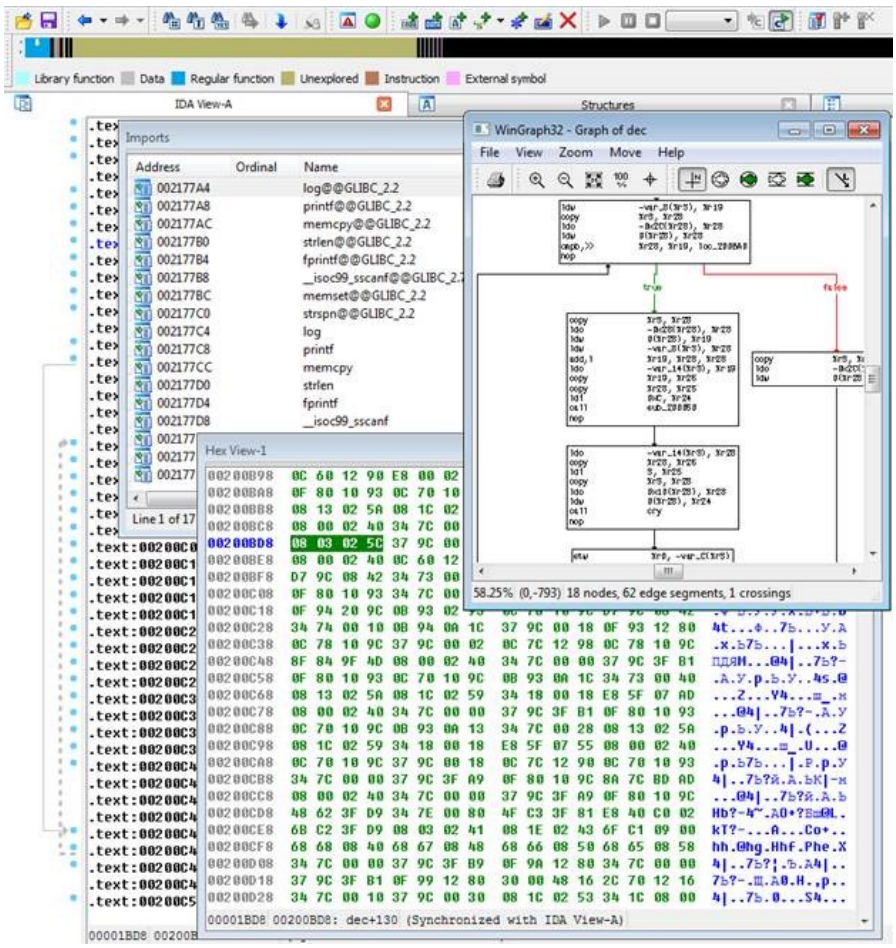
Positive Research Center

"for positive ideas"

WEDNESDAY, JULY 22, 2015

Best Reverser Write-Up: Analyzing Uncommon Firmware



While developing tasks for PHDays' contest in reverse engineering, we had a purpose of replicating real problems that RE specialists might face. At the same time we tried to avoid allowing cliche solutions.

Let us define what common reverse engineering tasks look like. Given an executable file for Windows (or Linux, MacOS or any other widely-used operating system). We can run it, watch it in a debugger, and twist it in virtual environments in any way possible. File format is known. The processor's instruction set is x86, AMD64 or ARM. Library functions and system calls are documented. The equipment can be accessed through the operating system only. Using tools like IDAPro and HexRays makes analysis of such applications very simple, while debug protection, virtual machines with their own instruction sets, and obfuscation could complicate the task. But large vendors hardly ever use any of those in their programs. So there's no point in developing a contest aimed at demonstrating skills that are rarely addressed in practice.

However, there's another area, where reverse engineering became more in-demand, that's firmware analysis. The input file (firmware) could be presented in any format, can be packed, encrypted. The operating system could be unpopular, or there could be no operating system at all. Parts of the code could not be changed with firmware updates. The processor could be based on any architecture. (For example, IDAPro "knows" not more than 100 different processors.) And of course, there's no documentation available, debugging or code execution cannot be performed—a firmware is presented, but there's no device.

