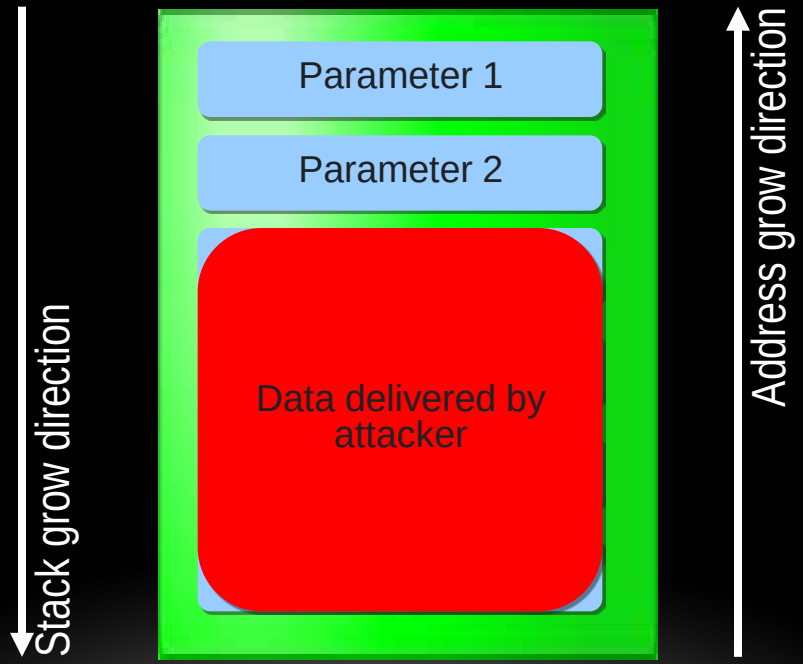
DESKTOP APPLICATIONS VULNERABILITIES

Grzegorz Niemirowski

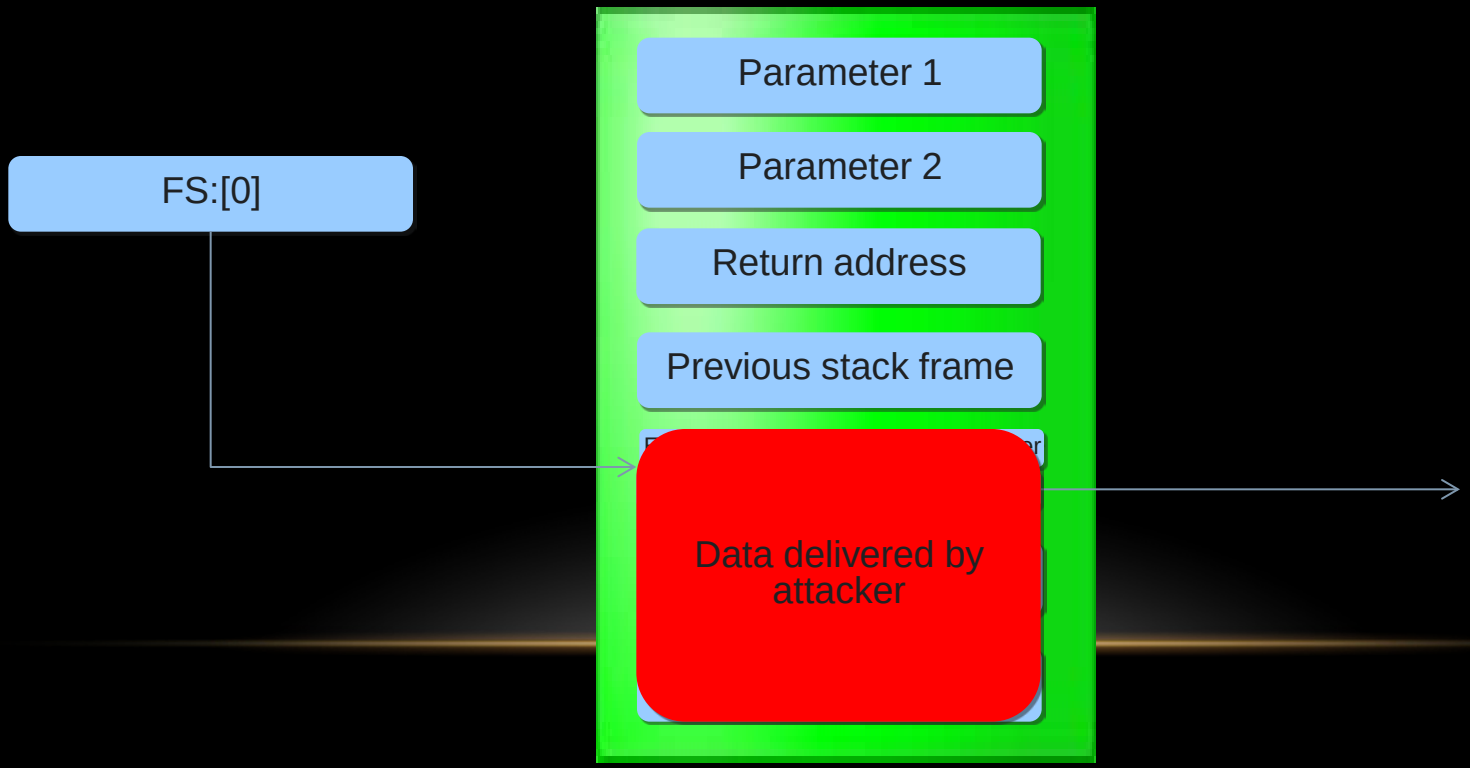
BUFFER OVERFLOW ON STACK

- Classic, one of the oldest techniques of attack
- The goal is to overwrite return address from function and to take control over EIP register



