Exploit Writing Technique

2012. 12.01

서만기 연구원

AhnLab

Contents

AhnLab

01 Buffer Overflow

AhnLab

Buffer overflow 개요

v Buffer overflow란?

- u 임시저장 공간보다 더 큰 사이즈의 데이터를 저장 하게 되면서 할당된 buffer의 공간을 넘어서 다른 주요 데이터를 덮어 쓰게 되는 것

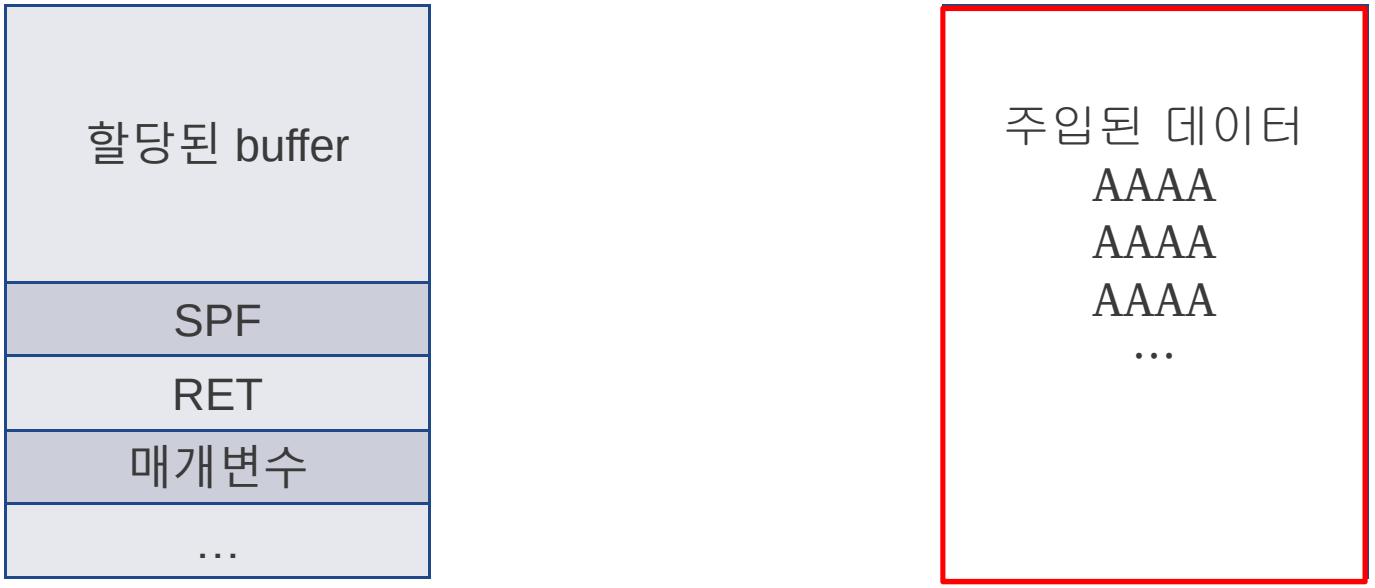
- u 주요 Buffer overflow의 종류
  - § Integer overflow
  - § Stack-based overflow
  - § Heap-based overflow

Stack-based overflow 개요

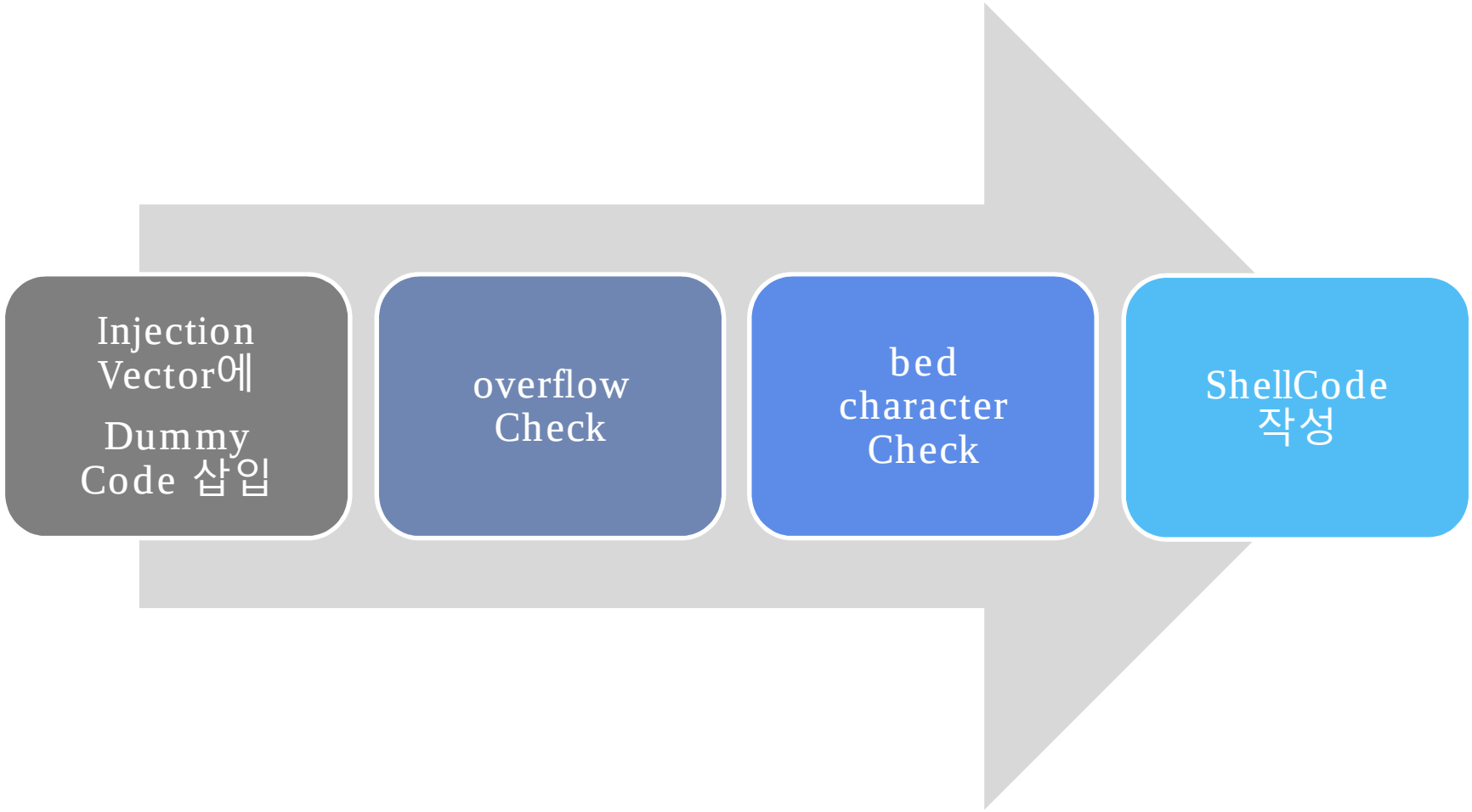
v Stack-based overflow

- u 할당된 buffer보다 더 많은 데이터가 주입되면서 stack에 저장된 주요 데이터 (SPF, RET, 각종 변수)를 덮어 쓰게 되면서, 공격자의 의도 대로 프로그램의 흐름을 제어 할 수 있게 되는 취약점
- u 주로 문자열 데이터를 처리하는 과정 중에 발생

v Stack-based overflow 구조



Exploit Writing 과정



Memory Protection에 따른 Exploiting Technique

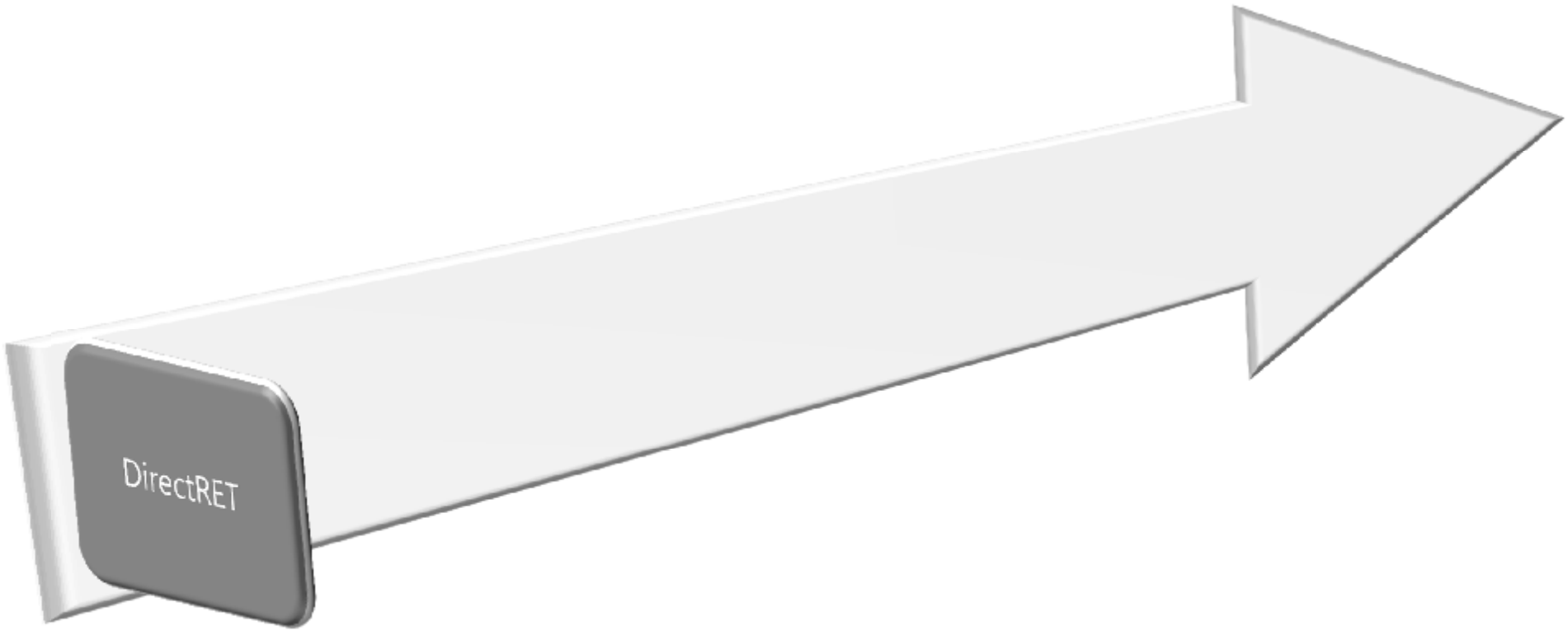
AhnLab

Stack-based overflow 대응책(cont)