Our contest's participants needed to analyze an executable file and find the correct key and the relative

email (any internet user was able to take part in the contest).

Part One: Loader

At the first stage, the input file is an ELF file compiled with a cross compiler for the PA-RISC architecture. IDA can work with this architecture, but not as good as with x86. Most requests to stack variables are not identified automatically, and you'll have to do it manually. At least you can see all the library functions (log, printf, memcpy, strlen, fprintf, sscanf, memset, strspn) and even symbolic names for some functions (c32, exk, cry, pad, dec, cen, dde). The program expects two input arguments: an email and key.

```
.text:002013E0                ldo             0xE0(%r19), %r25 # .LC8 # ". Usage: %s <email> <key>\n"
.text:002013E4                copy            %r28, %r24
.text:002013E8                call            _fprintf
```

It's not hard to figure out that the key should consist of two parts separated by the "-" character. The first part should consist of seven MIME64 characters (0-9A-Za-z+/), the second part of 32 hex characters that translate to 16 bytes.

```
.text:002014A0                ldo             0x114(%r19), %r25 # .LC10 # "%02x"
.text:002014A4                copy            %r28, %r24
.text:002014A8                call            _sscanf
```

Further we can see calls to c32 functions that result in:

```
t = c32(-1, argv[1], strlen(argv[1])+1)
k = ~c32(t, argv[2], strlen(argv[2])+1)
```

Name of the function is a hint: it's a CRC32 function, which is confirmed by the constant 0xEDB88320.

Next, we call the dde function (short for doDecrypt), and it receives the inverted output of the CRC32 function (encryption key) as the first argument, and the address and the size of the encrypted array as the second and third ones.

Decryption is performed by BTEA (block tiny encryption algorithm) based on the code taken from Wikipedia. We can guess that it's BTEA from the use of the constant DELTA==0x9E3779B9. It's also used in other algorithms on which BTEA is based on, but there are not many of them.

The key should be of 128-bit width, but we receive only 32 bits from CRC32. So we get three more DWORDs from the exk function (expand_key) by multiplying the previous value by the same DELTA.

However, the use of BTEA is uncommon. First of all, the algorithm supports a variable-width block size, and we use a block of 12-bytes width (there are processors that have 24-bit width registers and memory, then why should we use only powers of two). And in the second place, we switched encryption and decryption functions.

Since data stream is encrypted, cipher block chaining is applied. Enthropy is calculated for decrypted data in the cen function (calc_enthropy). If its value exceeds 7, the decryption result is considered incorrect and the program will exit.

The encryption key is 32-bit width, so it seems to be easily brute-forced. However, in order to check every key we need to decrypt 80 kilobytes of data, and then calculate enthropy. So brute-forcing the encryption key will take a lot of time.

But after the calculation, we call the pad function (strip_pad), which check and remove PKCS#7 padding. Due to CBC features, we need to decrypt only one block (the last one), extract N byte, check whether its range is between 1 and 12 (inclusive) and each of the last N bytes has value N. This allows reducing the number of operations needed to check one key. But if the last encrypted byte equals 1 (which is true for 1/256 keys), the check should be still performed.

The faster method is to assume that decoded data have a DWORD-aligned length (4 bytes). Then in the last DWORD of the last block there may be only one of three possible values: 0x04040404, 0x08080808 or 0x0C0C0C0C. By using heuristic and brute force methods you can run through all possible keys and find the right one in less than 20 minutes.

If all the checks after the decryption (entropy and the integrity of the padding) are successful, we call the fire_second_proc function, which simulates the launch of the second CPU and the loading of decrypted data of the firmware (modern devices usually have more than one processor—with different architectures).

If the second processor launches, it receives the user's email and 16 bytes with the second part of the key via the function send_auth_data. At this point we made a mistake: there was the size of the string with the email instead of the size of the second part of the key.

Part Two: Firmware

The analysis of the second part is a little bit more complicated. There was no ELF file, only a memory image —without headings, function names, and other metadata. Type of the processor and load address were unknown as well.

