Hacking .NET Applications: The Black Arts

Jon McCoy
www.DigitalBodyGuard.com
THIS WILL COVER

- How-To Attack .NET Applications
- Tools and Methodology of Attacking
- Overcome “secure” .NET Applications
- Building KeyGen/Crack/Hacks/Malware
- Reverse Engineering for Protection
Attacking/Cracking
IN MEM  ||||  ON DISK

blackhat USA + 2011
ATTACK OVERVIEW

Attack on Disk
- Access Logic
- Infect Logic
- Hook Logic
- Decompile
- Recompile
- Debug

Attack in Memory/Runtime
- Inject Structure
- Navigate Structure
- Edit/Control Structure
Attack The Source

In Memory OR On Disk

Find the weak spot
Subvert the Logic/State
Control what you need
ATTACKING ON DISK
DEMO

GrayWolf – IL_Spy – Reflector
101 - ATTACK ON DISK

Connect/Open - Access Code

- Decompile - Get code/tech
- Infect - Change the target's code
- Exploit - Take advantage
- Remold Application - WIN
101 - Recon

EHSHELL

.NET Framework
Ver 3.5

Un-0bfu$ca7ed

Crash Reporting
Watson

Coded in C#
101 - Recon

File Location
C:\Windows\ehome\ehshell.dll

StrongName KEY
d:\w7rtm.public.x86fre\internal\strongnamekeys\fake\windows.snk

Registry
CurrentUser OR LocalMachine
SOFTWARE\Microsoft\Windows\CurrentVersion\Media Center

Web Host Address
www.microsoft.com/WindowsMedia/Services/2003/10/10/movie
CRACK THE APP

- Flip The Check
- Cut The Logic
- Return True
- Access Value
- Set Value is “True”
SET VALUE TO “TRUE”

bool Registered = false;

If(a != b)
bool IsRegistered()
{
    Return "TRUE"
}
bool ValidPassword(int x) {
    ShowKey(Pass);
    Return (x==Pass);
}
CRACK

CRACK the weak
Media Center
public static bool CheckPin(string pin)
{
    ParentalControl.Settings.PIN = null;
    ParentalControl.Settings.Load();
    string text = ParentalControl.Settings.PIN;
    if (text == null)
    {
        return 1;
    }
    if (text.Length > 0)
    {
        if (text.get_Chars(0) == 58)
        {
            goto Block_6;
        }
    }
    ParentalControlPin.StoreNewPin(text);
    return text == pin;
    Block_6:
    return text == ParentalControlPin.HashForPin(pin);
}
CRACK

DEMO
CRACK THE KEY
Attack the STRONG

“I’m sure they protected the registration check”
CRACK THE KEY

Complex Math == Complex Math
Public/Private == Change Key
Challenge == Make it EZ
3/B==Name*C == ASK what is /B?
Call Server == Hack the Call
COMPLEX MATH

1. Chop up the Math
2. Attack the Weak
3. ????????????
4. Profit
CHANGE THE KEY

If you can beat them
Why join them

Key = “123456ABCDE”
CHALLENGE

Complex Math
OR
Control the Challenge
REG CODE

Name: JON DOE

Code: 5G9P3

WIN
1. Seed the Request
2. Fake the Reply
3. Kill the Call
DEMO

CRACK A KEY
IL – Intermediate Language
Code of the Matrix |||| NEW ASM
IT CAN’T BE THAT EZ

NO
PROTECTION ON DISK

Protection - Security by 0b$cur17y

- Code Obfuscation
- Logic Obfuscation
- Shells / Packers / Encrypted(code)
- Unmanaged calls

SHUTDOWN

Decompilation
public static bool $ca7ed$
{
    try
    {
        bool flag = ( & 4) == 4;
    }
    catch (Exception exception)
    {
        $ca7ed$(arg_0F_0, box(Application));
        throw;
    }
    return flag;
}
PROTECTION ON DISK

Protection – Security by security
  Signed code (1028 bit CRYPTO)
  Verify the creator
  ACLs........ M$ stuff

