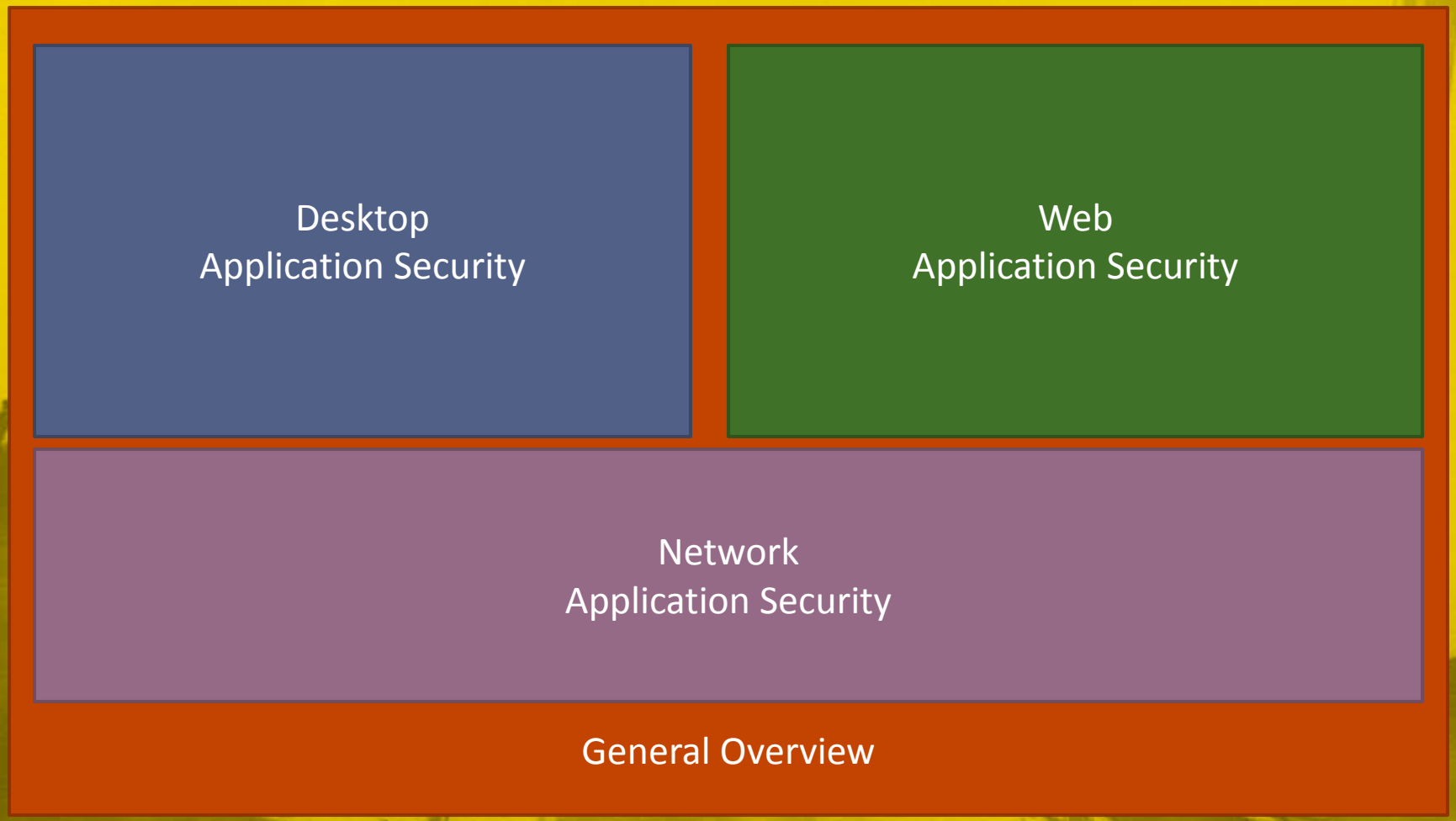
Contents of this presentation are for your educational purpose. Knowledge sharing is good, as long as you keep it within a good circle. People Incidents represented in this presentation are purely fictitious. Resemblance to anyone living or dead is purely coincidental.