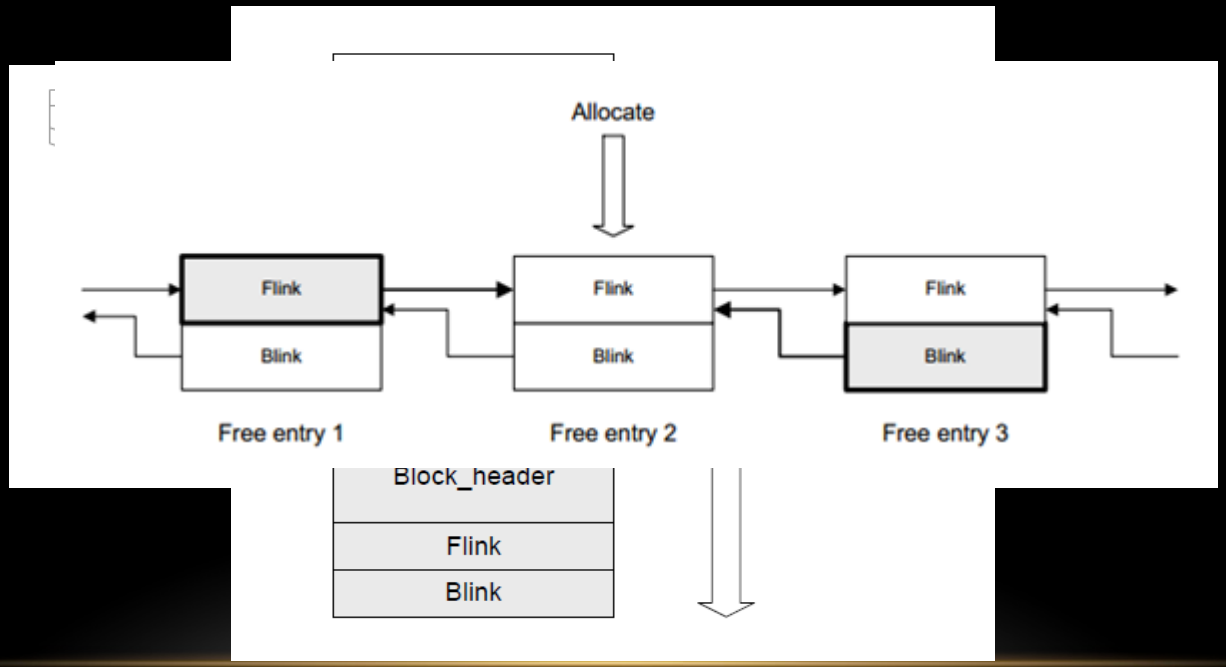
STRUCTURED EXCEPTION HANDLING

- Exception handling supported by operating system
- Makes use of structures stored on stack
- Attacker can overwrite exception handler pointer



BUFFER OVERFLOW ON HEAP

- Attacker can use buffer stored on heap to overwrite metadata used to manage the heap.
- In Windows XP the Blink and Flink fields were targetted



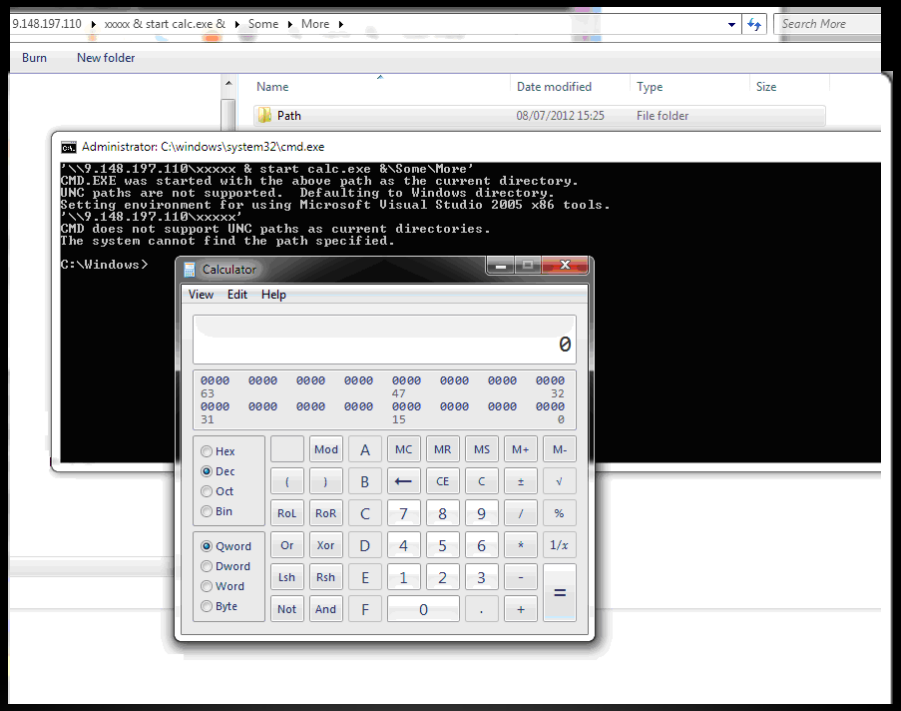
INTEGER OVERFLOW

- Exploits limited capacity of of variables and bugs in handling signed and unsigned vartibales.
- Example: CVE-2010-3970, vulnerability in function ConvertDIBSECTION located in shimgvw.dll

```
mov [ebp+bmi.bmiHeader.biClrUsed], eax
; [ . . . ]
mov ecx, eax
cmp ecx, 100h
jg error
add esi, 28h
lea edi, [ebp+bmi.bmiColors]
rep movsd;   move ds:esi -> ds:edi, repeated ecx times
```

