SAND2016-2370PE

Exceptional service in the national interest

Sandia National Laboratories

```
; Attributes: bp-based frame

; int __cdecl _tmainCRTStartup()
__tmainCRTStartup proc near

var_2C= dword ptr -2Ch
var_28= dword ptr -28h
nested= dword ptr -24h
```

```
ELSE (* Near absolute call *)
    IF OperandSize = 64
        THEN
            tempRIP ← DEST; (* DEST
            IF stack not large enough
                THEN #SS(0); FI;
            Push(RIP);
            RIP ← tempRIP;
    FI;
```

```
6854 7369 6920 2073 756a 7473 6120 6420
6d75 2062 6574 7478 6620 6c69 2c65 6220
7475 4920 6e20 6565 6564 2064 6f73 656d
7420 7865 2074 6f73 7420 6168 2074 2049
6f63 6c75 2064 6174 656b 6120 7320 7263
6565 736e 6f68 2074 666f 7320 6d6f 2065
7573 6570 2072 7773 6565 2074 6f6c 6b6f
6e69 2067 6962 616e 7972 202e 6649 7920
756f 6720 746f 6220 726f 6465 6520 6f6e
6775 2068 6f74 6620 6769 7275 2065 756f
2074 6877 7461 7420 6968 2073 6173 7379
202c 6f79 2075 6873 756f 646c 6520 6d2d
6961 206c 656d 6120 2074 7473 6167 6e69
4065 6173 646e 6169 672e 766f 002e
```

Debugging Demonstration

Sherry Gaines

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Sandia National Laboratories

But first….

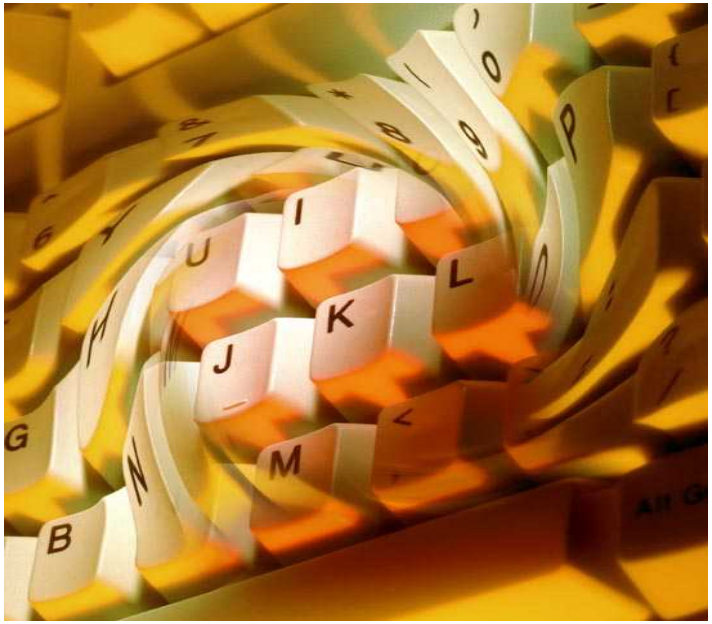
A BIT ABOUT SANDIA AND MY GROUP

Sandia National Laboratories

Advanced Software Engineering

Advanced Software Engineering (ASE) Organization 9520

- Vision
  - We are the recognized provider of secure intelligent technology solutions.
- Mission
  - We develop solutions for National Security Missions, providing technologies that are leveraged by Mission Support.



Sandia National Laboratories

Advanced Software Engineering

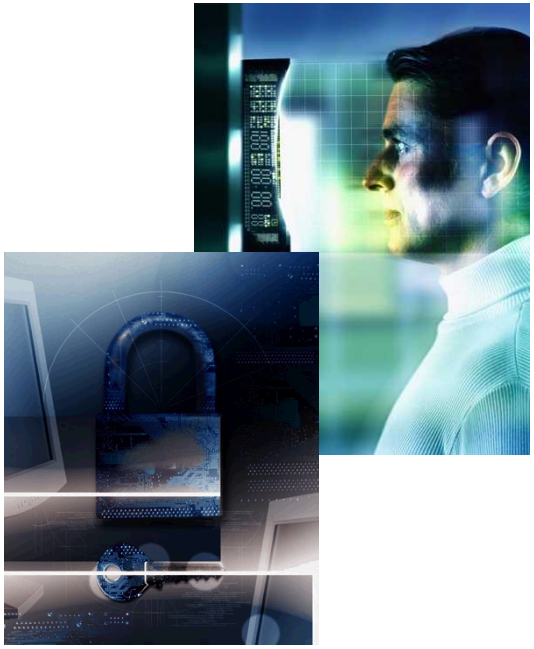
Core Competencies

- Algorithmic Research
- Technology Assessments - Application & Infrastructure Security
- Business Modeling & Analysis, including Requirements Engineering
- Custom Software & Application R&D
- Cyber Modeling & Simulation Architecture Systems
- Cyber Security R&D (encryption, large scale simulations, other…)
- Trusted Systems R&D
- Data Science & Analytics Services (Predictive Analytics)
- Integration Frameworks
- Engineering R&D Solutions into Production Environments
- Satellite Ground System & Data Management Software Systems
- Safeguards & Security Information Systems
- Native Mobile Application Software & Infrastructure (iOS, Surface, Android)
- Emerging Technology Applications

Security Information Systems

Enabling our customers by providing technology solutions that improve their productivity and deliver a substantive return on investment.