v Windows Memory Protection

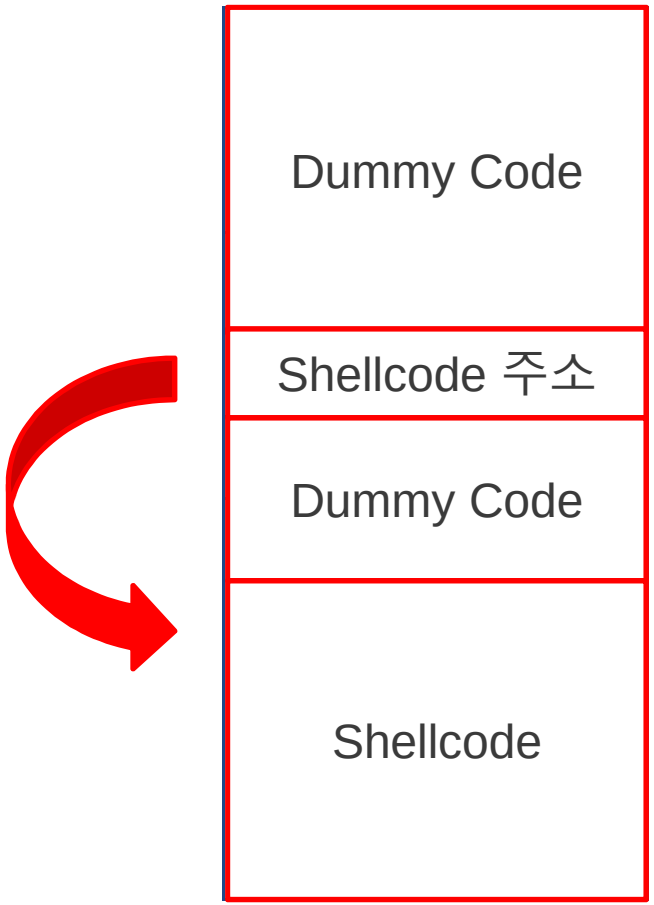
		XP SP2, SP3	2003 SP1 SP2	Vista SP0	Vista SP1	2008 SP0
GS	stack cookies	yes	yes	yes	yes	yes
	variable reordering	yes	yes	yes	yes	yes
SafeSEH	SEH handler validation	yes	yes	yes	yes	yes
	SEH chain validation	no	no	no	yes	yes
Heap protection	safe unlinking	yes	yes	yes	yes	yes
	safe lookaside lists	no	no	yes	yes	yes
	heap metadata cookies	yes	yes	yes	yes	yes
	heap metadata encryption	no	no	yes	yes	yes
DEP	NX support	yes	yes	yes	yes	yes
	permanent DEP	no	no	no	yes	yes
	OptOut mode by default	no	yes	no	no	yes
ASLR	PEB, TEB	yes	yes	yes	yes	yes
	heap	no	no	yes	yes	yes
	stack cookies	no	no	yes	yes	yes
	images	no	no	yes	yes	yes

