Bitdefender

SDL - Buffer Overflows

Reference. Based on a presentation provided by Microsoft in relation to the Security Development Lifecycle – Developer Starter Kit
http://www.microsoft.com/en-us/download/details.aspx?id=4645

Buffer Overflows Overview

Bitdefender

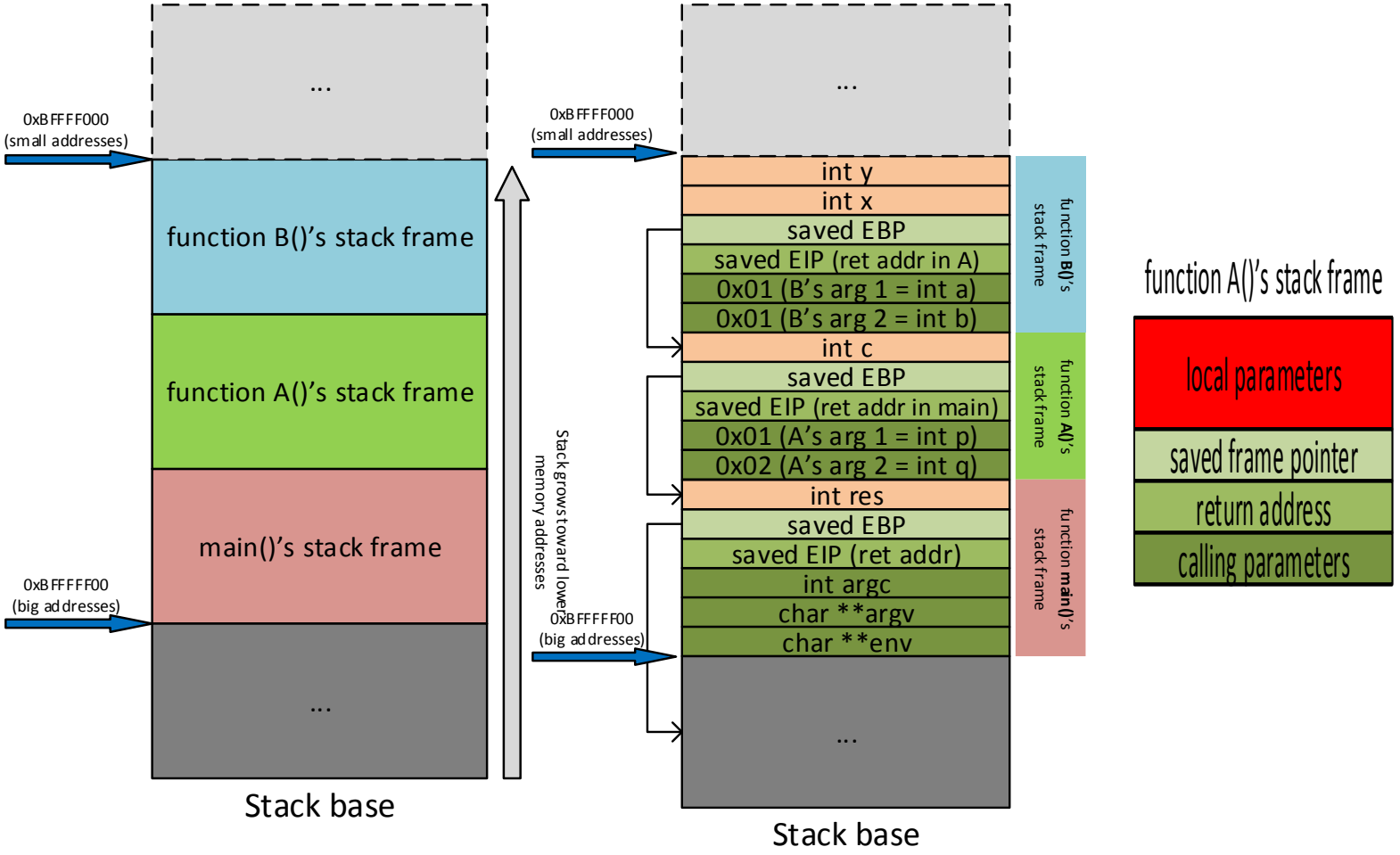
Buffer Overflow: Occurs when data is written into a fixed-length buffer and the size of that data exceeds the capacity of the receiving buffer

