.:: Reversity GuideLines Speech ::.

Cryptography and Reverse Engineering

Author: **Evilcry alias Giuseppe Bonfà**
E-Mail: **evilcodecave (at) gmail (dot) com / evilcry (at) gmail (dot) com**
Web Site: http://evilcry.altervista.org

Index

Introduction To The Speech

The topic we will cover today is **Cryptography** and **Applied Reverse Engineering**, the preferred OS will be obviously Windows, but Concepts and Techniques can be applied also for UNIX Like platforms, and partially on some kind of Crypto Hardware Devices such FPGAs.
It's important to well define what kind of Speech will be this, surely not an How To Crack speech, here we are Reversers and Not Crackers, people with Crack intentions can join to #Crackversity :)

So what is the principal objective of this meet?
To give a good and detailed Information about Reverse Engineering for Cryptography, you will find

- **Basics needed to understand Critical Security Applications**
- **Basic Knowledge on Cryptography**
- **Well Defined Learning Paths**
- **Pointers and Links Useful for the learning process**
- **Reversing hints for most common Protection Schemes**

The approach is a classical one, **CryptoRev.**

Meaning of the term CryptoRev

Many of you surely does not have ever heard the term CryptoRev, I've created this one to identify a specific Branch of Reverse Engineering, expressively designed for Software/Hardware targets that implements Crypto Secured Schemes.

Please note that CryptoRev could be taken as equivalent to Cryptanalysis, but it's important to specify that, Cryptanalysis Approach is used when you have the Algorithm, many times isolated from the context in which this one is applied.

For Example just immagine to have a new crypto algorithm, X Cipher..
The Crypographer will deals only with X's Vulnerabilities, Possible Attacks and Real Attacks.

CryptoReverser, starts from a lower level faces different difficulties, as:

- **Code Localization**
- **Code Identification**
- **Software Architecture Used**
- **Weak/Wrong Conceptual Implementations**

It's important to clarify the meaning of "Software Arch. Used", many times can happen that we are in front off Strong CryptoSignatures Schemes, that does not care about protecting Informations (Keys, Passwords, Generation Strings) into the Memory.

Actual State of Art of Applied Cryptography in Software Security Schemes

Actually, cryptography covers a truly important role in the most significant/professional **Software Protection Schemes**, 80% of them uses Classical Standardized algorithms such as **RSA, Blowfish, ElGamal, ECCs**, and a little part implements truly Dangerous home made algorithms or untested ones.

It's important to say that in the latest years fortunately, many big Software Houses (Autodesk for example) uses for its Software Protection Schemes, Strong Crypto Algorithms as for example ECDSA. This is a great trend, because implies that Sw Houses understood the necessity to protect Sensitive Strings (Passwords, Serial Numbers etc.) with Strong Algorithms.

This trend was also encouraged by the by the fact that we are in the era of **MASSIVE FRAMEWORKIZATION.**

Obviously as in all things there are Advantages and Disadvantages:

Advantages: Fast and Easy Coding, free of basical potential bugs, in our case, code errors/bugs in the effective Algorithm Implementations.

Disadvantages: Crypto Frameworkization leads to a big fundamental problem, many not well formed coders implements protection schemes with **Critical Conceptual Errors**.
In the worst cases, traces of Passwords or any other String to protect can be Sniffed from memory.

So you can understand that a well coded Set of Crypto APIs, is not enough because human Conceptual Architecture errors can lead a well designed code to the failure.

The Necessity of an RCE Approach for Crypto Sec Based Schemes

At this point, as you should understand, it's truly important for Software Houses (or Malicious Reversers) to know first of all the basical differences between common Security Routines and Crypto Based Schemes.

To understand better why this, just Imagine a 100000$ Protection Scheme, is in the interest of the SwHouse to have a Secure Prediction over his product, and now appears the figure of CryptoReverser.
This is only a field of possible applications, each Secure Communication software could be weak, think for example about a Plugin for MSN that Encrypts our conversations, if is used a badly implemented Algorithm the Key Exchangement Procedure leaves in the Memory or Net sensitive data that can be reconstructed by a sort of MITM Attack.

A Step Forward in the CryptoRev Scenario (Formation Guide Lines)

What is of critical importance to understand is that, CryptoReversing needs high knowledge of mathematics.