Exploit Writing 과정

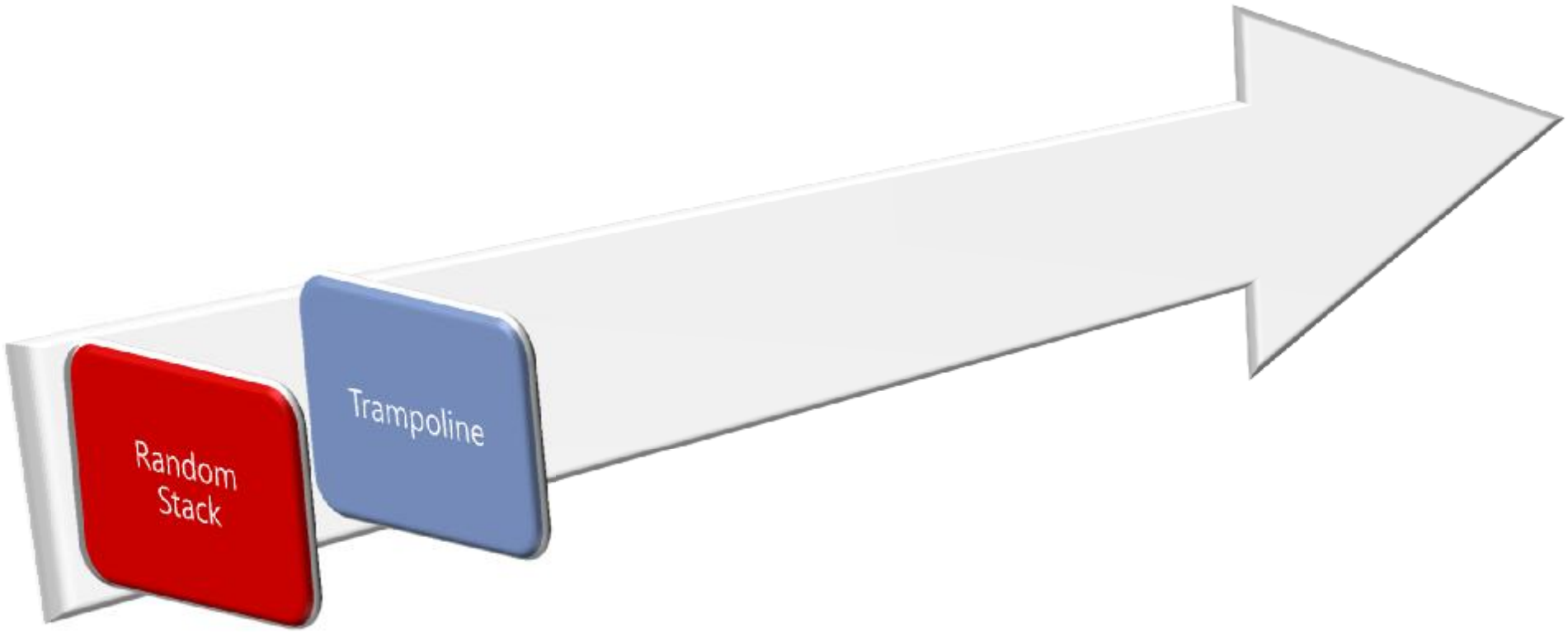


v DirectRET

- u overflow를 통해서 RET(리턴주소)를 조작하여 shellcode를 동작시킴



Exploit Writing 과정



Trampoline 개요

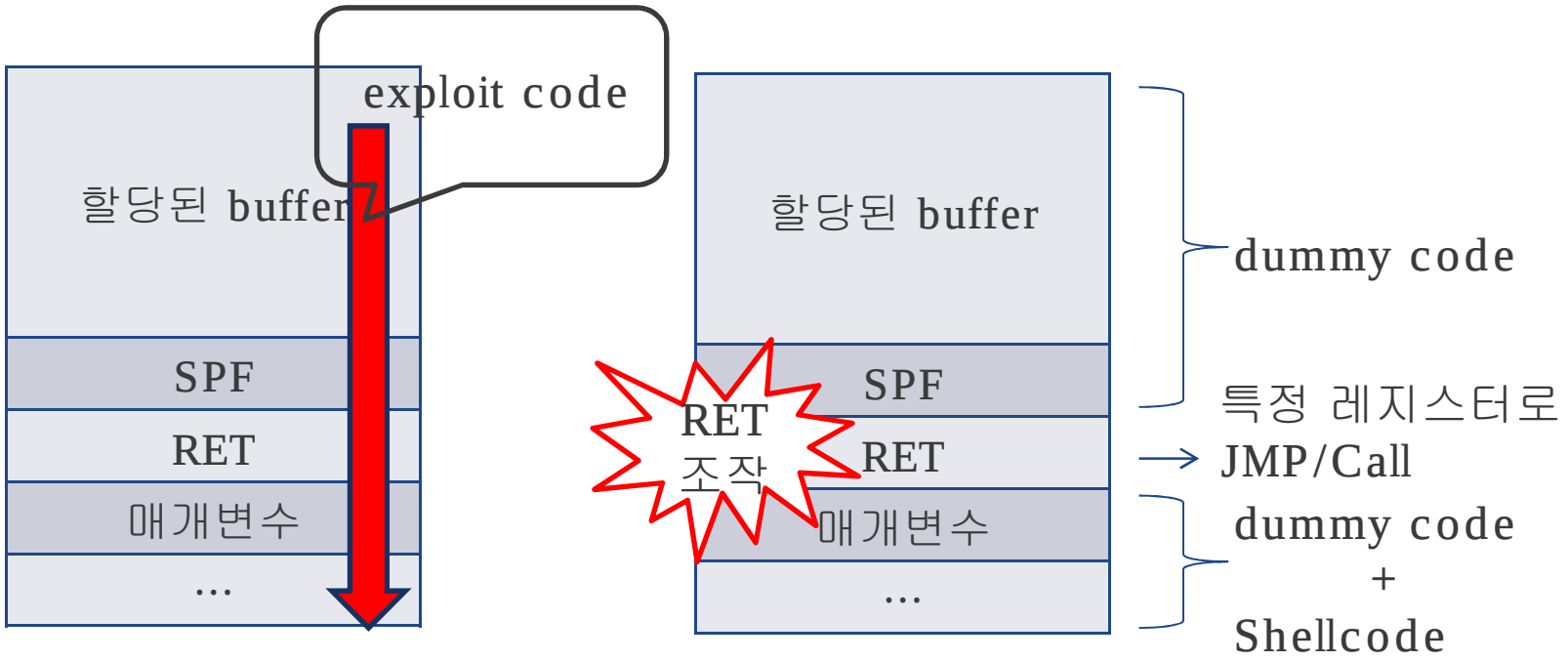
v shellcode 실행의 문제점

- u Shellcode의 주소값을 예측하기 힘듬
- u 주소값을 확인 하더라도 stack주소가 변경될 경우 일회성 공격으로 끝남

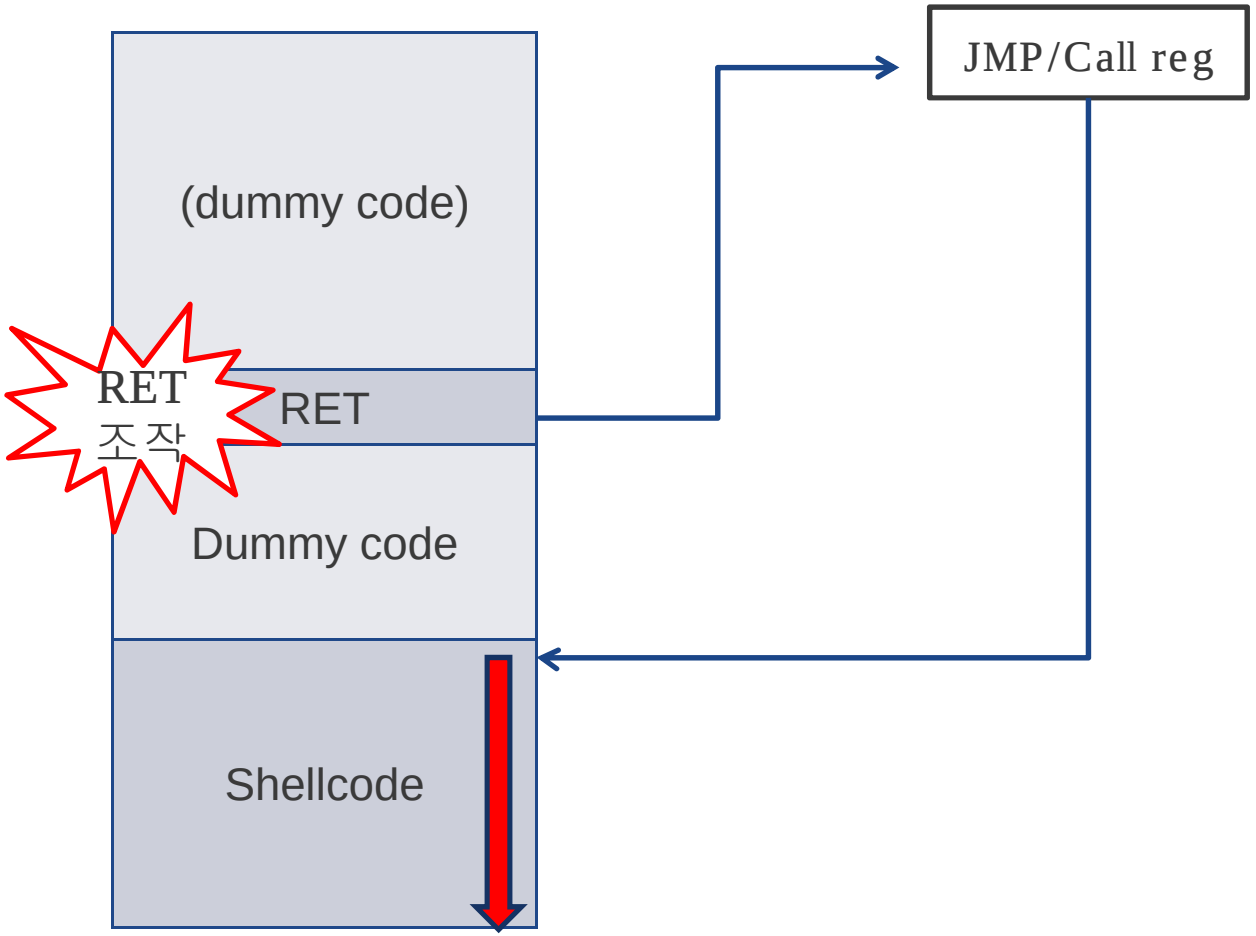
v Trampoline 기법

- u RET의 값을 shellcode가 저장되는 값으로 하드코딩 하지 않고 주입된 데이터의 특정 위치를 레지스터를 활용하여 shellcode를 동작시키는 기법
- u 가상메모리 주소값이 바뀌더라도 레지스터가 가리키는 지점의 offset값은 변화가 없다는 점을 활용

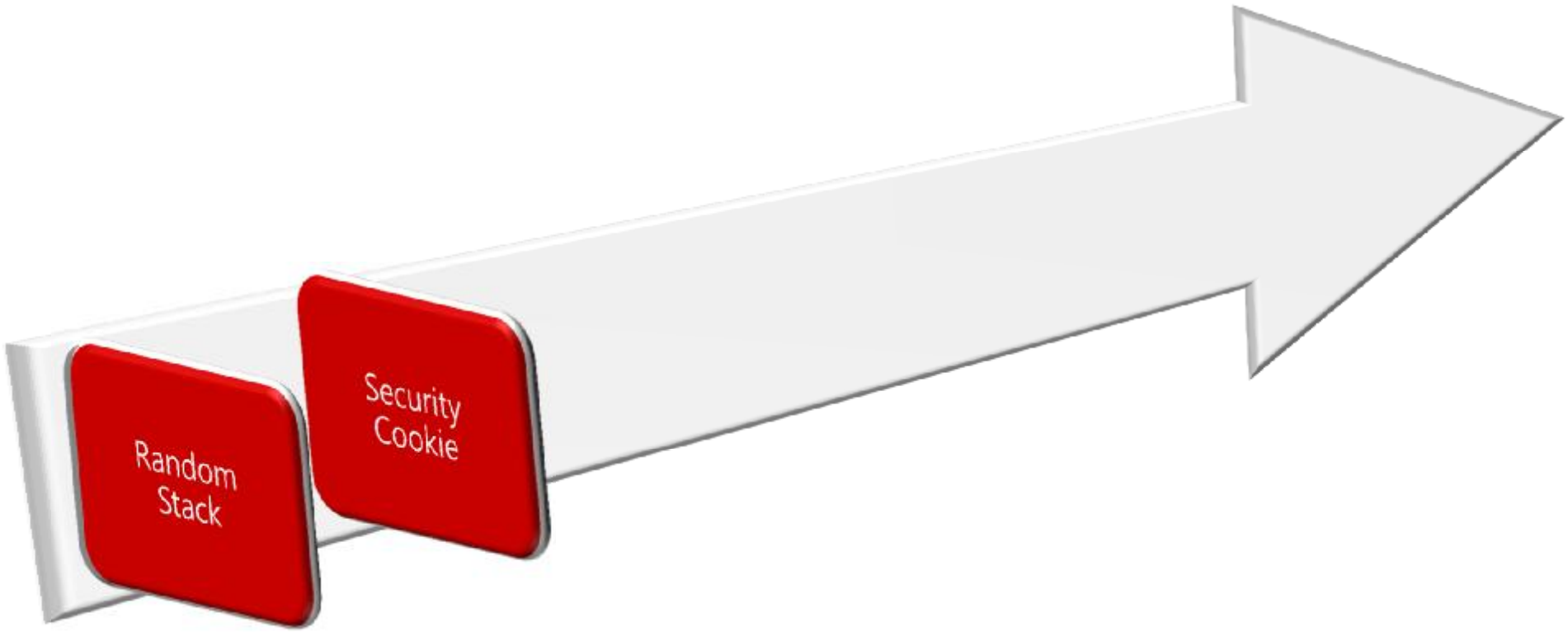
Trampoline 진행 과정(cont)



Trampoline 진행 과정(cont)