<Writing Secure Code>

Jairam Ramesh
Security Research Consultant | Microsoft Corporation

<Agenda>


Desktop
Application Security
Web
Application Security
Network
Application Security
General Overview

<Not going to cover>

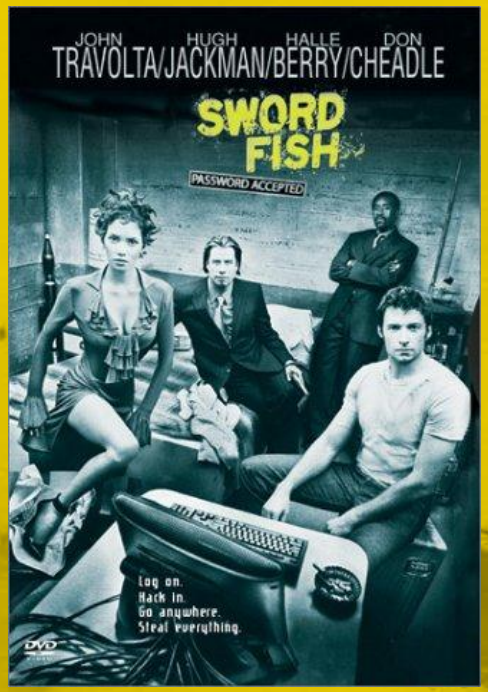
- Admin based protection
- Server level protection
- For asp.net try and play around with security features in IIS

<General Overview>

- Concept of Computation has evolved
- The user domain has increased
- Everything is computerized
- Banking, Finance, Business, etc..
- Anything is possible

<General Overview>

- Attracts bounty hunters
- More of Organized Crime
- Your Application is My Application



<General Overview>

- The more secure you make it the more interesting it is.
- Some people find it “Amusing” some for “respect”

<General Overview>

- I have a desktop software what can others do?
  - Inject code
  - Perform remote exploits
  - Use it as a carrier
  - What not?

<General Overview>

- I have a server client software what can others do?
  - Capture sensitive data
  - Interrupt communication stream
  - Use it as a carrier
  - What not?

<General Overview>

- I have a web based software what can others do?
  - Inject code
  - Phish data
  - Acquire User Credentials
  - What not?

<Buffer Overflow>

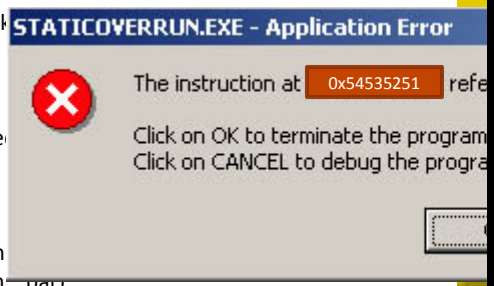
- Most Common bug
  - First Major Exploit: 1988 finger Worm.
- Often leads to total compromise of host
- Mostly due to poor coding practices
- Attacker needs to have expertise and patience.

<Buffer Overflow>

Static Buffer Overrun

- A static buffer overrun occurs when a buffer declared on the stack is overwritten by copying data larger than the buffer.

```
#include <stdio.h>
#include <string.h>
void foo(const char* input)
{
   char buf[10];
   printf("My stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n\n");
   strcpy(buf, input);
   printf("%s\n", buf);
   printf("Now the stack looks lik
}
void bar(void)
{
   printf("Augh! I've been hacke
}
int main(int argc, char* argv[])
{
   printf("Address of foo = %p\n
   printf("Address of bar = %p\n
   foo(argv[1]);
   return 0;
}
```




```
d:\devstudio\myprojects\staticoverrun perl HackOverrun.pl
Address of foo = 00401000
Address of bar = 00401045
My stack looks like:
77FB80DB
77F94E68
7FFDF000
0012FF80
0040108A
00410ECA

ABCDEFGHIJKLMNOPE@
Now the stack looks like:
44434241
48474645
4C4B4A49
504F4E4D
00401045
00410ECA
Augh! I've been hacked!
```

```
$arg = "ABCDEFGHIJKLMNOP".“\x45\x10\x40";
$cmd = "StaticOverrun ".$arg;
system($cmd);
```

Desktop Application Security

<Buffer Overflow>

Static Buffer Overrun

- What if an attacker replaces the initial string with the malicious code and returns the pointer to the beginning of the stack.
- Or an attacker could even make it run at another memory location.