- Transforming prototype systems into reliable and maintainable solutions.
- Automating processes to allow for reliable solutions and redirecting personnel from administrative to substantive activities.
- Developing solutions that incorporates the necessary engineering controls and interfaces to ensure the safety and security of the laboratories.

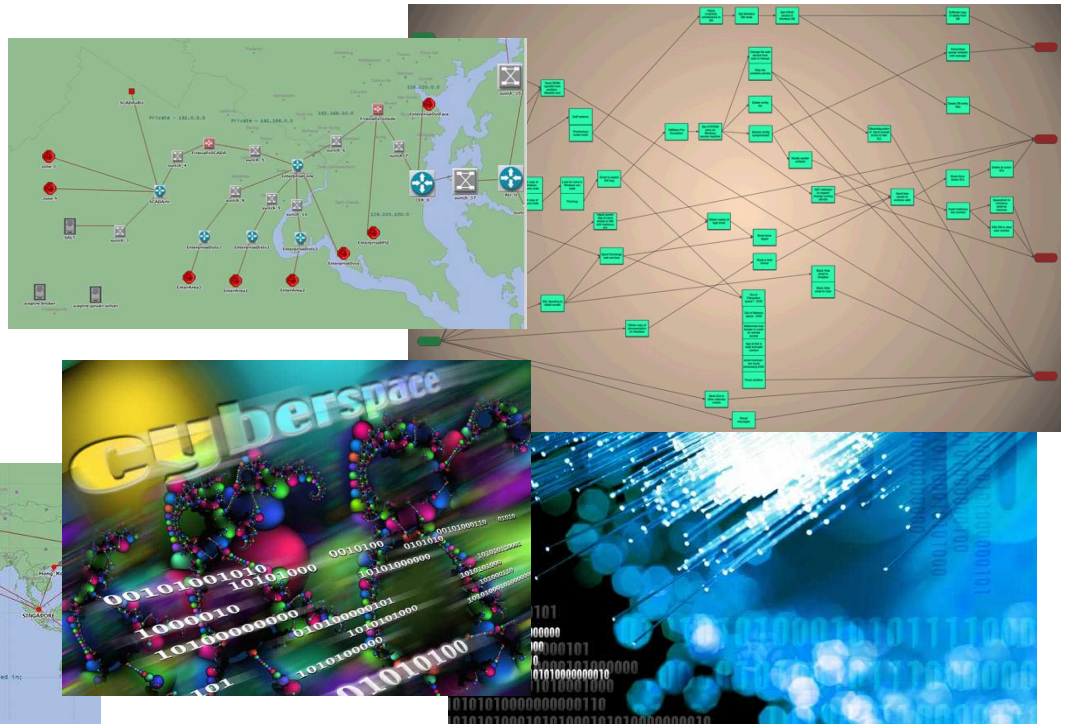


Sandia National Laboratories

Cyber Analysis R&D Solutions

Information security-related research and development as well as providing security expertise in support of the many phases in the software engineering lifecycle.

- Cyber Modeling & Simulation Architecture Security Services
- Large-Scale Emulytics™
- Application Security Services
- Cyber Technical Assessments
- Cyber Security Research



Sandia National Laboratories

Now onto debugging….

WHY DO I CARE?

Usually…. pointers

https://xkcd.com/138/

* 37prime.com

https://xkcd.com/371/

Which of these are pointers?

Sandia National Laboratories

1. char * superSweetChar;
2. char superSweetChars[42];
3. char lonelySweetChar;
4. UINT64 addressOfThing;
5. PUINT64 addressOfThing2;
6. int thisNeatFunction(int, char);
7. int (*thisOtherNeatFunction)(int, char);

What do we use pointers for?

Sandia National Laboratories

- Pointing to dynamically allocated data
- Referencing data within structures
- Typically for pointing to beginning of strings
- Pointing to start of arrays
- ….

Why do our programs crash?

Sandia National Laboratories

- What about using pointers incorrectly makes our programs crash?

- Usually Operating System memory layout and protections!
  - Programs cannot read into each others memory
    - Unless specifically allowed
  - Programs cannot read address 0!
    - NULL == 0
  - User mode programs cannot read kernel mode memory
    - With new hardware features, the reverse is true in some cases as well
  - Page Tables!
    - Control what application can access what memory

Pointer problems cause vulnerabilities!