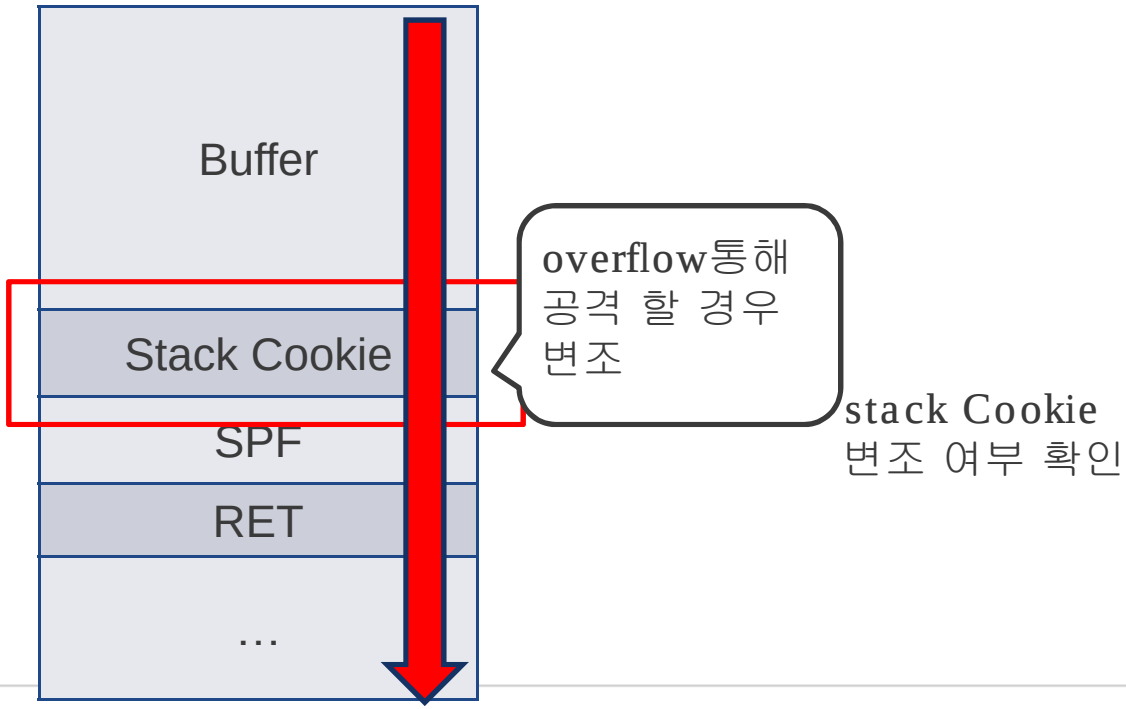
Exploit Writing 과정



Stack Cookies 개요

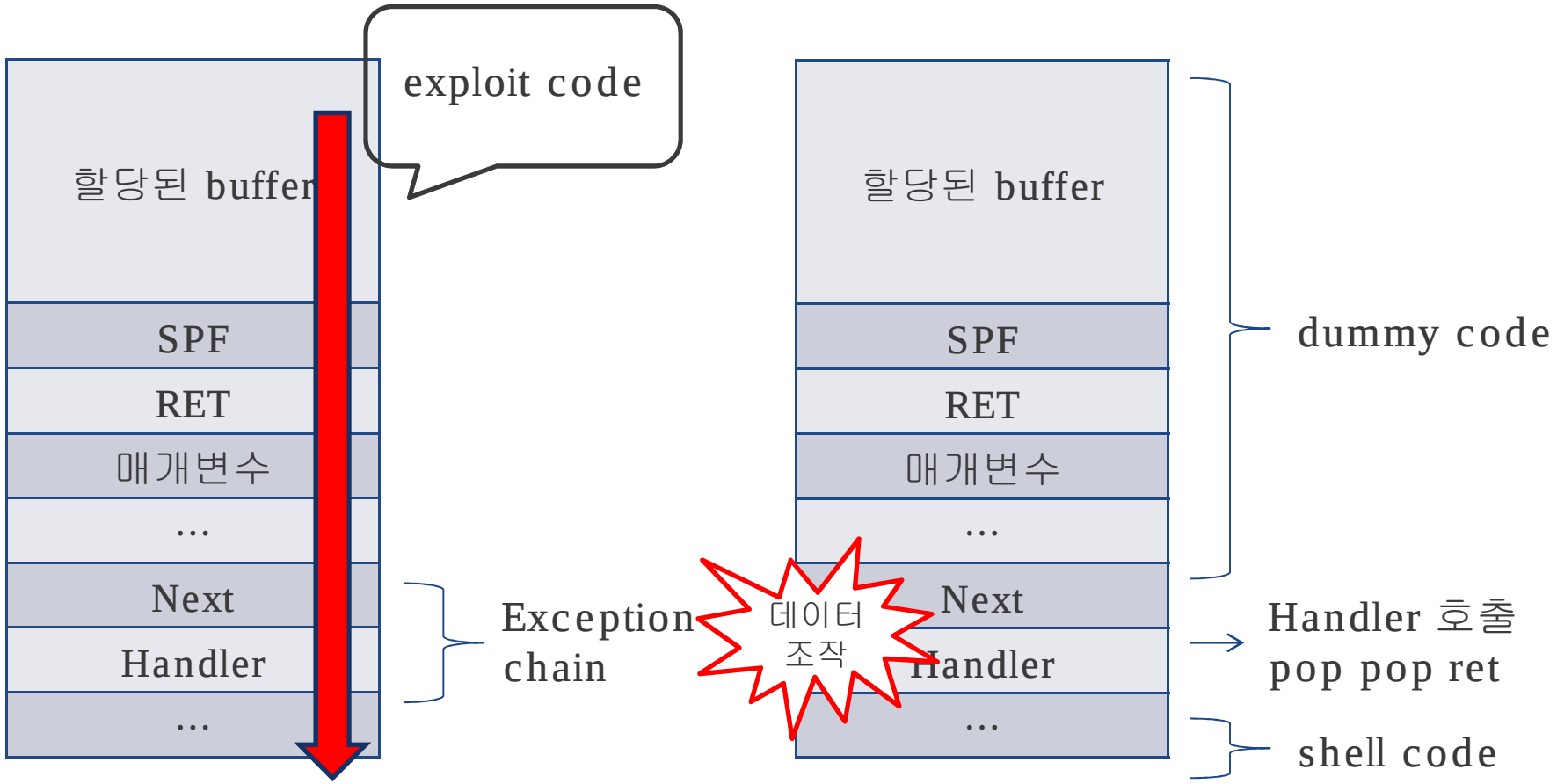
v Stack Cookies?

- u Stack based overflow로 인한 공격을 방지하기 위한 4바이트 데이터
- u 프롤로그 과정에서 값이 계산되어 stack에 저장되었다가 에필로그 과정에서 변조 되었는지 검증
- u 문자열을 저장하기 위한 buffer가 할당되었을 때 활용된다.