<Buffer Overflow>

Static Buffer Overrun - Prevention

- Black Box testing
- Adding canary
- Stack Guard
- Address Obfuscation
- Avoid strcpy, strcat, sprintf
- Code Patch for linux and solaris
- Mark Stack as non execute
- eEye Retina, ISIC

<Buffer Overflow>

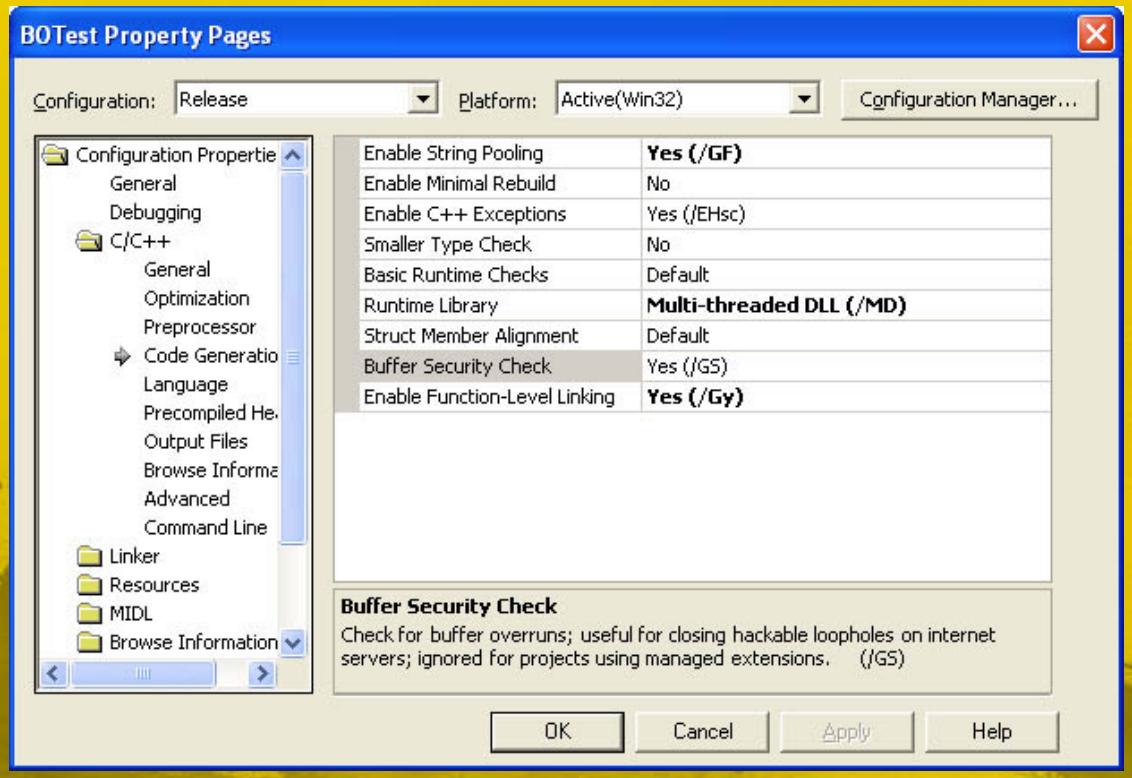
Static Buffer Overrun – Family Attacks

- Heap Overruns
- Array Indexing
- Format String
- Unicode and ANSI Buffer Size Mismatch
  (MultiByteToWideChar)

<Buffer Overflow>

Static Buffer Overrun – .net

- Enter the new /GS option in Visual C++ .NET. This new option, called the buffer security check, inserts special code into the application or DLL startup code, as well as special code into certain functions prolog and epilog code.