For a basic/intermediate CryptoReverser not all sub disciplines of math needs to be heavy studied, but for sure there are a few branches that are Fundamental:

- **Basic Mathematics**
- **Fundamental Mathematical Analysis**
- **Number theory its very important Modular Math**
- **Abstract algebra**
- **Probability Theory**

I'll report for you some link that should be read for a better understanding:

The Handbook Of Applied Cryptography is the first book that i suggest in particular the following chapters:

- **Chapter 2 - *Mathematics Background***
- **Chapter 3 - *Number-Theoretic Reference Problems***

here the link http://www.cacr.math.uwaterloo.ca/hac/

My hint is to read the entire book, because is the Bibble of Cryptography, and during the practice you will use it heavy!

Cryptography: Theory and Practice (Discrete Mathematics and Its Applications) (Hardcover) by Douglas R. Stinson (Author)

The Crypto Tutorial

- http://www.antilles.k12.vi.us/math/cryptotut/home.htm

Pure mathematics is fundamental but we are here to learn, CryptoReversing, so all this mathematics needs a Coding counterpart, so here a little list of links for **Coding Theory**

http://www.adastral.ucl.ac.uk/~helger/crypto/link/coding_theory/

After mathematics, we have obviously to know the most common Crypto Algorithms involved in secure protections but this is not an easy task and needs Time and Good Directions..

- **Basic Ciphers (Caesar, Substitution, PolyAlphabetic, F-Pos like)**
- **Shift Register Sequences (Kluwer)**
- **Block Ciphers**
- **Public Key Cryptography**
- **Discrete Logarithm Based Problems**
- **RSA Based Schemes (Needs a study really Deep)**
- **Stream Ciphers**
- **Digital Signatures**
- **Key Management Techniques**
- **Elliptic Curve Cryptography**

The Practical Part

Frameworks and Libraries

Due to the extreme complexity involved into actual Trusted CryptoSystems, Coders makes heavy use of Libraries and Function Collections. Essentially during our reversing sessions we can meet varius kinds of Logical Organizationsof these Libraries, some of them deals directly with the mathematics necessary for the Crypto Algorithm others offer a faster and easiest way of coding, you have directly the Algorithm and the only thing you need is the Key and the Plain/Cipher-Text.

Let's see the most spreaded libraries..

- **MIRACL** Multiprecision Integer and Rational Arithmetic C/C++ Library
- **Crypto++ Library -**
- **Win32 OpenSSL -**
- **Widowds CryptoAPI -**
- **NET Framework cryptography**
- **The Legion of the Bouncy Castle**
- **free-LIP by Arjen K. Lenstra**
- **NTL**

Obviously there are many many others Libraries less or more trusted,
yeah on Cryptography is important also to have Trusted code, many are the Statal Cryptographic Backdoors, for example on CryptoAPI the famous _NSA Key :)

MIRACL - is a great Big Number Library which implements all of the primitives necessary to design Big Number Cryptography, as for example RSA Schemes, it's not diffused on commercial schemes because need very well qualified CryptoCoders, possibility of wrong implementations are High.

Crypto++ - is a collection of cryptographic schemes, and is preferred especially on MFC Applications (this is a my obervation based on experience) especially when is required AES or RSA Protection.

OpenSSL- is fundamentally designed to provide a commercial-grade, full-featured tookit for SSL/TLS, this library is not used for Product Key Attivation, but is really intersting from the Reverser / Security Researcher poit of view because, the major part of High Confidental Software (such as for Trusted Networking) implements OpenSSL.

Windows CryptoAPI – Are really flexible and powerful they can find application in a big variety of security tasks, such as managing certificates, and developing customizable public key infrastructures. Certificate and smart card enrollment, certificate management, and custom module developmen. This set of APIs is a bit different from the others because they are based on CSP (Crypto Service Provider)
Reversing Targets which implements CryptoApi is a bit more complex due to the different organization and diversification, indeed we can encounter there sub-services:

- **CryptoAPI Tools** - code signing, signature verification
- **CSPs** - Cryptographic Service Providers (CSP) that contain implementations of cryptographic standards and algorithms.
- **CAPICOM** - Authenticode digital signatures, file-based credentials, certificates management, support for AES
- **WinTrust** – Is based essentially on a multi purpose function**WinVerifyTrust**() that performs a Trust Verification by calling a Trust Provider.

NET Framework cryptography - The .NET Framework provides implementations of many standard cryptographic algorithms. These algorithms are easy to use and have the safest possible default properties. In addition, the .NET Framework cryptography model of object inheritance, stream design, and configuration are extremely extensible. Accomplished tasks are the same of CryptoApis, Secret-key encryption, PK, Signing, Hashing.

First Step – Library Identification

The first fundamental step for a CryptoReverser is to identify the library used. So we have to disassemble the target and if not packed the first check can be done on Strings.Now we will see some common sign that can help to recognize the previously seen libraries

Let's suppose that we don't have any IDA Signature..