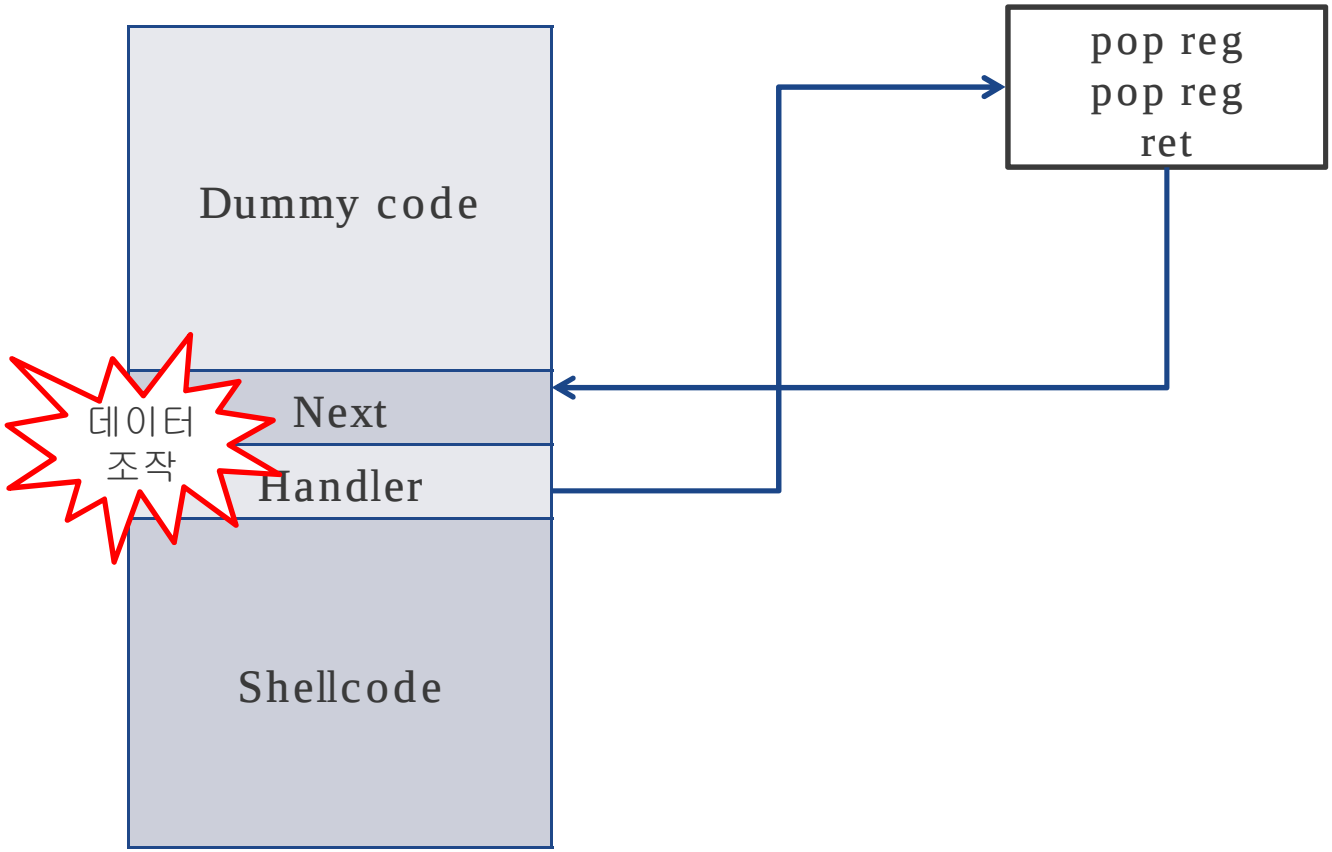


Exploit Writing 과정

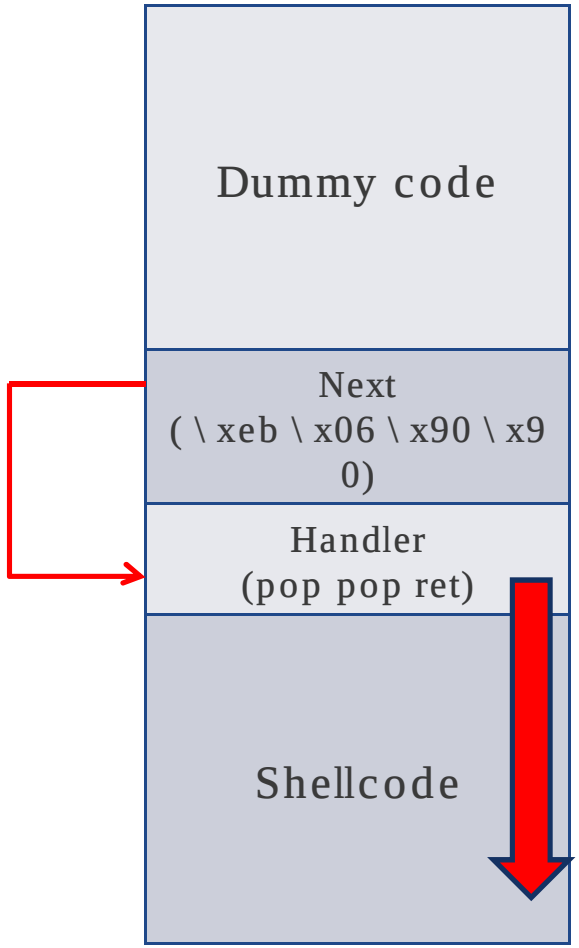
SEH overwriting 진행 과정



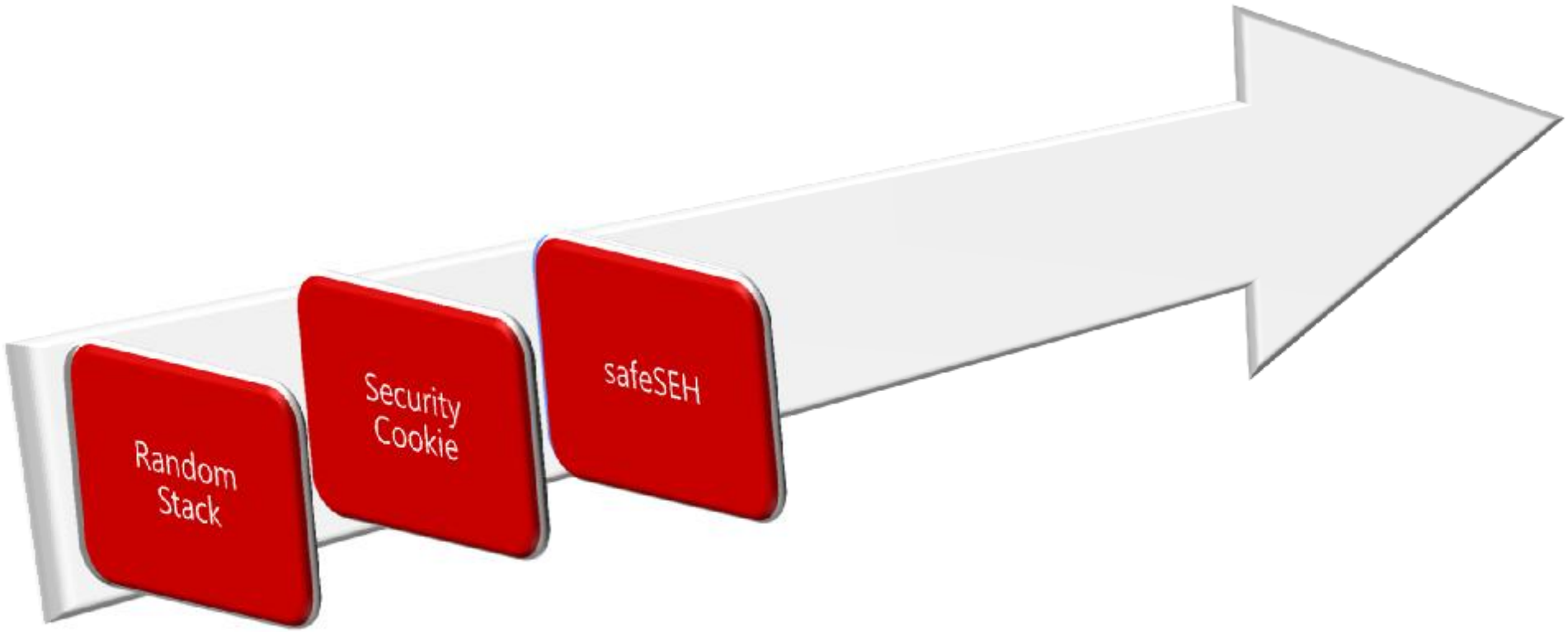
SEH overwriting 진행 과정(cont)



SEH overwriting 진행 과정(cont)



Exploit Writing 과정

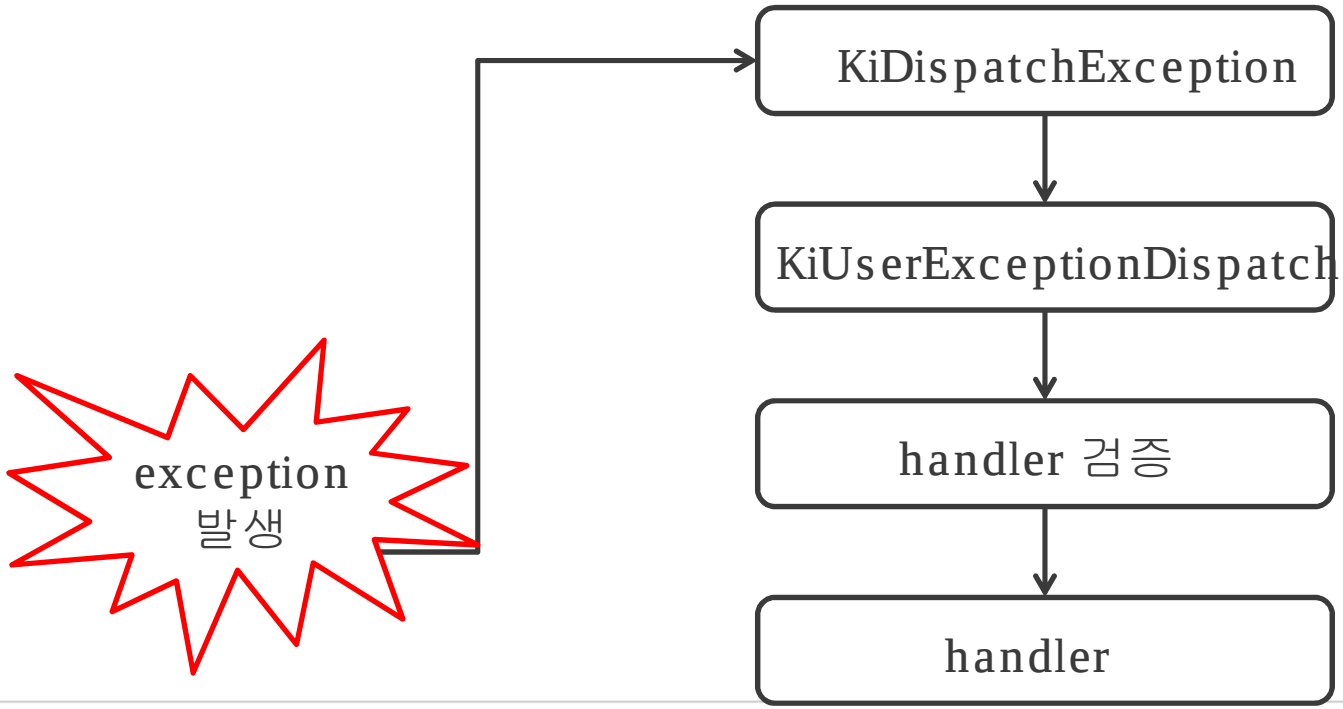


safeSEH 개요

v safeSEH?