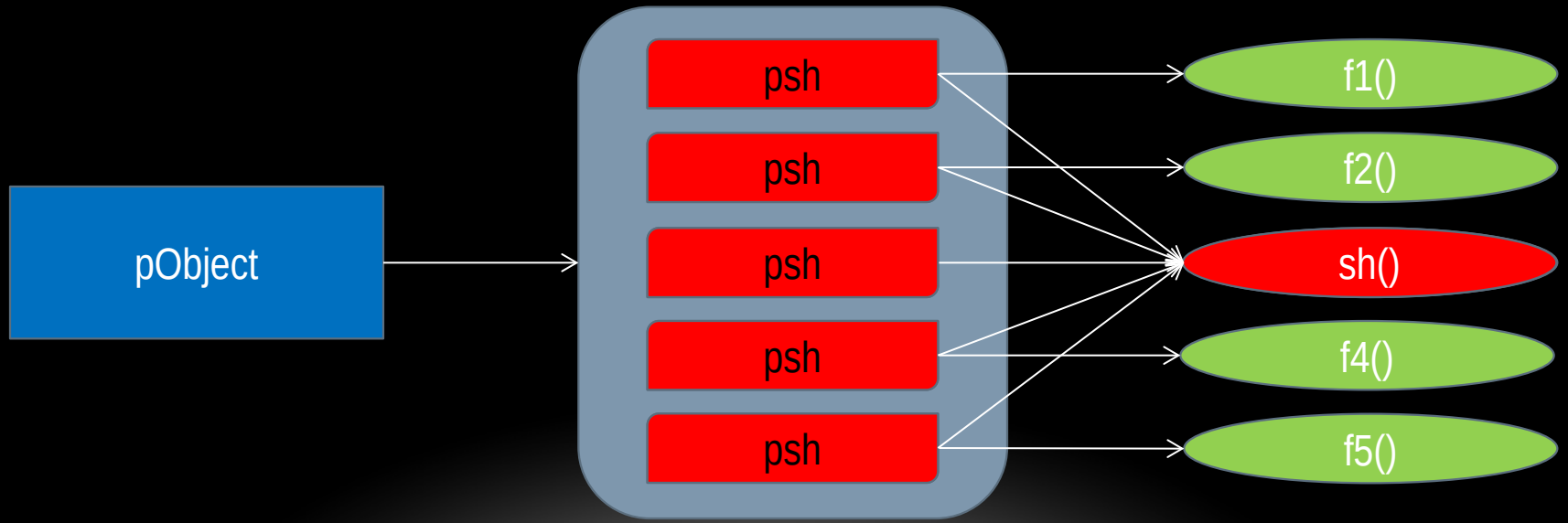
PATH TRAVERSAL

- Improper path validation allows unauthorized access to directories and shell command execution



USE AFTER FREE

- Frequent vulnerability in web browsers
- Call to memory which was freed and then overwriteen by an attacker
- Overwrite is usually done using heap spray



LOADING UNTRUSTED LIBRARIES

- Affects Windows applications written without respecting rules of writreing secure code
- Library loaded from working directory, which may be controlled by attacker
- Microsoft has published 27 security bulletins covering own application affected by this vulnerability (as of end of 2012)

TIME OF CHECK TO TIME OF USE

- Conditions may by checked correctly, but they can change just before action is taken
- Attack requires synchronization with attacked application

```
if (access("file", W_OK) != 0) {
        exit(1);
}
fd = open("file", O_WRONLY);
write(fd, buffer, sizeof(buffer));
```

NULL POINTER DEREFERENCE

- Attack leverages the fact that kernel address space is mapped in application address space
- By mapping memory page at null address it is possible to use kernel bugs to elevate privileges