<ACL>

- Microsoft offers many means to limit who has access to what.
- Fundamental Part of NT, 2000, XP
- Literally your applications last backstop against an attack.
- If an attacker can access any resource his job is basically done

<ACL>

Detour for registery keys

```
#define MAX_BUFF (64)
#define MY_VALUE "SomeData"
BYTE bBuff[MAX_BUFF];
ZeroMemory(bBuff, MAX_BUFF);

// Open the registry.
HKEY hKey = NULL;
if (RegOpenKeyEx(HKEY_LOCAL_MACHINE,"Software\\MYCOMPANY",0,KEY_READ,&hKey) == ERROR_SUCCESS)
{
                // Determine how much data to read.
                DWORD cbBuff = 0;
                if (RegQueryValueEx(hKey,MY_VALUE,NULL,NULL,NULL,&cbBuff) == ERROR_SUCCESS)
                {
                                // Now read all the data.
                                if (RegQueryValueEx(hKey,MY_VALUE,NULL,NULL,bBuff,&cbBuff) ==ERROR_SUCCESS)
                                 {
                                                // Cool!
                                                // We have read the data from the registry.
                                }
                }
}
if (hKey)
RegCloseKey(hKey);
```

The code incorrectly assumes that the data held in the registry is no bigger than 64 bytes in size

RegQueryValueEx reads the data size from the registry, and the second call to RegQueryValueEx reads into the local buffer. A potential buffer overrun exists if this value is greater than 64 bytes.

<ACL>

Access Control List

- Windows NT and later contains two types of ACLs.
  - discretionary access control lists (DACLs) determines access rights to secured resources
  - system access control list (SACLs) determines audit policy for secured resources.

<ACL>

Access Control List

- What are the resources that can be secured using DACLs and SACLs:
  - Files & Directories
  - File Shares
  - Registry Keys
  - Shared Memory
  - Job Objects
  - Mutexes
  - Named Pipes
  - Printers
  - Semaphores
  - Active Directory Objects

<ACL>

Access Control List

- To define an appropriate ACL for your application
  - Determine the resources you use
  - Determine the business defined access requirements
  - Determine the appropriate access control technology

<Running with least Privilage>

- Identify the privilege needed for your application.
- The lesser the privilege the lesser will be the impact caused by a flaw in your application.
- All Windows NT, Windows 2000, and Windows XP user accounts have privileges, or rights, that allow or disallow certain privileged operations affecting an entire computer rather than specific objects.

Display Name	Internal Name	#define (Winnt.h)
Backup Files And Directories	SeBackupPrivilege	SE_BACKUP_NAME
Act As Part Of The Operating System	SeTcbPrivilege	SE_TCB_NAME
Debug Programs	SeDebugPrivilege	SE_DEBUG_NAME
Replace A Process Level Token	SeAssignPrimaryTokenPrivilege	SE_ASSIGNPRIMARYTOKEN_NAME
Increase Quotas	SeIncreaseQuotaPrivilege	SE_INCREASE_QUOTA_NAME

<Using Tokens>

- When a user logs into a windows machine and the account is authenticated a token is created.
- Token is applied to every process and thread that the user starts up.
- Token contains users Session ID
- Thus token can identify the capabilities of the user.
- Current Windows token contains SID's and privileges

<Random Numbers>

- Quite often your program needs a random number.

```
#include <stdlib.h>
void main()
 {
          srand(12366);
          for (int i = 0; i < 10; i++)
          {
                    int i = rand() % 100;
                    printf("%d ", i);
          }
}
```

```
52 4 26 66 26 62 2 76 67 66
```

```
using System;
class RandTest
{
        static void Main()
        {
                Random rnd = new Random(1234);
                for (int i = 0; i < 10; i++)
                {
                        Console.WriteLine(rnd.Next(100));
                }
        }
}
```

```
39 89 31 94 33 94 80 52 64 31
```

<Random Numbers>

- The random numbers used to generate the Secure Sockets Layer (SSL) keys in earlier Netscape were highly predictable, rendering SSL encryption useless.
- ASF Software Texas Hold em Up Poker classic example where after 5 cards you can predict the rest.

<Random Numbers>

Remedy

- Do not call rand instead call a more robust source of data in windows such as CryptGenRandom.
- CryptGenRandom has unpredictability and even value distribution.
  - The current process ID (GetCurrentProcessID)
  - The current thread ID (GetCurrentThreadID)
  - The ticks since boot (GetTickCount)
  - The current time (GetLocalTime)
  - Various high-precision performance counters (QueryPerformanceCounter)
  - MD4 hash of user block

<Cryptography>

- Good to use encryption
- Better using ECB, CBC etc..
- AES, 3DES, etc..
- Where do you store the key?