- u SEH기반 exploit 공격을 방어하기 위한 메커니즘
- u 컴파일 옵션(/safeSEH)을 통해서 설정 가능
- u exception handler frame이 조작되었을 경우 handler호출 불가능

v safeSEH 동작 패턴



safeSEH 개요(cont)

- safeSEH 동작 패턴
  - 검증1
    - Stack으로 돌아와서 코드 수행되는 것을 방지하기 위해 TEB값 참조하여 stack 주소 범위 확인
    - exception pointer가 이 범위 안에 포함되게 되면 handler호출하지 않음
  - 검증2
    - handler pointer가 로드 되어 있는 모듈의 주소범위에 포함 되는지 확인
    - 포함되어 있을 경우 등록된 handler인지 아닌지 확인하여 호출여부 결정

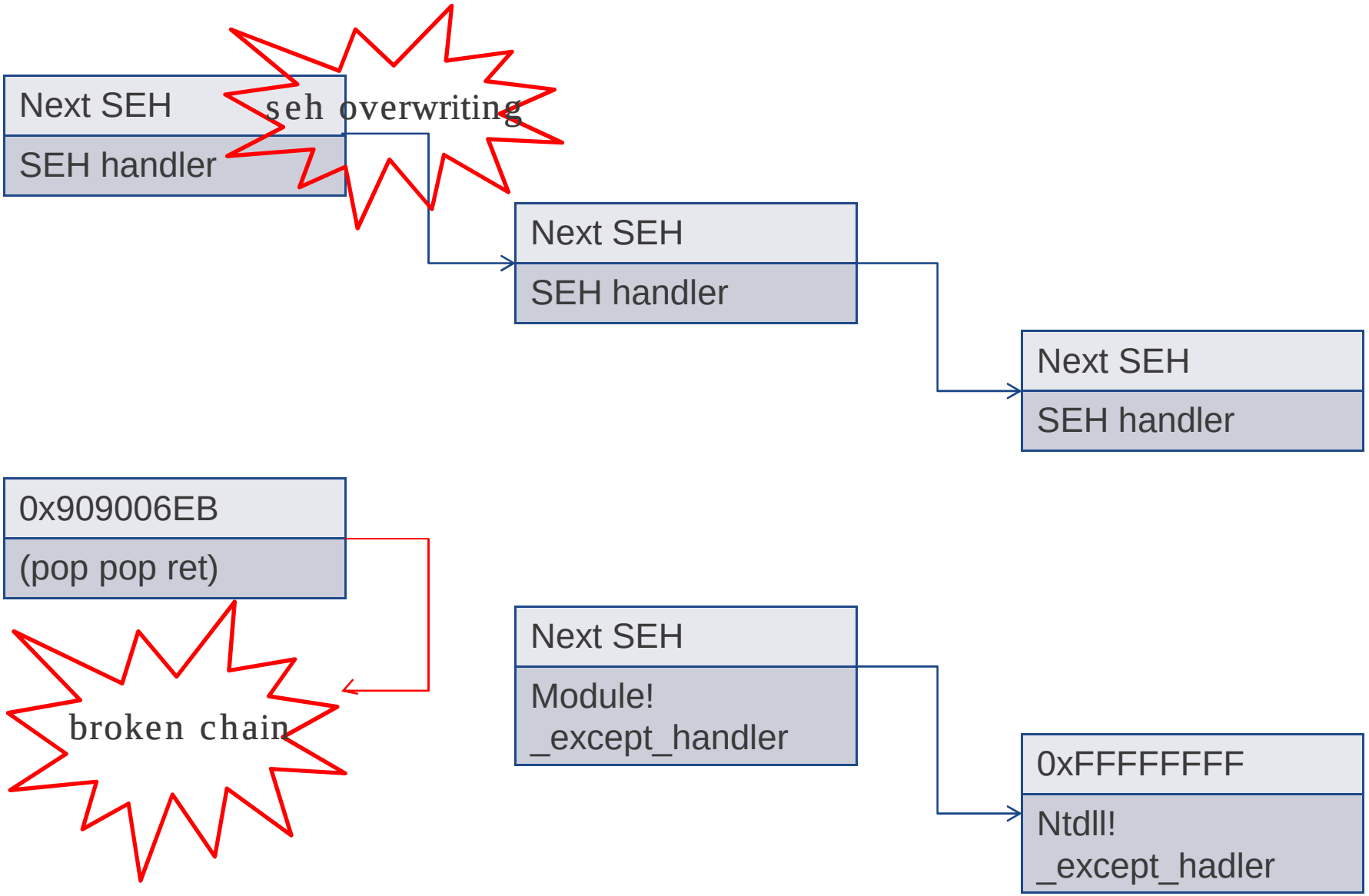
Exploit Writing 과정

SEHOP 개요

- SEHOP?
  - Structured Exception Handling Overwrite Protection
  - SEH 마지막 chain이 ntdll!_except_handler인지 아닌지 점검

- SEH overwriting 한계점
  - handler에 “pop pop ret” 명령어 코드의 주소를 넣고 nSEH값을 (JMP or nop)로 조작 할 경우 SEH chain이 깨어짐

SEHOP 개요(cont)

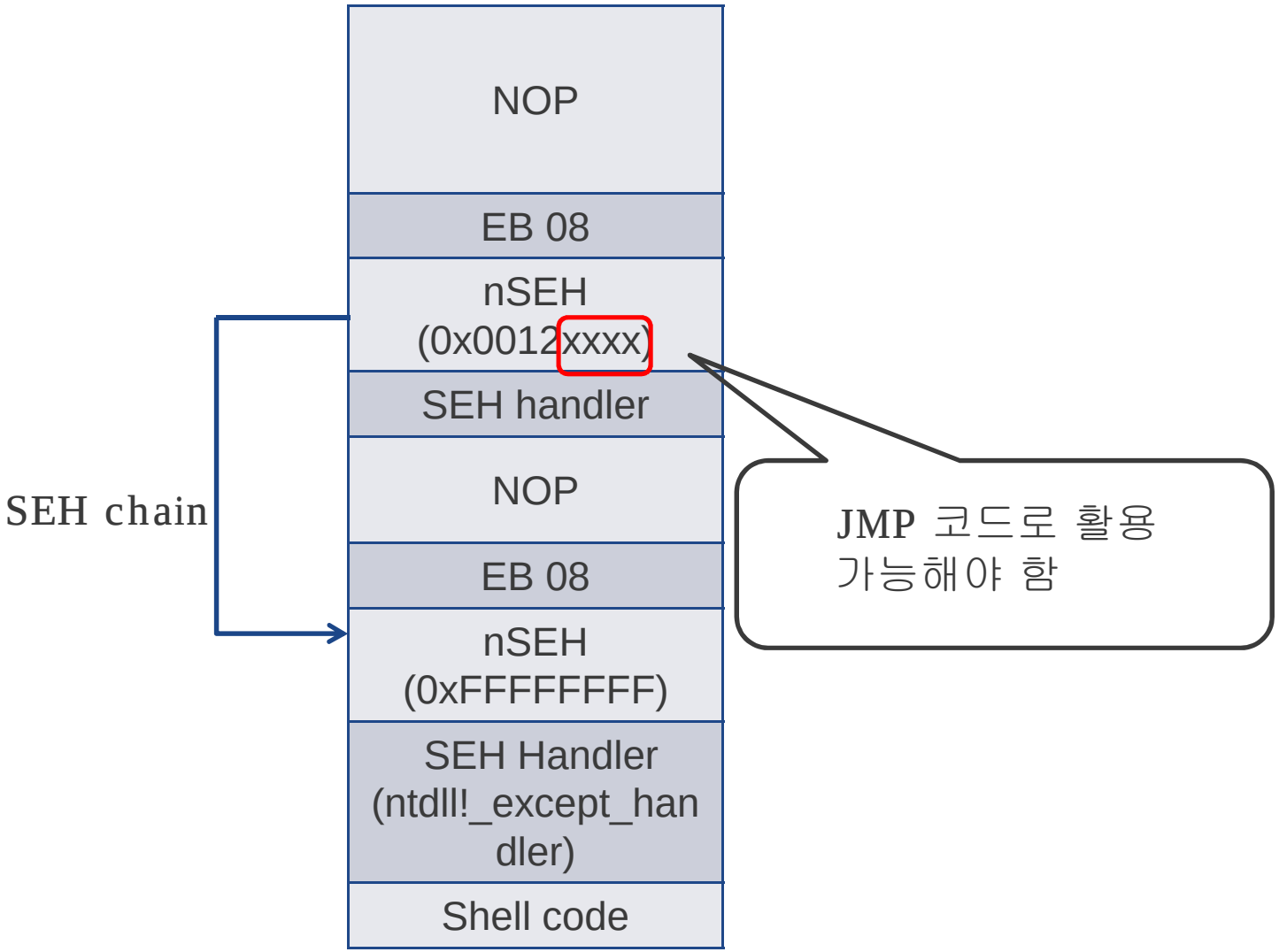


SEHOP 개요(cont)