- **Primary Risks:** Corrupt data, crash programs and control execution flow
- Common in native applications (C/C++)
  - Rare, but still possible in managed code (.NET, Java)
- Cause is failing to validate input
- Can occur on stacks and heaps

Review of Application Stack Frames (detailed)

```
int function_B(int a, int b)
{
  int x, y;

  x = a * a;
  y = b * b;

  return (x + y);
}

int function_A(int p, int q)
{
  int c;

  c = p * q * function_B(p, p);

  return c;
}

int main(int argc, char **argv, char **env)
{
  int res;

  res = function_A(1, 2);

  return res;
}
```

```
    push EBP
    mov EBP, ESP // local variable
    sub ESP, 48h
    ...

    push EBP
    mov EBP, ESP // local variables
    sub ESP, 44h
    ...

    push 1
    push 1
    call function_B

    push 2
    push 1
    call function_A
    ...
```



Reference. M. Down et al., *The Art of Software Security Assessment*, Addison Wesley, 2012, pg. 175

Stack-Based Buffer Overflows

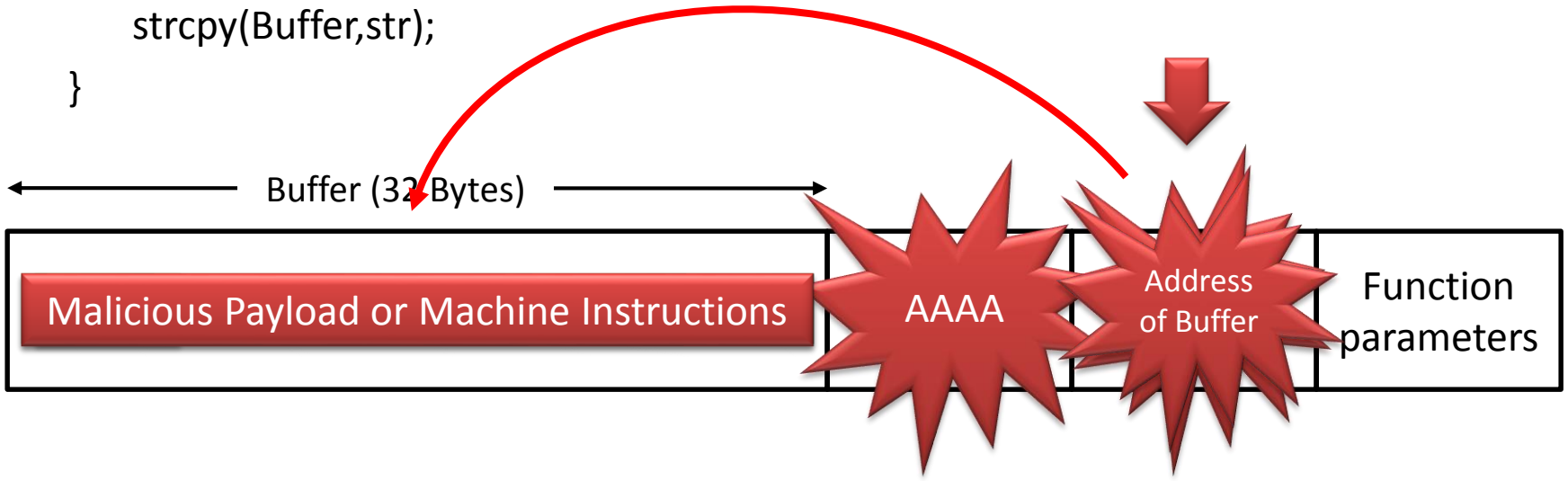
Bitdefender

Primary Risk: Ability to overwrite control structures

```
/* UNSAFE Function */
void UnsafeFunction(char * str)
{
    char Buffer[32];

    /* Copy str into Buffer */
    strcpy(Buffer,str);
}
```