MIRACL: This library could be easly recognized, due to the fact thas a direct management of BigNums, stored in a struct of that kind:

1)

```
typedef struct {
int Size;
char bytes[24]; } big_num;
```

So in Asm we have the following situation

```
push 0
call _mirvar
mov dword ptr ds:[414B70],eax
```

So by watching the stack we can find the following corrispondences

- **+04h Absolute Address of the BigNum**
- **+0Ch Absolute Value of The BigNum Stored in Little Endian format**

2)

The second best known and used method to search about the

mov dword ptr [esi+eax*4+20], 17h

where the 17h, is a function identifier. Black_Eye already compiled a list of Indexes that can be downloaded here http://beatrix2004.free.fr/pamplemousse/magic.table.txt

3)
IDA Signatures -> http://149.156.124.1/~cauchy//get.php?id=1

Crypto++: By analysing String table of Crypto++ is easy to confuse Crypto++ with Win CryptoApi, cause the usual presence of **CryptoAcquireContext()** and **CryptoReleaseContext()**. Singular Crypto++ functions are easy to detect, just take a look here:

?AVException**@CryptoPP@@**
unicode 0, <BYTE iv[CryptoPP::AES::BLOCKSIZE] =>

It's clear that **@CryptoPP@@** stands for Crypto++

Win32 OpenSSL: Also this library is easy to detect, from Import Section search about **LIBEAY32**, and **OPENSSL_add_all_algorithms_noconf**

Widowds CryptoAPI: As previously said CryptoApi are based on Services Providers, so in every case they needs to acquire the relative context, to do this is called **CryptoAcquireContext().**

NET Framework cryptography: Just matter of Reflector, the rest is clear :)

Basical Most Common Implementations

Today the 90% of **Product Key Activation Schemes** are based upon the following algorithms

- **RSA**
- **BlowFish**
- **ElGamal**

Fast RSA

The first and most famous Key Validation is **RSA**, used for classical SerialNumber activation and also on Keyfile Protections.

RSA is a public_key_cipher borned at M.I.T. on 1978 thank to the collaboration between Ron **R**ivest, Adi **S**hamir e Les **A**ndleman. This is an asymmetric cryptosystem which bases is "power" on a series of features that the prime numbers had.

Due to the fact that RSA is an Asymmetric one we have Public and Private Parameters, and SerialNumber activation is accomplished essentially in 3 steps:

- **Generate an RSA Key Pair**
- **Export the Private Key (the checker)**
- **Import the Public Key (who needs to be validated)**

First reverser's task is to know/enstabish parameters that are public and/or private, here a list of them

N = P * Q
N -> Public
P,Q -> Private

O(n)=(p-1)*(q-1) -> Not Known

e = O(n) -> (Not know)

d -> Decrypt key (Known)

At this table we also add X (the plaintext) and Y (the ciphertext). Now we are into the heart of the algorithm!, the famous:

C=M^e mod n

Where M is the Message to crypt and C is the Crypted message. I think that you are all able to obtain the inverse formula, but to be complet:

M=C^d mod n

Usually in reversing we are in that situation:

- **We know the correct but crypted serial, in other words we know C**
- **We know the modulus N, that was used to crypt our serial.**
- **We know E (with E and P,Q we can found D).**

There is a tool around here, explicitly done for reversers, is RsaTool2 of The Egoiste TMG, that is able to factorize N, to obtain P and Q. Next is matter of Mdular math, usually I implement little C programs with MIRACL libs to do that task.

How to recognize RSA

RSA could be implemented in various ways also without libraries because as you've seen is easy, but into Commercial Apps, **MIRACL** and **Crypto++** are the most used.

RSA Pointers

Crackmes
Lockless 3 Crackme
http://crackmes.de/users/amenesia/basis2_rsa___wiener/
http://crackmes.de/users/eod/ninja_crackme_by_bart__xtreeme/
http://crackmes.de/users/thigo/tmg_official_keygenme_2/
http://crackmes.de/users/tsc/rsa_me/
http://crackmes.de/users/thigo/thigocrkme/

Papers
http://reteam.org/papers/e74.pdf
http://www.codeproject.com/KB/security/ProductActivation.aspx

Fast Blowfish

Blowfish is a cipher based on blocks and uses also a secret-simmetric key (so there is only a key, used both for encryption and decryption. Blowfish is based also on a Feistel-Network, for people that do not know what a Feistel, iterates a specific function a pre choised number of times, usually 16. Blowfish has also two fundamental elements, S-Boxes and P-Boxes, that are arrays of Constant values.