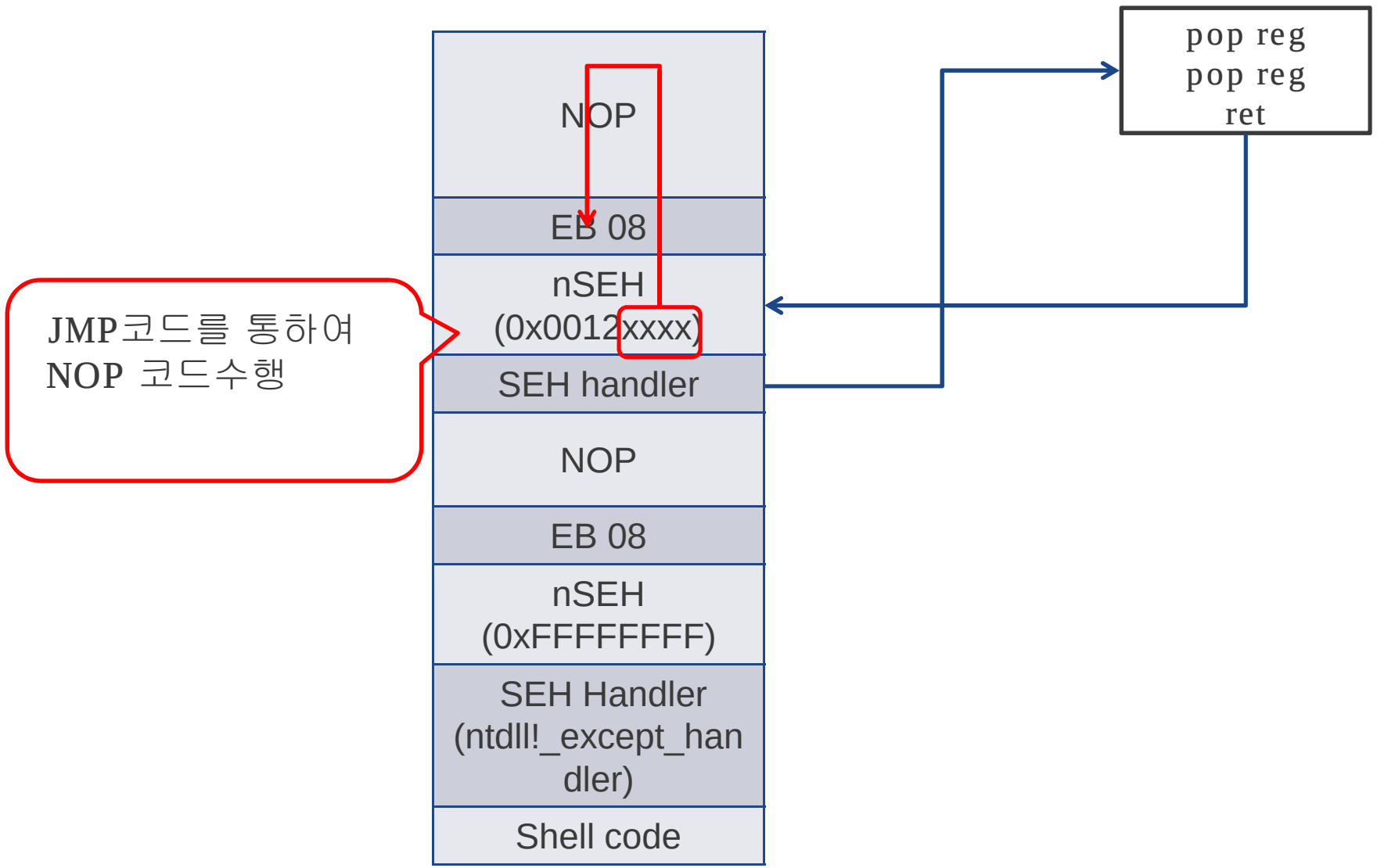
- SEHOP 적용 버전
  - Microsoft Windows 2008 SP0
  - Microsoft Windows Vista SP1
  - Microsoft Windows 7

Exploit Writing 과정

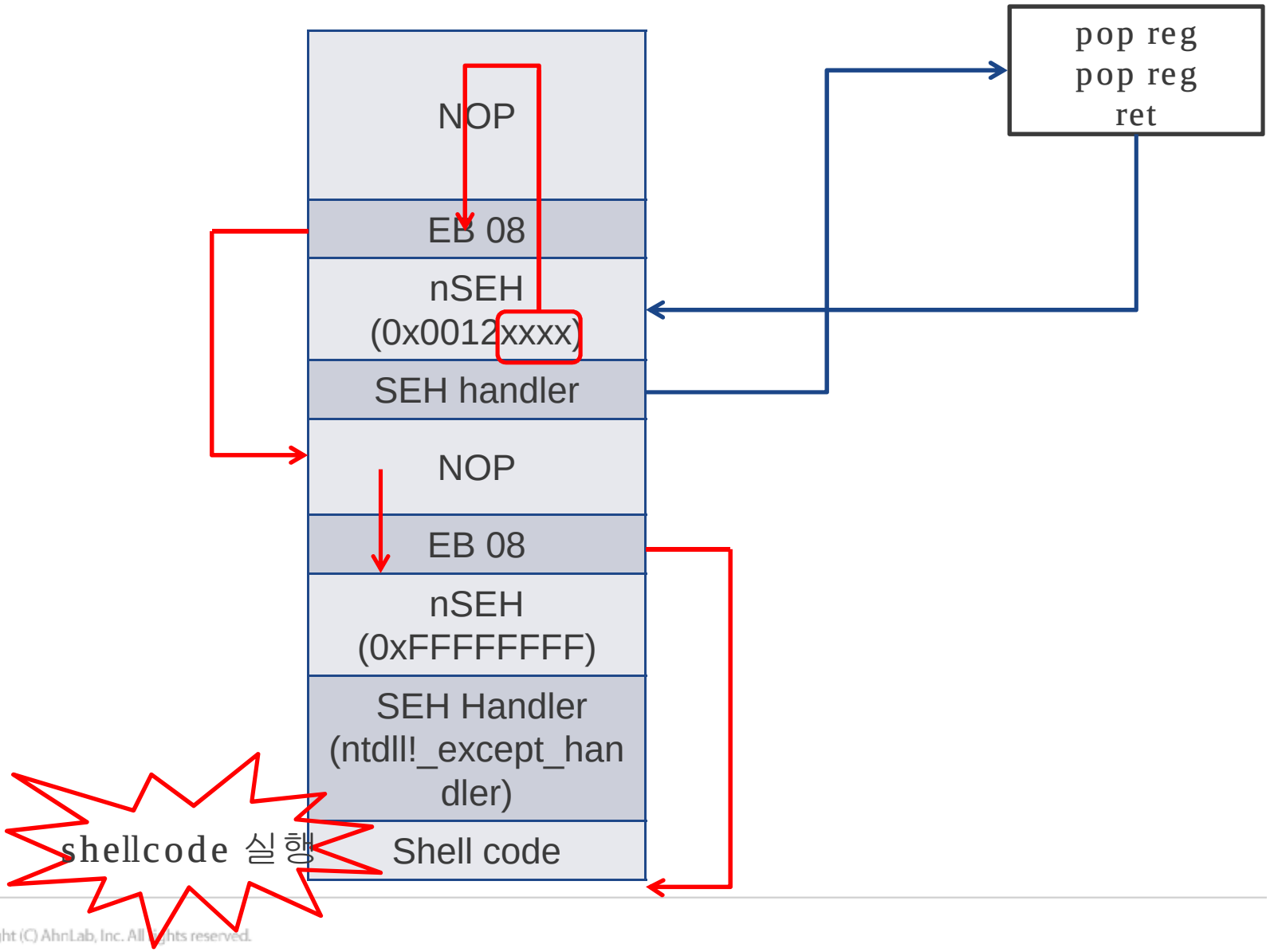
exploit code 진행 과정



exploit code 진행 과정(cont)



exploit code 진행 과정(cont)



Exploit Writing 과정

DEP 개요

- DEP?
  - 데이터 실행 방지(Data Execution prevention)
  - stack/stack의 일부분을 non-executable page로 설정하여 stack에서 shellcode가 실행되지 못하게끔 함

- DEP 모드
  - Hardware DEP
    - NX bit((No Execute page protection – AMD)
    - XD bit(execute Disable – INTEL)
    - 하지만 지원하지 않는 CPU일 경우 활용할 수 없다.
  - software DEP
    - CPU가 지원하지 못할 경우 Windows DEP는 Software SEP로 동작

DEP 개요(cont)

- DEP 설정 값
  - OptIn
    - 일부 시스템 바이너리와 프로그램에 대해 DEP 적용
  - OptOut
    - DEP가 적용되지 않는 특정 프로그램 목록에 있는 것 외에 모든 프로그램 DEP 적용
  - AlwaysOn
    - DEP제외 목록을 사용할 수 없으며 DEP 시스템 호환성 수정 프로그램이 적용되지 않는다.
  - AlwaysOff
    - Hardware DEP지원 관계 없이 DEP가 시스템 전체를 보호하지 않음

DEP 개요(cont)