We thought of brute force as the algorithm of determining the processor architecture. Open in IDA, set the following type, and repeat until IDA shows something similar to a code. The brute force should lead to the conclusion that it is big-endian SPARC.

Now we need to determine the load address. The function 0x22E0 is not called, but it contains a lot of code. We can assume that is the entry point of the program, the start function.

In the third instruction of the start function, an unknown library function with one argument == 0x126F0 is called, and the same function is called from the start function four more times, always with arguments with similar values (0x12718, 0x12738, 0x12758, 0x12760). And in the middle of the program, starting from 0x2490, there are five lines with text messages:

```
00002490        .ascii "Firmware loaded, sending ok back."<0>
000024B8        .ascii "Failed to retrieve email."<0>
000024D8        .ascii "Failed to retrieve codes."<0>
000024F8        .ascii "Gratz!"<0>
00002500        .ascii "Sorry may be next time..."<0>
```

Assuming that the load address equals 0x126F0-0x2490 == 0x10260, then all the arguments will indicate the lines when calling the library function, and the unknown function turns out to be the printf function (or puts).

After changing the load base, the code will look something like this:

```
ROM:00012540                save    %sp, -0x2F8, %sp
ROM:00012544                set     aFirmwareLoaded, %o0 ! "Firmware loaded, sending ok back."
ROM:0001254C                call    puts
ROM:00012550                nop
ROM:00012554                set     0x0BA0BAB0, %o0
ROM:0001255C                call    sub_12194
ROM:00012560                nop
ROM:00012564                add     %fp, var_108, %g1
ROM:00012568                mov     %g1, %o0
ROM:0001256C                mov     0, %o1
ROM:00012570                mov     0x101, %o2
ROM:00012574                call    sub_24064
ROM:00012578                nop
ROM:0001257C                add     %fp, var_210, %g1
ROM:00012580                mov     %g1, %o0
ROM:00012584                mov     0, %o1
ROM:00012588                mov     0x101, %o2
ROM:0001258C                call    sub_24064
ROM:00012590                nop
ROM:00012594                add     %fp, var_108, %g1
ROM:00012598                mov     %g1, %o0
ROM:0001259C                call    sub_121BC
ROM:000125A0                nop
ROM:000125A4                mov     %o0, %g1
ROM:000125A8                cmp     %g1, -1
ROM:000125AC                bne     loc_125CC
ROM:000125B0                nop
ROM:000125B4                set     aFailedToRetrie, %o0 ! "Failed to retrieve email."
ROM:000125BC                call    puts
```

The value of 0x0BA0BAB0, transmitted to the function sub_12194, can be found in the first part of the task, in the function fire_second_proc, and is compared with what we obtain from read_pipe_u32 (). Thus sub_12194 should be called write_pipe_u32.

Similarly, two calls of the library function sub_24064 are memset (someVar, 0, 0x101) for the email and code, while sub_121BC is read_pipe_str (), reversed write_pipe_str () from the first part.

The first function (at offset 0 or address 0x10260) has typical constants of MD5_Init:

Next to the call to MD5_Init, it is easy to detect the function MD5_Update () and MD5_Final (), preceded by the call to the library strlen ().

```
ROM:00010260 MD5_Init:
ROM:00010260
ROM:00010260 ctx             =  0x44
ROM:00010260
ROM:00010260                 save    %sp, -0x60, %sp
ROM:00010264                 st      %i0, [%fp+ctx]
ROM:00010268                 ld      [%fp+ctx], %g1
ROM:0001026C                 clr     [%g1+4]
ROM:00010270                 ld      [%fp+ctx], %g1
ROM:00010274                 ld      [%g1+4], %g2
ROM:00010278                 ld      [%fp+ctx], %g1
ROM:0001027C                 st      %g2, [%g1]
ROM:00010280                 ld      [%fp+ctx], %g1
ROM:00010284                 set     0x67452301, %g2
ROM:0001028C                 st      %g2, [%g1+8]
ROM:00010290                 ld      [%fp+ctx], %g1
ROM:00010294                 set     0xEFCDAB89, %g2
ROM:0001029C                 st      %g2, [%g1+0xC]
ROM:000102A0                 ld      [%fp+ctx], %g1
ROM:000102A4                 set     0x98BADCFE, %g2
ROM:000102AC                 st      %g2, [%g1+0x10]
ROM:000102B0                 ld      [%fp+ctx], %g1
ROM:000102B4                 set     0x10325476, %g2
ROM:000102BC                 st      %g2, [%g1+0x14]
ROM:000102C0                 restore
ROM:000102C4                 retl
ROM:000102C8                 nop
ROM:000102C8 ! End of function MD5_Init
```

```
ROM:00012604                 add     %fp, MD5_CTX, %g1
ROM:00012608                 mov     %g1, %o0
ROM:0001260C                 call    MD5_Init
ROM:00012610                 nop
ROM:00012614                 add     %fp, email, %g1
ROM:00012618                 mov     %g1, %o0
ROM:0001261C                 call    strlen
ROM:00012620                 nop
ROM:00012624                 mov     %o0, %g3
ROM:00012628                 add     %fp, MD5_CTX, %g2
ROM:0001262C                 add     %fp, email, %g1
ROM:00012630                 mov     %g2, %o0
ROM:00012634                 mov     %g1, %o1
ROM:00012638                 mov     %g3, %o2
ROM:0001263C                 call    MD5_Update
ROM:00012640                 nop
ROM:00012644                 add     %fp, MD5_CTX, %g1
ROM:00012648                 mov     %g1, %o0
ROM:0001264C                 call    MD5_Final
```

Not too

many unknown functions are left in the start() function.

The sub_12480 function reverses the byte array of specified length. In fact, it's memrev, which receives a code array input of 16 bytes.

Obviously, the sub_24040 function checks whether the code is correct. The arguments transfer the calculated value of MD5(email), the array filled in function sub_12394, and the number 16. It could be a call to memcmp!

The real trick is happening in sub_12394. There is almost no hints there, but the algorithm is described by one phrase—the multiplication of binary matrix of the 128 by the binary vector of 128. The matrix is stored in the firmware at 0x240B8.

Thus, the code is correct if MD5(email) == matrix_mul_vector (matrix, code).

```
ROM:0001263C                call    MD5_Update
ROM:00012640                nop
ROM:00012644                add     %fp, MD5_CTX, %g1
ROM:00012648                mov     %g1, %o0
ROM:0001264C                call    MD5_Final
ROM:00012650                nop
ROM:00012654                ldd     [%fp+var_220], %g2
ROM:00012658                std     %g2, [%fp+var_288]
ROM:0001265C                ldd     [%fp+var_218], %g2
ROM:00012660                std     %g2, [%fp+var_280]
ROM:00012664                add     %fp, code, %g1
ROM:00012668                mov     %g1, %o0
ROM:0001266C                mov     0x10, %o1
ROM:00012670                call    sub_12480
ROM:00012674                nop
ROM:00012678                add     %fp, code, %g2
ROM:0001267C                add     %fp, var_298, %g1
ROM:00012680                mov     %g2, %o0
ROM:00012684                mov     %g1, %o1
ROM:00012688                call    sub_12394
ROM:0001268C                nop
ROM:00012690                add     %fp, var_298, %g2
ROM:00012694                add     %fp, var_288, %g1
ROM:00012698                mov     %g2, %o0
ROM:0001269C                mov     %g1, %o1
ROM:000126A0                mov     0x10, %o2
ROM:000126A4                call    sub_24040
ROM:000126A8                nop
ROM:000126AC                mov     %o0, %g1
ROM:000126B0                cmp     %g1, 0
ROM:000126B4                bne     loc_126D4
ROM:000126B8                nop
ROM:000126BC                set     aGratz, %o0     ! "Gratz!"
ROM:000126C4                call    puts
ROM:000126C8                nop
ROM:000126CC                ba      loc_126E4
ROM:000126D0                nop
ROM:000126D4 ! ---------------------------------------------------------------------------
ROM:000126D4
ROM:000126D4 loc_126D4:                             ! CODE XREF: start+174↑j
ROM:000126D4                set     aSorryMayBeNext, %o0 ! "Sorry may be next time..."
ROM:000126DC                call    puts
```

Calculating the Key

To find the correct value of the code, you need to solve a system of binary equations described by the matrix, where the right-hand side are the relevant bits of the MD5(email). If you forgot linear algebra: this is easily solved by Gaussian elimination.