Sandia National Laboratories

- Buffer overflow
- Heap overflow
- Stack smashing
- Double free
- Use after free
- Data corruption
- Data leakage
- Unintended code flows

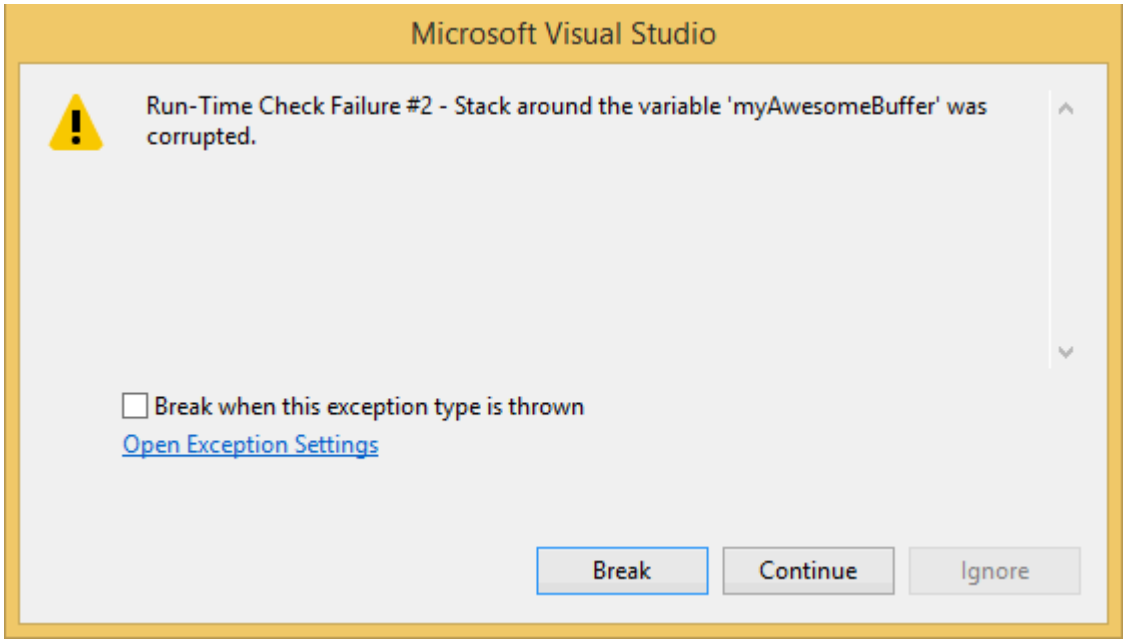
That's scary!

Sandia National Laboratories

- However, there are mitigations to what can be accomplished by an attacker via pointer issues
  - DEP/NX
  - ASLR
  - Memory manager improvements

Example!

- Lets take a simple program that has a buffer overflow vulnerability via the stack
  - DebuggingStackOverflow.cpp



Tips and Tricks

Sandia National Laboratories

- RSP is the stack pointer
- RIP is the instruction pointer
- Stacks grow DOWN (contrary to how they are usually drawn)
- Heaps grow UP (ish)

Questions?

Sandia National Laboratories

```
/*
DebuggingDemo.cpp : Defines the entry point for the console application.

Copyright 2016 Sandia Corporation. Under the terms of Contract
DE-AC04-94AL85000, there is a non-exclusive license for use of this work by or
on behalf of the U.S. Government. Export of this data may require a license
from the United States Government.

For five (5) years from 3/15/2016, the United States Government is granted for
itself and others acting on its behalf a paid-up, nonexclusive, irrevocable
worldwide license in this data to reproduce, prepare derivative works, and
perform publicly and display publicly, by or on behalf of the Government. There
is provision for the possible extension of the term of this license. Subsequent
to that period or any extension granted, the United States Government is
granted for itself and others acting on its behalf a paid-up, nonexclusive,
irrevocable worldwide license in this data to reproduce, prepare derivative
works, distribute copies to the public, perform publicly and display publicly,
and to permit others to do so. The specific term of the license can be
identified by inquiry made to Sandia Corporation or DOE.

NEITHER THE UNITED STATES GOVERNMENT, NOR THE UNITED STATES DEPARTMENT OF
ENERGY, NOR SANDIA CORPORATION, NOR ANY OF THEIR EMPLOYEES, MAKES ANY WARRANTY,
EXPRESS OR IMPLIED, OR ASSUMES ANY LEGAL RESPONSIBILITY FOR THE ACCURACY,
COMPLETENESS, OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT, OR PROCESS
DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS.

Any licensee of "DebuggingDemo v. 1.0" has the obligation and responsibility
to abide by the applicable export control laws, regulations, and general
prohibitions relating to the export of technical data. Failure to obtain an
export control license or other authority from the Government may result in
criminal liability under U.S. laws.
*/

#include "stdafx.h"

#define FUDGE_FACTOR 102
#define ARRAY_LEN 25
#define CHAR_IN_MY_BUFFER 'A'


bool bufferOverflowsAreBad(const char putThisCharInMyBuffer);


int _tmain(int argc, _TCHAR* argv[])
{
bufferOverflowsAreBad(CHAR_IN_MY_BUFFER);
return 0;
}

bool bufferOverflowsAreBad(const char putThisCharInMyBuffer)
{
unsigned int i = 0;
char myAwesomeBuffer[ARRAY_LEN];
for (i = 0; i < ARRAY_LEN + FUDGE_FACTOR; i++)
{
myAwesomeBuffer[i] = putThisCharInMyBuffer;
}
return true;
}
```