IMPROPER SOFTWARE CONFIGURATION AND PRIVILEGES VERIFICATION

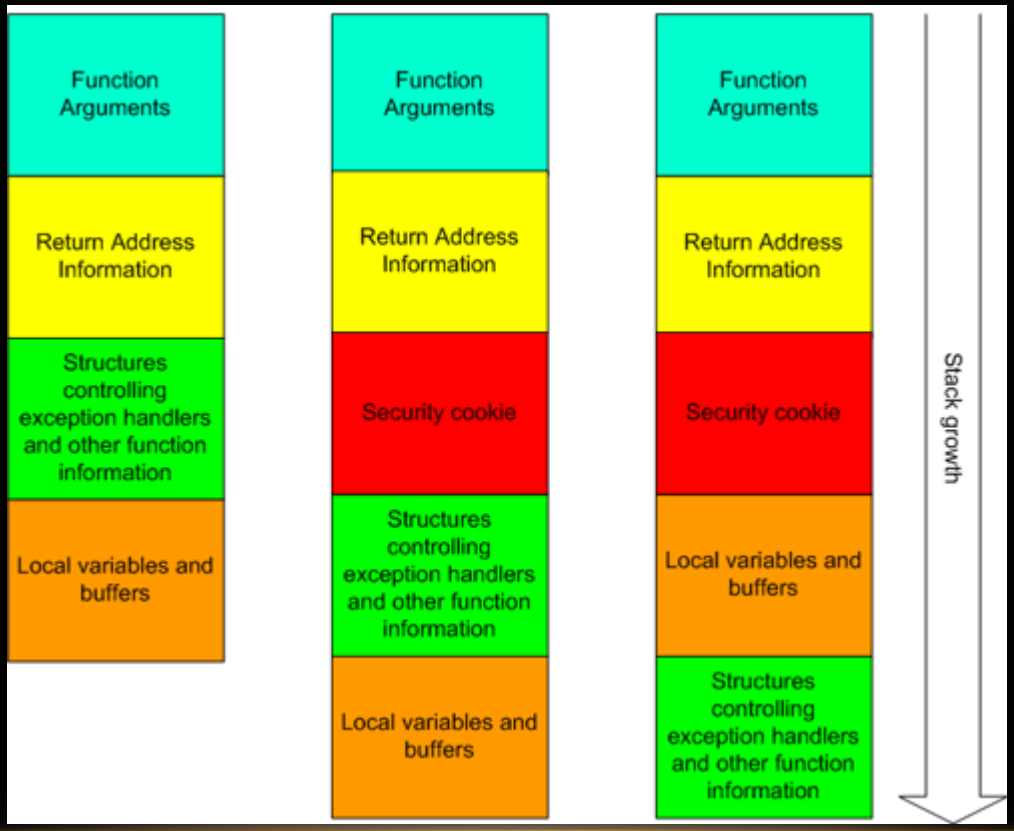
- Bugs in firewall rules
- Too high privileges for account
- Service/daemon running with too high privileges
- Lack of proper protection of sesnsitive data, e.g. private keys
- Buggy algorithm for privileges verification
- Design flaws

SHAREMOUSE

- Sends UDP broadcasts with information about computer
- Mouse and keyboard data can be encrypted, but…
  - The password is broadcasted
  - The encryption is just XOR and base64
- File transfer and clipboard sharing not encrypted at all
- Files and clipboard data transfered without any authorization, anyone can download arbitrary file or replace clipboard data. Even if password protection is enabled.
- There is possibility of DoS attack by sending random data

STACK COOKIES

- Additional value places on stack, verified in function's epilogue
- In case of detected modification an exception is risen