If the right-hand side of the key is known (32 hexadecimal characters), we can try to guess the first seven characters so that the CRC32 calculation result was equal to the value found for the key BTEA. There are about 1024 of such values, and they can be quickly obtained by brute-force, or by converting CRC32 and checking valid characters.

Now you need to put everything together and get the key that will pass all the checks and will be recognized as valid by our verifier :)
We were afraid that no one would be able to solve the task from the beginning to the end. Fortunately, Victor Alyushin showed that our fears were groundless. You can find his write-up on the task at http://nightsite.info/blog/16542-phdays-2015-best-reverser.html. This is the second time Victor Alyushin has won the contest (he was the winner in 2013 as well).

A participant who wished to remain anonymous solved a part of the task and took second place.

Thanks to all participants!

Автор: Positive Research на 12:42 AM

Ярлыки: best reverser, phdays, security, write-up

7 comments:

Kate Mark August 3, 2015 at 3:57 AM

HACK ATM AND BECOME RICH TODAY
How to hack an ATM MACHINE or BANK ACCOUNT
You can hack and break into a bank's security ATM Machine without carrying guns or any weapon.
How is this possible? First of all we have to learn about the manual hacking of ATM MACHINES and BANKING ACCOUNTS HOW THE ATM MACHINE WORKS. If you have been to the bank you find out that the money in the ATM MACHINE is being filled right inside the house where the machine is built with enough security.to hack this machine We have develop the special blank ATM Card which you can use in any ATM Machine around the world. this card is been programmed and can withdraw 2000 USD within 24 hours in any currency your country make use of. The card will make the security camera malfunction at that particular time until you are done with the transaction you can never be trace. getting the card you will forward the company your address details so we can proceed to send the card to you once you agree to the terms and conditions. you can

contact us on email now atmmachinehackes@gmail.com

Reply

Replies

Marian Law August 21, 2015 at 7:19 AM

HOW I GET A LEGIT LOAN @ 2% INTEREST RATE
I was not sure of getting a legit loan lender online But when i could not face my Debt any more, my son was on hospital bed for surgery that involve huge money and i also needed some money to refinance and get a good home then i have to seeks for Assistance from friends and when there was no hope any more i decide to go online to seek a loan and i find VICTORIA LAWSON Loan company (marianlawson@outlook.com) with 2% interest Rate and applied immediately with my details as directed. Within seven Days of my application She wired my loan amount with No hidden charges and i could take care of my son medical bills, Renew my rent bill and pay off my debt. I will advice every loan seeker to contact Victoria Lawson Company with marianlawson@outlook.com For easy and safe transaction.

*Full Name:_____

*Address:_____

*Tell:_____

*loan amount:_____

*Loan duration:_____

*Country:_____

*Purpose of loan:_____

*Monthly Income:_____

*Occupation_____

*Next of kins :_____

*Email :_____

Contact her company Via Email: marianlawson@outlook.com

johnson December 1, 2015 at 7:34 PM

Hello everyone, i want to share my testimony on how i got my ATM black magic card which have change my life today. i was once living on the street where by things were so hard for me, even to pay off my bills was very difficult for me i have to park off my apartment and start sleeping on the street of Vegas. i tried all i could do to secure a job but all went in vain because i was from the black side of America. so i decided to browse through on my phone for jobs online where i got an advert on Hackers advertising a Blank ATM card which can be used to hack any ATM Machine all over the world, i never thought this could be real because most advert on the internet are based on fraud, so i decided to give this a try and look where it will lead me to if it can change my life for good. i contacted this hackers and they told me they are from Australia and also they have branch all over the world in which they use in developing there ATM CARDS, this is real and not a scam it have help me out. to cut the story short this men who were geeks and also experts at ATM repairs, programming and execution who taught me various tips and tricks about breaking into an ATM Machine with a Blank ATM card.i applied for the Blank ATM card and it was delivered to me within 3 days and i did as i was told to and today my life have change from a street walker to my house, there is no ATM MACHINES this BLANK ATM CARD CANNOT penetrate into it because it have been programmed with various tools and software before it will be send to you. my life have really change and i want to share this to the world, i know this is illegal but also a smart way of living Big because the government cannot help us so we have to help our self. if you also want this BLANK ATM CARD i want you to contact the Hackers email on johnsonatmblackmagiccreditcard@yahoo.com OR Call +2348104244364 and your life will never remain the same.