This can SHUTDOWN
Tampering
IT CAN’T BE THAT EZ

NO

YES
NEW ATTACK VECTOR

SIGNED CODE HACKING

Signed code is based on

- Private Key - 1028 bit
- Signed Hash of Code

SIGNED CODE CHECKING IS OFF BY DEFAULT
NEW ATTACK VECTOR
ASM THE OLD IS NEW

Shell Code - ASM
UNmanaged
NO .NET Security

Attack from ASM(unmanaged)
The Gate is Down
Attacking / Cracking
In MEM |||| On Disk

blackhat USA + 2011
ATTACKING .NET APPLICATIONS: AT RUNTIME
ATTACKING APPS

Gain Full Access
Reverse Engineer
Attack (on Disk or in MEM)
Take out the “Security”
Control the Program
DEMO: GOD MODE

Inject and Control
PAST TALKS

Hacking .NET Application: A Runtime Attack

Control the Runtime
Control the Application
IF YOU’RE NOT A HACKER
WHY SHOULD YOU CARE?

Defend your Applications

Defend your Systems

Verify your Tools\Programs
VERIFY YOUR APPLICATIONS

What is the Crypto & KEY
What info is it sending home
Does it have Backdoors?
Is your data Secure?
REVERSE ENGINEERING

What is going on?
What technology is used?
Any Malware?
AM I safe?
REVERSE ENGINEERING

Hack your applications

Don't be helpless
DON'T LOOK
SECURITY

The Login security check is:

- Does \( A == B \)
- Does \( \text{MD5\%5} == X \)
- Is the Pass the Crypto Key
DATA LEAK

The Data sent home is

■ Application Info
■ User / Serial Number
■ Security / System Data
KEY

The Crypto Key is

- A Hard Coded Key
- The Licence Number
- A MD5 Hash of the Pass
- 6Salt 6MD5 Hash of the Pass
The Crypto is
- DES 64
- Tripple DES 192
- Rijndael AES 256
- Home MIX (secure/unsecure)
BAD SOFTWARE

Y U NO CRYPT

Y ASK 4 PASSWORD
THE OLD IS NEW AGAIN
ASM-SHELLS

Pointers in .NET

What are they good for?
Are they safe?
What about the Runtime?
So ASM-Shells....
MALWARE T1M3

Protection (Shell Crypto)
Attack (Unmanaged Calls)
Protection (Obfuscated Code)
Fake (Signed DLL Protection)

blackhat USA + 2011
Intelligent names

Code style

Don’t use loops

Don’t use one area for your Vars

Access the normal program

Link to Events

Use Timers

Call back into your target

Spread out your Vars and code
this.filesToAdd = new List<string>();
base..ctor();
this.InitializeComponent();
this.Text = this.AssemblyTitle + " " + this.AssemblyVersion;
if ((int)par.Length == 1)
{
    if (par[0].Contains("en"))
    {
        this.langParEn = 1;
    }
}
private void x03a69b6bf16c508c()
{
    var arg_AA_0 = this.xef9c50c23fdde0e7;
    object[] array = new object[][][6];
    array[0] = this.x991baaf3e2f1814.getTranslation("mejfjaagfahgfaog", 127490266));
    array[1] = string.Intern(x1110bdd110cdcea4._d577);
    array[2] = box(System.Int32, this.xe25232a1a3e32);
    array[3] = string.Intern(x1110bdd110cdcea4._d577);
    array[4] = box(System.Int32, this.xe25232a1a3e32);
    array[5] = string.Intern(x1
    arg_AA_0.Text = string.C
}
PROTECTION FOR WHO?

Obfu$ca73
WHAT M$ DID RIGHT

Un-obfuscated Code
∑ Good for user security
∑ User can see what they are running

.NET Framework Security
∑ Targeted Security Access
∑ Protect the Computer from the app

Giving Reduced Rights Inside Code
∑ Put venerable code in a box
∑ Mitigate Risk, Segment Risk
MixModeCode – Bad for security
- This allows unmanaged code
- This breaks out of .NET security

GAC & Native Image Override
- Removes ability to secure code

Not Hash Checking Code
- Good for hackers
ATTACKING APPS

- Read my papers: Reflections Hidden Power & Attacking .NET at Runtime
- Watch 2010 Presentations on Attacking .NET
  DefCon 18, AppSec-DC, DojoCon
- Look up Presentations and Research from
  Andrew Willson, Erez Ezule, Arndet Mandent
- Use tools: Visual Studio/MonoDev Reflector/
  GrayWolf/ILspy/…/ILASM/ILDASM
FIN
MORE INFORMATION @:
www.DigitalBodyGuard.com

FIN = 1
HACKER VS ATTACKER