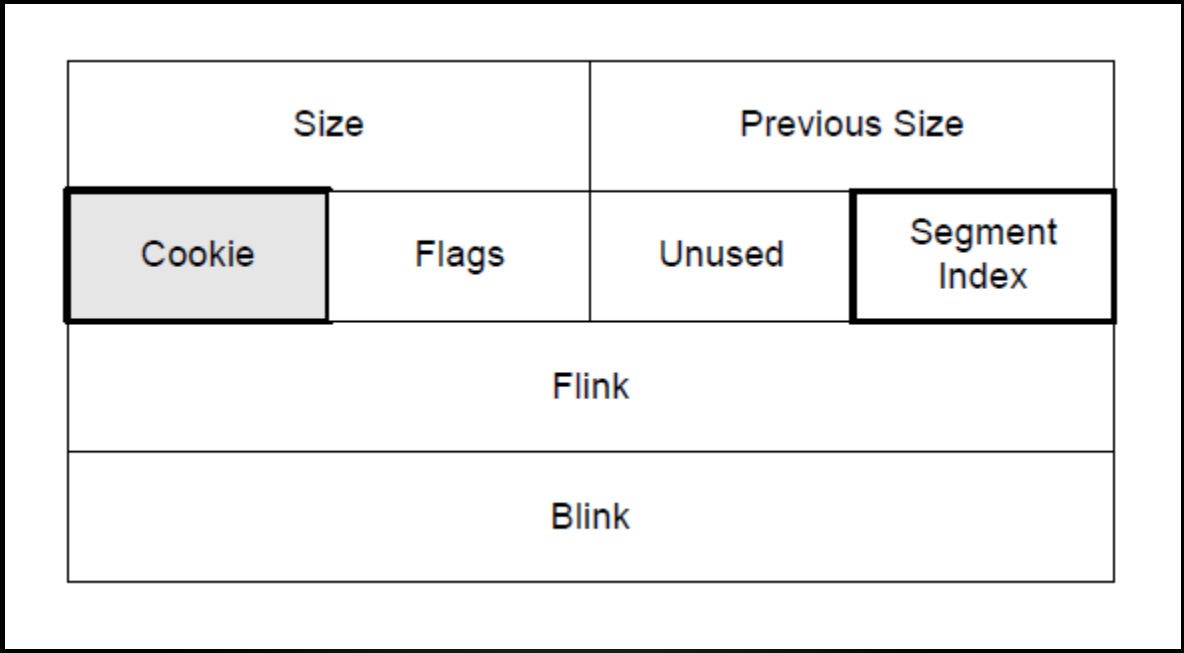


THE /GS SWITCH IN VISUAL STUDIO

- Variables reordering (array variables at higher addresses)
- Arguments shadowing
- Checking null-termination in strings
- Cookie modification exception handled by operating system procedure