SAMPLE INPUTS (STR VALUES):

1. "Kevin"
2. "A" repeated 40 times

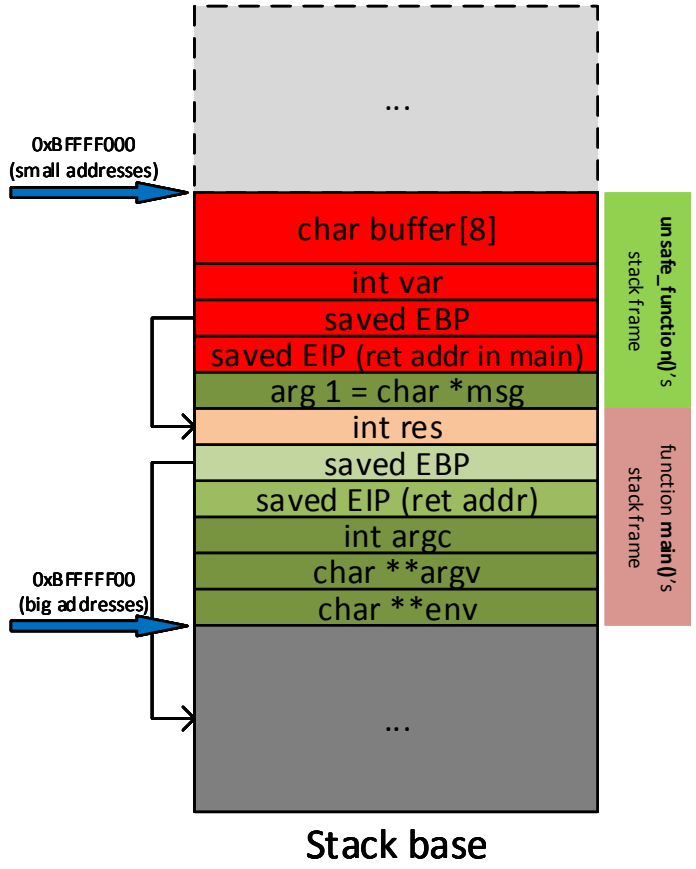


Stack-Based Buffer Overflows (details)

Bitdefender

Primary Risk: Ability to overwrite control structures

```
int unsafe_function(char *msg)
{
  int var;           // local variables
  char buffer[8];

  var = 10;
  strcpy(buffer, msg);

  return var;
}

int main(int argc, char **argv, char **env)
{
  int res;

  /* Buffer overflow for "strlen(argv[1]) >= 8"
  res = unsafe_function(argv[1]);

  return res;
}
```



Heap-Based Buffer Overflows

Bitdefender

Primary Risk: Ability to write arbitrary 4 byte DWORD anywhere in memory (return address, pointers, etc.)

```
/* UNSAFE Function */
void UnsafeFunction(char * str)
{
    /* Allocate 32 bytes heap space */
    char * Buffer = (char *)malloc(32);

    /* Copy str into Buffer */
    strcpy(Buffer,str);
}
```