Many times, into protection schemes, to recognize some famous algorithm, it's used to check for some constant values (for example in MDx, TEA, Ghost, etc.). Into Blowfish we have some starting values, for is P-array:

0x243F6A88L

and for S-box1 the first element is:

0xD1310BA6L

Could happen, that some Programs implements Modified versions of that algo. So tools like PEId or RDG could fail the detection, so pay attention!!

Architecture of Blowfish, could be easly resumed in 3 functions:

- **Blowfish_Initialization( )** for Expansion Step (or initialization function, this is the most complex phase)
- **Blowfish_encrypt( )**
- **Blowfish_decrypt( )**

Blowfish functions have all the Context parameter, a struct of that kind:

```
typedef struct {
unsigned long P[16 + 2];
unsigned long S[4][256]; } BLOWFISH_CTX;
```

Used for S-Boxes and P-Boxes.

1)
Blowfish_Init(&blowfishctx, &szSetKey, 0x1f);

szSetKey is the key to expand, and 0x1f his lenght

2)

Now Encryption could be called : **Blowfish_Encrypt(&blowfishctx, &dtA, &dtB);**

&dtA and &dtB can composes our P-Boxesand obviously comes from the Expansion Step.

3)

Blowfish_Decrypt(BLOWFISH_CTX *ctx, unsigned long *xl, unsigned long *xr)

Last stage is the Decryption, as the encryption function takes as input CTX, DataL and DataR, and as return value we have DataL and DataR decrypted.

In the totality of cases, Product Validation Schemes, uses the Init Function and the Encrypt, so reverser work in this case is easy, he just have to implement a decryption function and needs to know the key (usually sniffed from memory).
Not very frequently, program inverts the functions, so program implements Init and Decrypt, and Reverser need the Encryption one. I've said this to let you pay attention on functions that you really have.

The worst case that you can see is the Modified Blowfish, in other words protection schemes does not implements the canonic blowfish, but a version changed in some aspect. Usually the modifications that you'll meet are not many, here the most common: :

- **Altered number of Feistel rounds (commonly is 32)**
- **S-boxes have different strating values, this is also an anti-detect system for automated recognizing software**
- **F function is modified and frequently returns S[2][d]-S[1][b])^(S[0][c]+S[3][a]**

My hint for code developers is to NOT implement any form of Blowfish.

Blowfish Pointers

Crackmes

Raskcrackme

http://www.crackmes.de/users/silver/silvers_dx_crackme_1/
http://www.crackmes.de/users/adjiang/keygenmemd5tean_blowfishcrc32_boom/
http://www.crackmes.de/users/blowfish/blowfish_crackme1/
http://www.crackmes.de/users/blowfish/blowfish_crackme2/

Papers

http://www.reteam.org/papers/e75.pdf
http://www.schneier.com/paper-blowfish-fse.html
http://tripteam.free.fr/files/Blowfish.pdf

Fast ElGamal

ElGamal is Public Key Algorithm, and is an asymmetric one, that bases is power on DL (Discrete Logarithm problem).

The algorithm could be resumed in the following steps:

- **Domain Parameter Generation**
- **Key Generation**
- **Encryption**
- **Decryption**

Domain parameters are two, p (large prime number) and g (generator that comes form GF(p))

Key Generation depends on d(p,g)

- **x Randomly generated**
- **y = g^x MOD p**

y is the public key, x is the secret key.

Signing of message M (call is Encryption)

- **Randomly generated k**
- **a=g^k Mod p**
- **b= (M- x * a) * $k^{(-1)}$ (mod p - 1) (M is our message)**
- **a and b are the Signatures of M**

Essentially ElGamal is a great choise for Product Key Protection, exist some truly well coded examples of that, for example SecureCRT. Security and efficiency of ElGamal Schemes is based on the Discrete Logarithm Problem.

Attack Strategy

As should be clear, the first (and unique) strategy to attack DLP, is the determination of unknown parameters, in particular the Secret Key **x**

y= g^x Mod p

DLP can be breaked by using

- **index calculus method**
- **collision search method**

In the practice are used **Pollard's rho** or **Pohlig-Hellman** a bit more faster.

When we have the secret key, and no Memory Obfuscated Parameters, keygenning became a matter of Math Reversing. Keygeneration attacks are accomplished with MIRACL lib.

ElGamal Pointers

Crackmes

http://crackmes.de/users/bublic/elgamal_keygenme_1/
http://crackmes.de/users/thigo/tmg_official_keygenme_3/

Papers

http://www.ccs.neu.edu/home/yiannis/papers/eg.ps
http://dictionary.zdnet.com/definition/El+Gamal+algorithm.html
http://en.wikipedia.org/wiki/ElGamal_encryption