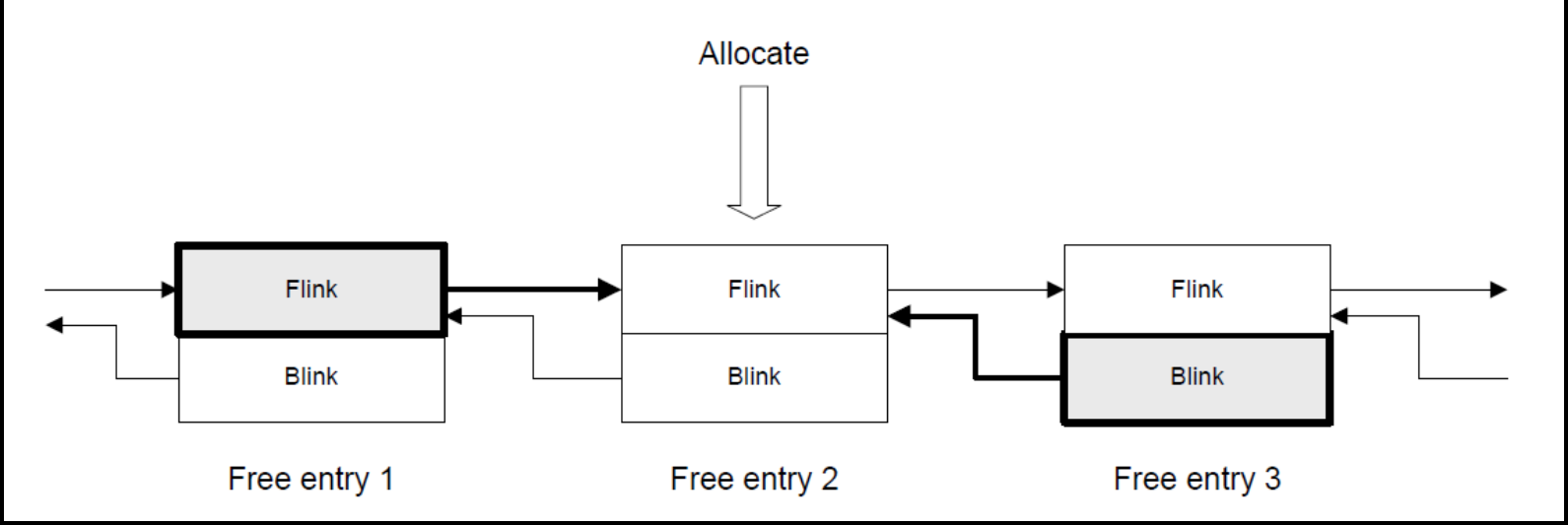
HEAP PROTECTION

- Cookies just like for stack



HEAP PROTECTION

- Safe unlinking



HEAP PROTECTION

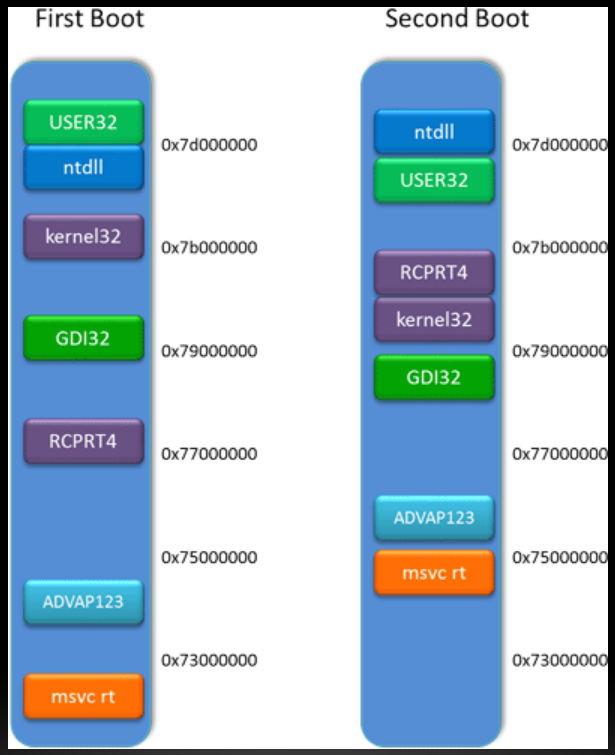
- The FreeList has been removed, and the lookaside list has been replaced by Low Fragmentation Heap (LFH)
- Introduced block headers encryption by XORing with random value
- Cookie is checked more frequenlty and used for integrity checking of other fields
- Heap base address is random, ASLR privides 5-bit entropy
- Process can be immediately terminated if heap structures modification is detected
- Allocation algorithms have been modified to behave less deterministically.
- In Windows 8 blocks are allocated in random order. Attacker can’t assume that next block is allockated just after the previous one.

DATA EXECUTION PREVENTION

- Hardware function, disables executability in given memory pages.
- Attempt to execute code from protected pages generates access violation exception.
- DEP limits exploitation possibilities by disabling executability for places in memory where shellcode is likely to be placed
- To bypass DEP malware authors use technique called Return Oriented Programming.