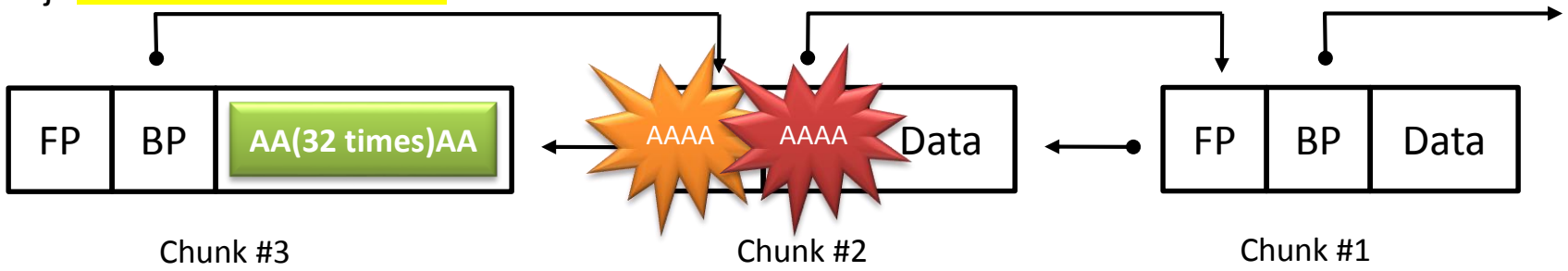
Pseudo-code For Chunk Freeing:

NextChunk = AAAA

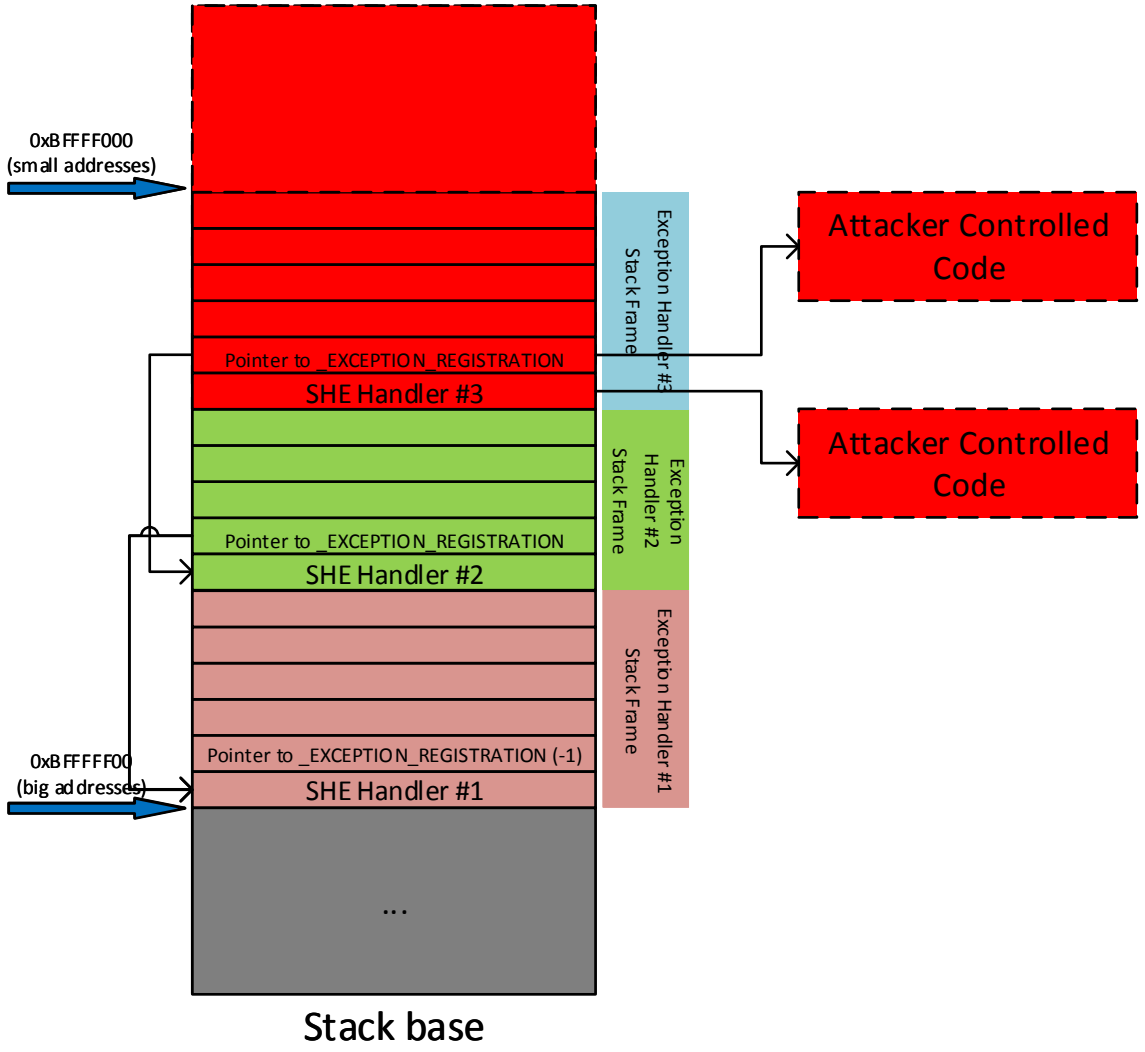
PreviousChunk = AAAA

AAAA ->BP = AAAA

PreviousChunk->FP = NextChunk



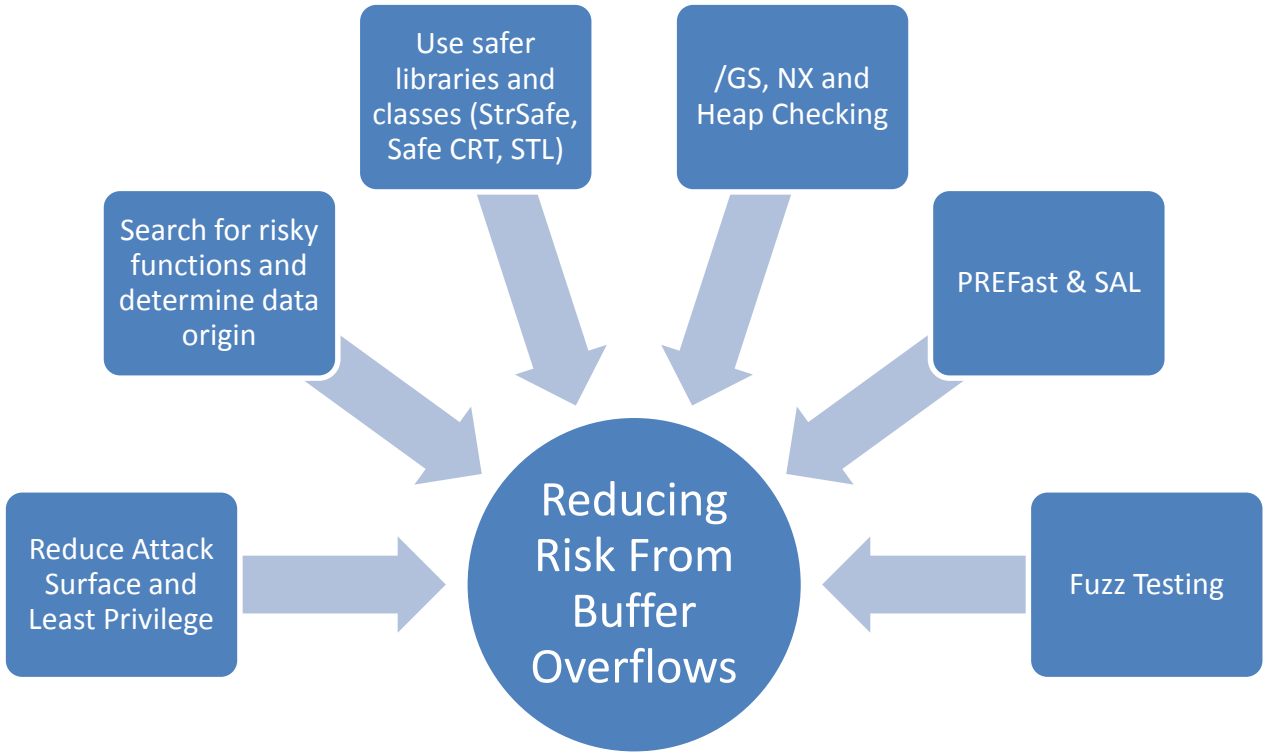
Structured Exception Handling (SEH)



