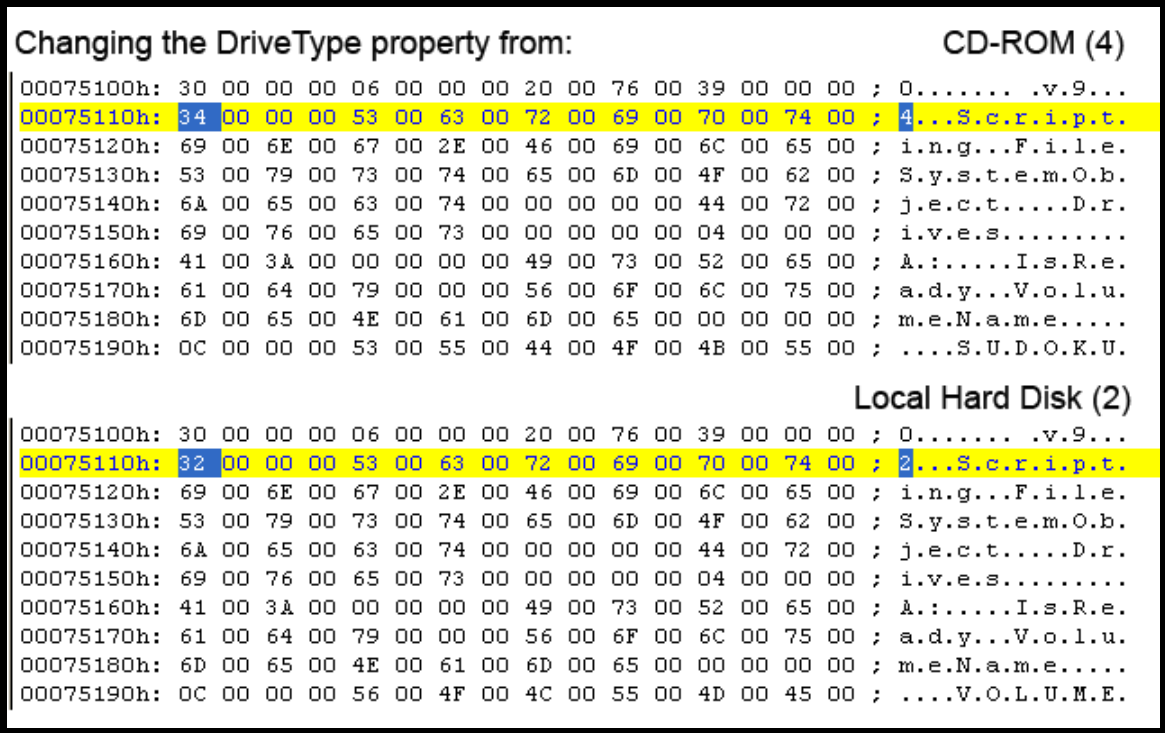
```
' Blacklisted411.net_Hacking_A_CD_Check_Function.vbs
' Hacking A CD Check Function
' By Israel Torres < israel@israeltorres.org >
'
' This example demonstrates how this CD-Check is checking its requirements
' It is not suggested to do it this way because it can easily be hacked by:
' Changing ReqDriveType from 4 to 2 (CD to Hard Disk)
' Changing ReqVolumeName from "SUDOKU" to "VOLUME" (arbitrary)
' ... allows the CD-Check function to be defeated within the compiled binary.

' Set up variables for use
Option Explicit
Dim fs, dc, d
Dim ReqDriveType, ReqVolumeName
Dim flagCD

' Declare variable values
flagCD = false

ReqDriveType = 4
' Defined Drive Types
' 0 = "Unknown"
' 1 = "Removable"
' 2 = "Fixed"
' 3 = "Network"
' 4 = "CD-ROM"
' 5 = "RAM Disk"

ReqVolumeName = "SUDOKU"

' Instantiate the FileSystemObject
Set fs = CreateObject("Scripting.FileSystemObject")

' Get the machine's Drive listing
Set dc = fs.Drives

' Loop through our drives until we find:
' A CD-ROM with the Volumename SUDOKU
For Each d in dc
        If d.DriveType = ReqDriveType Then
                If d.IsReady Then
                        If d.VolumeName = ReqVolumeName Then
                                flagCD = true
```

```
                        End If
                Else
                        MsgBox "CD Drive Not Ready"
            End If
        End If
Next

If flagCD = true Then
        MsgBox "Requirements Found (Check Verified)"
Else
        MsgBox "Requirements Not Found"
End If
```

I test it a few times with the CD Inserted and the CD ejected and it looks good and accurate to the behavior in the compiled code. It also looks like they didn't bother resetting the flags during enumeration or really anything other than check for anything other than a CD-ROM with a Volume Name of SUDOKU.

Going one step at a time I change one of my hard disk volume names to SUDOKU in preparation to "spoof" the Volume Name after I cripple the drivetype so the conditional will "think" everything is ok and all conditions are met.