<Cryptography>

Storing the secret

- Store in a dll
  - MalCode Analyst Pack – Idefense Labs
  - Use strings on the dll.
- How about a highly random password
  - nCipher attaches to the running password and scans the process memory looking for entropy.
  - Areas of high randomness are checked to see if it is the key.
- Definitely not in the application
- Store the key in the registry
  - ACL the registry key Creator/Owner and Administrator to have full control
  - If you are paranoid place an audit so you can see who is attempting to read the data.

<Cryptography>

Storing the secret in .net

- Use data protection API DPAI
  - System.runtime.InteropServices
- Always Demand Appropriate Permissions
- Be thread Aware
- Don’t be afraid to refuse permissions
  - Using system.security.permissions
- Disable tracing and debugging use release versions.
- Generate good random passwords
  - Using system.security.cryptography

<Network based security>

Socket Security

- Sockets heart of any TCP/IP protocol
- Protection is needed to be provided in each layer of the protocol.

<Network based security>

Server Hijacking

- application allows a local user to intercept and manipulate information meant for a server that the local user didn't start themselves.
- When a server starts up, it first creates a socket and binds that socket according to the protocol you want to work with.
- If it's a Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) socket, the socket is bound to a port.
- Less commonly used protocols might have very different addressing schemes.
- A port is represented by an unsigned short (16-bit) integer in C or C++, so it can range from 0 to 65535
- A socket bound to INADDR_ANY loses to a socket bound to a specific IP address

<Network based security>

Accepting Connections

- WINSOCK2 API is more powerful
- Users have the ability to process data coming from a specific client.
- Maximum segment Lifetime specifies the maximum time a packet can exists in the network before it is discarded

<Network based security>

Firewall Friendly applications

- Use one connection to do the job
- Don’t make connection back to the client from the server.
- Don’t try to multiplex connection over another protocol.
- Don’t embed host ip address in the application layer data.
  - WireShark and other network sniffer and analyzers

<Network based security>

Common Attacks

- DoS – ping of death
- CPU Starvation Attack.
- Memory Starvation attacks.
- Resource Starvation attacks
- Network bandwidth Starvation attacks

<Securing web services>

Myths

- Never trust user input.
- Validate Data at both ends.
- Use Hexadecimal Escape Codes
- UTF-8 Variable width Encoding
- HTTP Trust Issues
- Referrer Errors

<Web Security>

ASP.net

- Never trust user input.
- Validate Data at both ends.
- Use Hexadecimal Escape Codes
- UTF-8 Variable width Encoding
- HTTP Trust Issues
- Referrer Errors

<Securing the web>

Attempt to write 100% secure website


WEB COMPANIES
HACKERS

<Buzz in the web>

Fast Flux Bots
Botnets
Malwares
etc…

<I use HTTPS>

QUICK FACT: HTTPS Communication is interruptible.

<What am I going to face>

XSS (Cross site scripting)
SQL Injection
DDoS
etc..

<What am I going to face>

I can get the IPlogs

but what about the case of the user who uses a proxy say TOR, Anonymyser etc..

<Ever thought of Digital Evidence>

VHD in Win 7

Virtual Machines

Virtual PC, VMWARE

<Expect in the net>

No term called copyright

Web Application Security

<Best practices>

- Don’t tell attacker too much details
- World is Open Avoid Ctrl^C Ctrl^V
- Be careful with your keys.
- Use hashes whenever necessary SHA- 1 or at least MD5
- Create a strong security architecture based on your applications flow
- Identify areas of data flow to external objects emphasis more security in these areas
- Do not use untrusted third party API’s
- Choose a better password that password123
  - Most of the server passwords are sadly password123

What to expect Next

- Blue Pill - Virtualization
- More Malwares
- Advanced Packers
- Browsers are not trust worthy
  (MIB - man in the browser)
- Validate source of data.

<Queries>

{ v-jairar@microsoft.com }
{ myidis@gmail.com } | Jairam Ramesh