Reference. M. Down et al., *The Art of Software Security Assessment*, Addison Wesley, 2012, pg. 179-180

Reducing Exposure to Buffer Overflows with the Microsoft SDL

Bitdefender



- *Presentation content is available for all of these topics*

Platform Protection From Buffer Overflows

Bitdefender

- Modern day operating systems and processors have built-in buffer overflow protection
  - Address Space Layout Randomization (ASLR)
  - Data Execution Protection (DEP)
- However none of these are “silver bullets”
  - Denial of Service (DoS) attacks usually not prevented
  - More subtle attacks could still be performed
  - Developers still need to follow security best practices
  - Developers should always apply the Microsoft SDL

CWE Buffer-Overflow Related

- **CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer**
- **CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')**
  - Rank 3 in the Top 25
- **CWE-121: Stack-based Buffer Overflow**
- **CWE-122: Heap-based Buffer Overflow**
- **CWE-124: Buffer Underwrite ('Buffer Underflow')**
- **CWE-125: Out-of-bounds Read**
- **CWE-131: Incorrect Calculation of Buffer Size (!)**
  - Rank 20 in the Top 25
- **CWE-170: Improper Null Termination**
- **CWE-190: Integer Overflow (!)**
  - Rank 24 in the Top 25
- **CWE-193: Off-by-one Error**
- **CWE-805: Buffer Access with Incorrect Length Value**
- **…**

Reference. CWE = Common Weakness Enumeration (http://cwe.mitre.org)

Real-Life Examples

- First well-known Internet worm: *Morris finger worm* (1988)
- Common Vulnerabilities and Exposures (https://cve.mitre.org/find/index.html)
  - Searching string “buffer overflow” → “About 639 results” (actually few thousands)
- Vulnerability Notes Database (https://www.kb.cert.org/vuls/)
  - Searching string “buffer overflow” → “About 240 results”
- Examples
  - CVE-2015-0235 - GHOST: **glibc gethostbyname** buffer overflow
  - CVE-2014-0001 - Buffer overflow in client/mysql.cc in **Oracle MySQL** and MariaDB before 5.5.35
  - CVE-2014-0182 - Heap-based buffer overflow in the virtio_load function in hw/virtio/virtio.c in **QEMU** before 1.7.2
  - CVE-2014-0498 - Stack-based buffer overflow in **Adobe Flash Player** before 11.7.700.269
  - CVE-2014-0513 - Stack-based buffer overflow in **Adobe Illustrator** CS6 before 16.0.5
  - CVE-2014-8271 - **Tianocore UEFI implementation** reclaim function vulnerable to buffer overflow
  - CVE-2013-0002 - Buffer overflow in the **Windows Forms** (aka WinForms) component in Microsoft .NET Framework
  - CVE-2005-3267 - Integer overflow in **Skype** client … leads to a resultant heap-based buffer overflow
  - …

Conclusions

Bitdefender

- Buffer Overflow
  - classical, well known, still present (“oldie but goldie”)
  - due to
    - the usage of unsafe function and non-validated user input
    - logic (calculation) errors
- Recommendations for Code Developers
  - Do not use unsafe (string) functions
  - Use the right compiler/linker options
  - Check allocation size calculations
    - Do make size checking
    - Take care of automatic type casting and possible integer overflows
      - use **size_t** (when possible) for allocation size variables
      - take care at casts from **signed to unsigned**
      - ….
- Recommendations for Code Reviewers
  - Check for user input and trace it through the application
  - Check for unsafe functions
  - Check for allocation size calculations