Reply

Kathie Roper August 9, 2015 at 11:33 PM

TESTIMONY ON HOW I WAS RESCUED BY A GOD-SENT LENDER..

Hello everyone, am writing this Testimony because am really grateful for what Mason Diego did for me and my family, when I thought there was no hope he came and make my family feel alive again by lending us loan at a very low interest rate of 2%. Well I have been searching for a loan to settle my debts for the past three months all I met scammed and took my money until I finally met a God sent Lender. I never thought that there are still genuine loan lenders on the internet but to my greatest surprise i got my loan without wasting much time so if you are out there looking for a loan of any amount i would advise you to email Mr Diego via: { diegoloancompany@yahoo.com } and be free of internet scams. thanks... Kathie Roper from California, USA.

vulnerability scanner (1)
vulns (7)
waf (7)
waf bypas (1)
wasc (1)
web (1)
Web search (1)
web security (10)
web vulnerabilities (2)
web-vulnerabilities (1)
WinCC (2)
windows (4)
Windows 8 (3)
windows 8.1 (1)
Windows GDI (1)
Windows Kernel (1)
windows patchguard (1)
wireless controllers (1)
Wireshark (1)
workshop (1)
write-up (2)
XCCDF (1)
xenserver (1)
xpc (1)
XSS (4)
cc10 (1)

CONTRIBUTORS

Positive Research
Rublev Sergey
Andrew Abramov
houseofdabus
repdet
TeckLord
Denis Baranov
Feodor Kulishov
yunusov
Yuri

Reply

Sai Santosh August 18, 2015 at 1:31 AM

Reverse engineering and re engineering concepts work well with the people who are technically well versed and who can crack any technical issue like a jack. More challenges are in the way of data organization where software like hadoop helps a lot. I came to know about the reverse engineering when I was attending a course at hadoop training in hyderabad

Reply

Leo Meg August 29, 2015 at 12:12 AM

GET RICH IN LESS THAN 3DAYS
It all depends on how fast
you can be to get the new PROGRAMMED
blank ATM card that is capable of hacking
into any ATM machine,anywhere in the
world. I got to know about this BLANK
ATM CARD when I was searching for job
online about a month ago..It has really
changed my life for good and now I can
say I'm rich and I can never be poor
again. The least money I get in a day with
it is about $50,000.(fifty thousand USD)
Every now and then I keeping pumping
money into my account. Though is
illegal,there is no risk of being
caught ,because it has been programmed
in such a way that it is not traceable,it
also has a technique that makes it
impossible for the CCTV to detect
you..For details on how to get yours today,
email the hackers on : (oceancardhackers@gmail.com
). Tell your loved once too, and
start to live large. That's the simple
testimony of how my life changed for
good...Love you all ...contact them now, the email address
again is : oceancardhackers@gmail.com

or call +2348168199202

Reply

chi November 18, 2015 at 2:29 PM

CHI YOUNG TECH of the email address. (chiyoungtechworld@gmail.com) its at it again! Cool way to have financial freedom!!! Are you tired of living a poor life,then here is the opportunity you have been waiting for. Get the new ATM BLANK CARD that can hack any ATM MACHINE and withdraw money from any account. You do not require anybody's account number before you can use it. Although you and I knows that its illegal,there is no risk using it. It has SPECIAL FEATURES, that makes the machine unable to detect this very card,and its transaction is can't be traced . You can use it anywhere in the world. With this card,you can withdraw nothing less than $50,000 in a day. So to get the card,reach the hackers via email address : chiyoungtechworld@gmail.com.

Reply

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)