v Windows 버전 별 DEP 설정

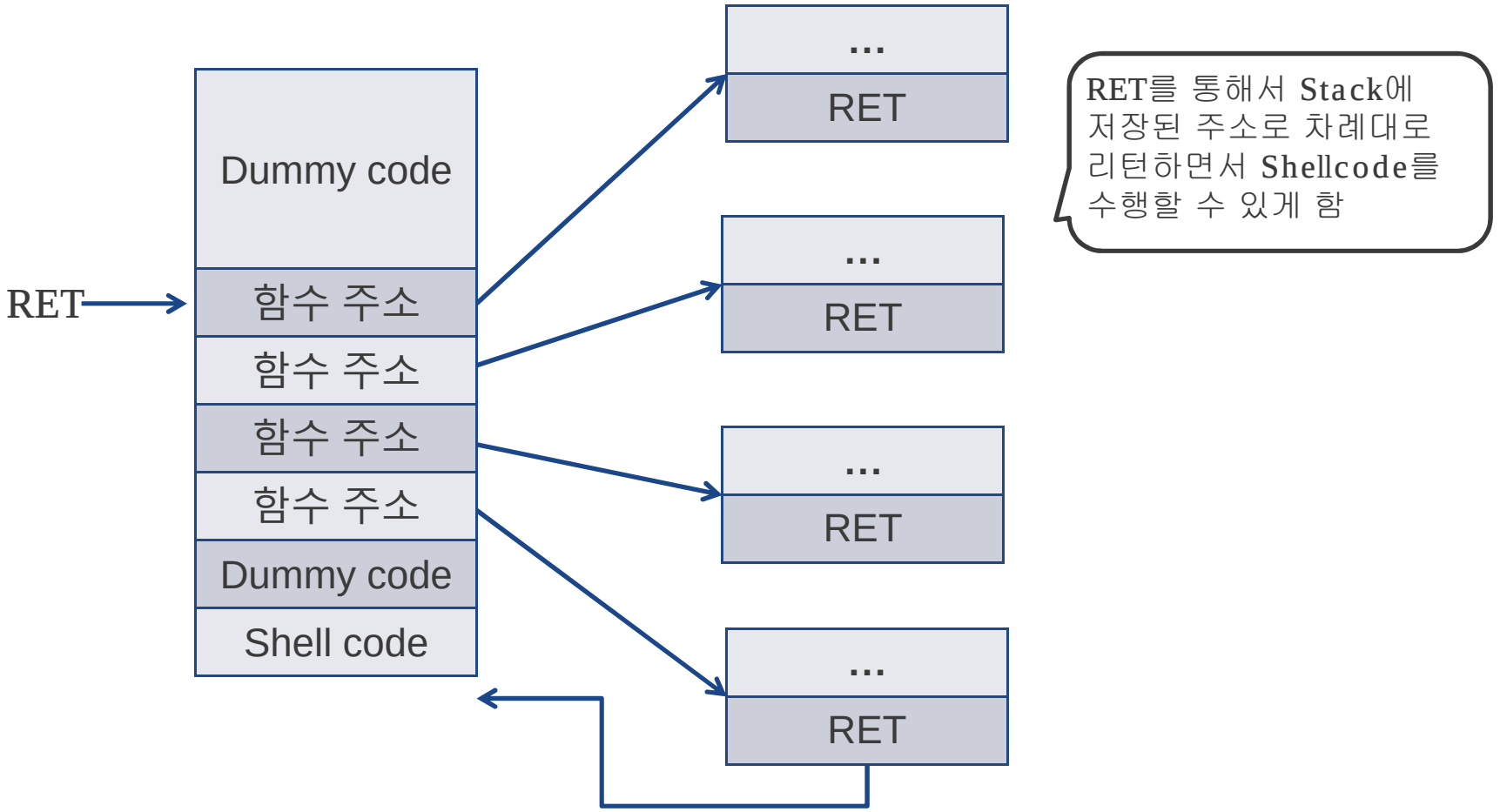
OS 버전	설정 값
Windows XP SP2, SP3, Vista	OptIn
Windows Vista SP1	OptIn + Permanent DEP
Windows 7	OptIn + Permanent DEP
Windows Server 2003 SP1	OptOut
Windows Server 2008	OptOut + Permanent DEP

Exploit Writing 과정

DEP 우회

v 기본 개념

- u 함수(필요한 코드가 있는) 호출 chain을 형성하여 DEP 우회



ROP 개요

v ROP?

- u Return-Oriented Programming
- u 기본적인 개념은 ret-2-libc와 동일함
- u 필요한 코드의 주소값 활용하여 Call/jmp를 반복하는 ROP chain을 생성하여 메모리 보호 기법 우회

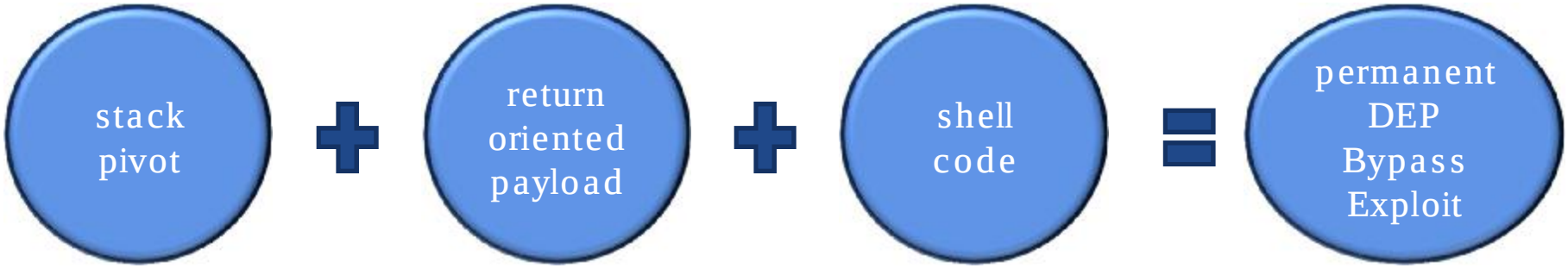
v ROP 요구 조건

- u shellcode를 실행 가능한 지역에 복사하고, 호출 가능해야 함
- u shellcode가 실행되기 전 DEP 설정을 변경 해야 한다.

v gadget

- u 목적을 달성하기 위해 필요한 코드들을 Call/ret를 반복 적으로 수행하는 ROP chain을 의미 함

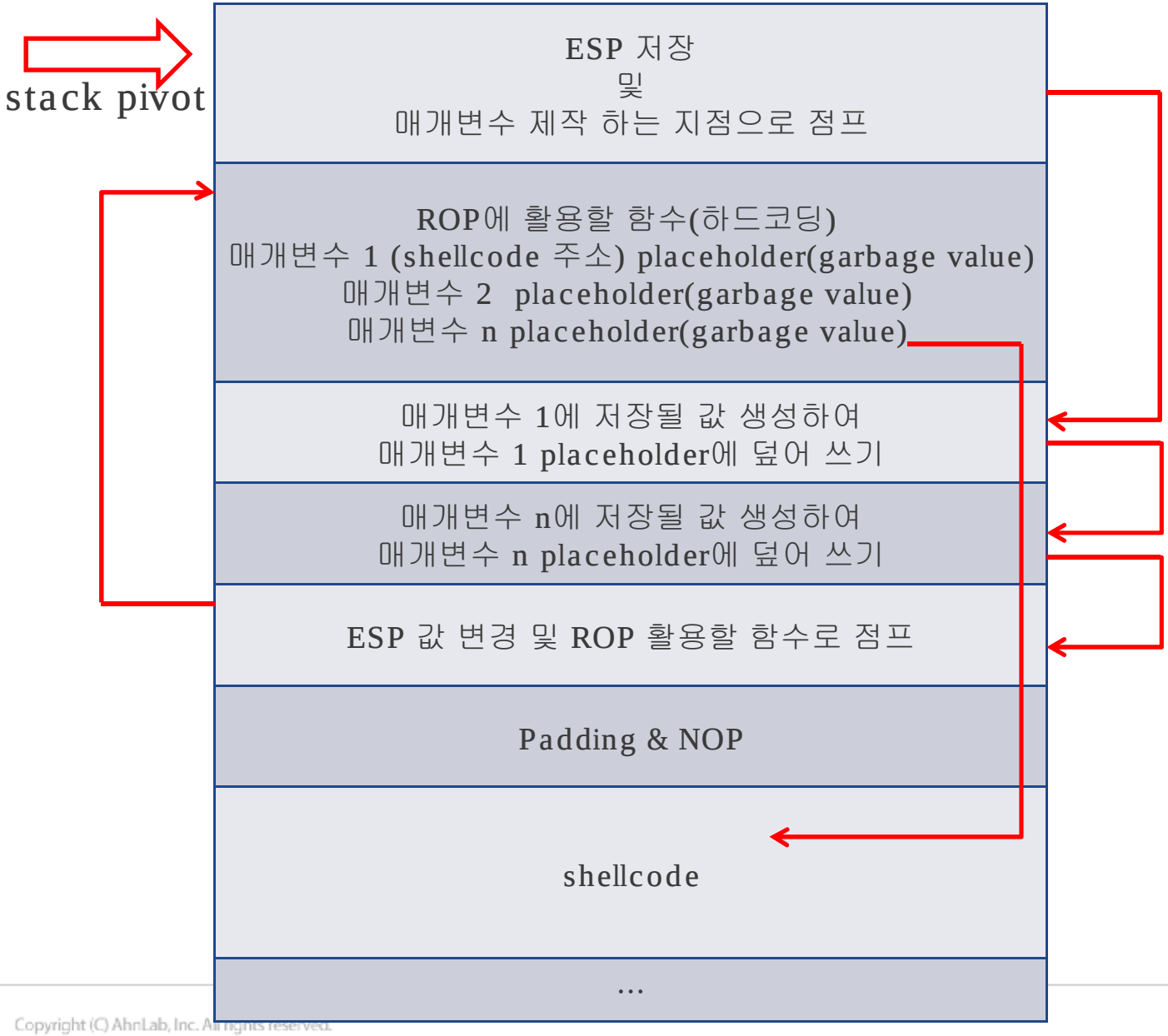
ROP 개요(cont)



ROP 기법 활용하기 위한 함수

- ROP 기법에 필요한 주요 함수
  - VirtualAlloc(MEM_COMMIT + PAGE_READWRITE_EXECUTE) + copy memory.
  - HeapCreate(HEAP_CREATE_ENABLE_EXECUTE) + HeapAlloc() + copy memory.
  - SetProcessDEPPolicy(). (Vista SP1, XP SP3, Server 2008, DEP 정책 설정이 OptIn, OptOut 일 때만 정책 수정 가능)
  - NtSetInformationProcess(). 현재 프로세스의 DEP 정책을 바꾸는 함수
  - VirtualProtect(PAGE_READ_WRITE_EXECUTE). 특정 메모리 페이지의 접근 권한을 설정하는 함수
  - WriteProcessMemory().

exploit code 진행 과정



Exploit Writing 과정

Are you ready?

AhnLab

thank you.

AhnLab