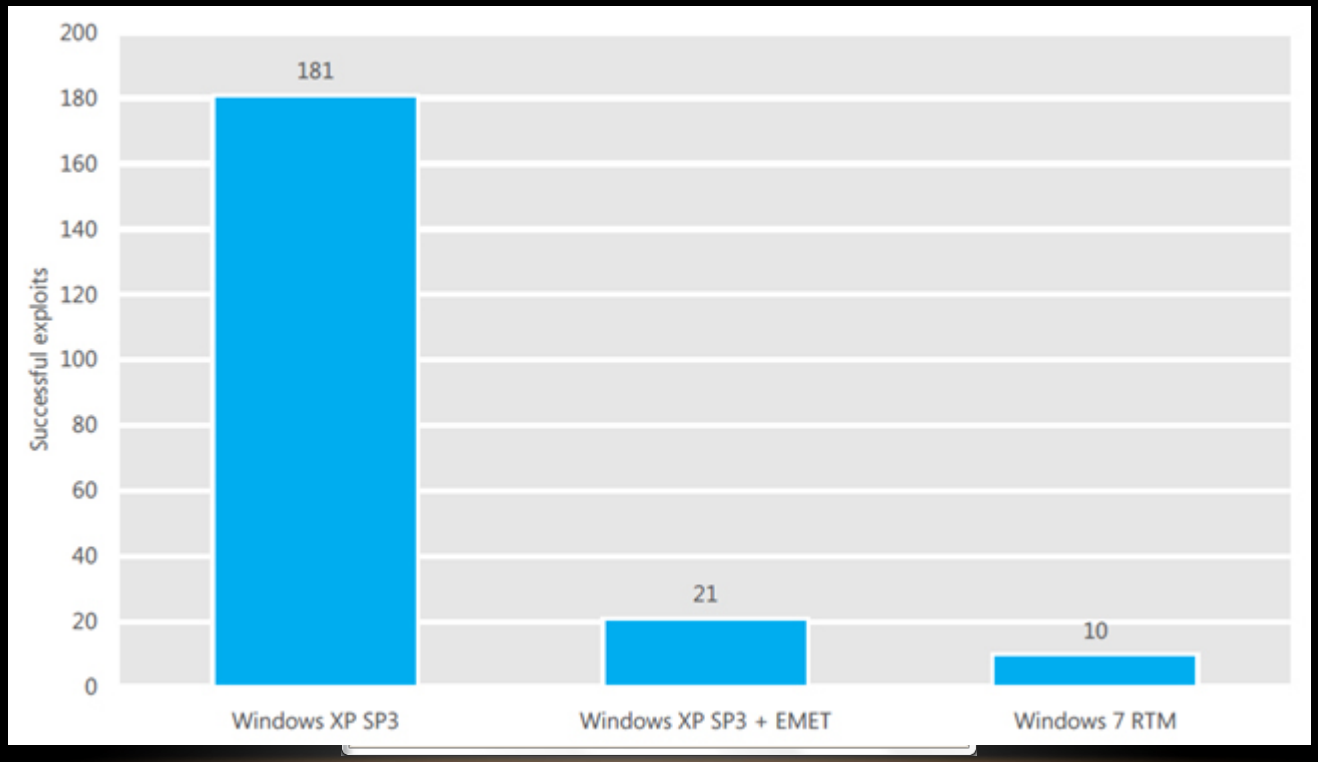
ADDRESS SPACE LAYOUT RANDOMIZATION

- Introduces randomness of location of such structures as stack, heap and DLL base addresses



ENHANCED MITIGATION EXPERIENCE TOOLKIT

- One of tools which can limit exploitation possibilities, does not modify executable files.
- Makes use of shims, mechanism created to solve problems with application compatiblity



FEATURES OF EMET 3.0

- SEHOP
- DEP
- heap spray prevention
- Allocating null memory page (protects against exploiting null pointer dereference)
- Mandatory ASLR
- EAF
- Bottom-up randomizing

NEW IN EMET 3.5 (BETA)

- Disables DLL loading from SMB shares
- Disables marking stack space as executable
- When some critical functions API functions are called, often used by exploits, there is a check whether the call has been made by CALL or RET instruction. In case of RET it is assumed the the application is being attacked and EMET terminates it.
- Additionally it is checked whether those functions are called using ROP gadgets. To detect such condition EMET emulates execution of instructions pointed by return address from the function,
- Detects stack pivoting – tampering ESP register to point to shellcode.

THANKS!