Using my hex-fu I throw the binary into my hex editor, search for Script (**53 00 63 00 72 00 69 00 70 00 74 00)** and change the value of **4** (ASCII 52 decimal -> 34 hexadecimal) to the value of **2** (ASCII 50 decimal -> 32 hexadecimal). I save the binary after making a backup of the original (*important when experimenting*) and click on the executable to see if things look good.



Things work just as planned – I am no longer receiving the error message requesting to insert the CD-ROM. This fulfills my needs, but one last thing I'd like to change is the Volume Name requirement from

SUDOKU to something else – Why? - Because I can. Personally I like to keep my volume labels simple and generic and not give more information to someone who may be shoulder surfing when I am not looking. I think of other six letter words that would replace SUDOKU and VOLUME fits the bill. (*Changing this is demonstrated in the same graphic above*). I test it again with and without the CD and the CD-Check has been completely defeated – its logic functions will always be true regardless whether the CD is inserted or not. (*During my experimentation I also tried this with a USB key by modifying the DriveType to removable drive (1) and it worked as expected.*)

The lesson to be learned is that this type of CD check is not "secure" - just annoying. If you are the programmer that wrote this function I hope you have learned something from this. Sure the whole "low-hanging fruit" deterrence concept used to be an excuse – used to be.

Now onto playing Sudoku! ;)

Keeping it 'rael,
Israel Torres

Source Code

The VBScript Source code demonstrated herein can be downloaded here:
VBScript
http://blacklisted411.israeltorres.org/Blacklisted411.net_Hacking_A_CD_Check_Function.vbs

Zip Archive:
http://blacklisted411.israeltorres.org/Blacklisted411.net_Hacking_A_CD_Check_Function.zip

MD5 hash of the source code can be validated at:
http://blacklisted411.israeltorres.org/Blacklisted411.net_Hacking_A_CD_Check_Function.md5.txt

note: you may have to copy and paste the addresses above if they wrap around!

References and Related Links

UltraEdit 32
http://www.ultraedit.com/

Depends.exe
http://www.dependencywalker.com/

Regmon
http://www.sysinternals.com/Utilities/Regmon.html

FileMon
http://www.sysinternals.com/Utilities/Filemon.html

Spy++
http://msdn.microsoft.com/library/en-us/vcug98/html/_asug_home_page.3a_.spy.2b2b.asp
http://msdn.microsoft.com/msdnmag/issues/06/04/ManagedSpy/

1001 Sudoku **(SBN: 1560258837)**
http://www.amazon.com/gp/product/1560258837/qid=1142478374/sr=2-1/ref=pd_bbs_b_2_1/102-2000420-8412957?s=books&v=glance&n=283155

Microsoft Windows Scripting

http://msdn.microsoft.com/scripting/

Microsoft Visual Basic Reference
http://msdn.microsoft.com/library/en-us/VBRef98/html/vbmscLROverview.asp

FileSystemObject Object
- Provides access to a computer's file system.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbenlr98/html/vaobjfilesystemobject.asp

Drives Collection
- Read-only collection of all available drives.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbenlr98/html/vaobjDrives.asp

Drive Object
- Provides access to the properties of a particular disk drive or network share.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbenlr98/html/vaobjDrive.asp

IsReady Property
- Returns True if the specified drive is ready; False if it is not.
http://msdn.microsoft.com/library/en-us/vbenlr98/html/vaproIsReady.asp

VolumeName Property
- Sets or returns the volume name of the specified drive. Read/write.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbenlr98/html/vaprovolumename.asp

DriveType Property
- Returns a value indicating the type of a specified drive.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbenlr98/html/vaprodrivetype.asp

The name of the Sudoku game and retailer mentioned herein are undisclosed for everyone's protection.

[5] ==Cryptogram==

March Cryptogram
By Israel Torres <israel@israeltorres.org>

You have one simple cryptogram to solve:

```
1241501411561530401711571650401461571620401601541411711511561 47
```

Hint: It is really easy.

Your mission if you choose to accept it is solving this cryptogram (if you can). To qualify as a winner you need to submit two things:

1. The solved message
2. How you solved the cryptogram by describing each step, algorithm name, etc. The more information you submit the more we'll believe you.

Send both items 1 and 2 to crypto@blacklisted411.net with the subject of "cryptograms"

Winners will be posted in order of submittals based on correctness and completeness above.
Winners will be posted right before the next edition is released.
Keeping it 'rael,

Israel Torres

[6] ==Favorite Photo==

A mint mp3 player
http://www.i-hacked.com/index.php?option=content&task=view&id=216

[7] ==Links==

Hacking Games
http://www.hackits.de/ - Hacking challenges (Games, Puzzles, Etc)
http://www.hackthissite.org/ - Offering encryption, hacking, real world challenges, and more

Computer Security News
http://www.securityfocus.com - A leader in security related news

http://www.topix.net/tech/computer-security - A collaboration of computer security news from around the web

[8] ==Credits==

Chief Editor: Zachary Blackstone

Online Edition Editor: Ustler

Authors: Israel Torres.