



Exploitation 102

Alexander Sotirov
alex@sotirov.net



Overview



- Exploiting stack overflows
- Exploitation mitigations in depth (GS and SafeSEH)
- Practical exploitation of Internet Explorer

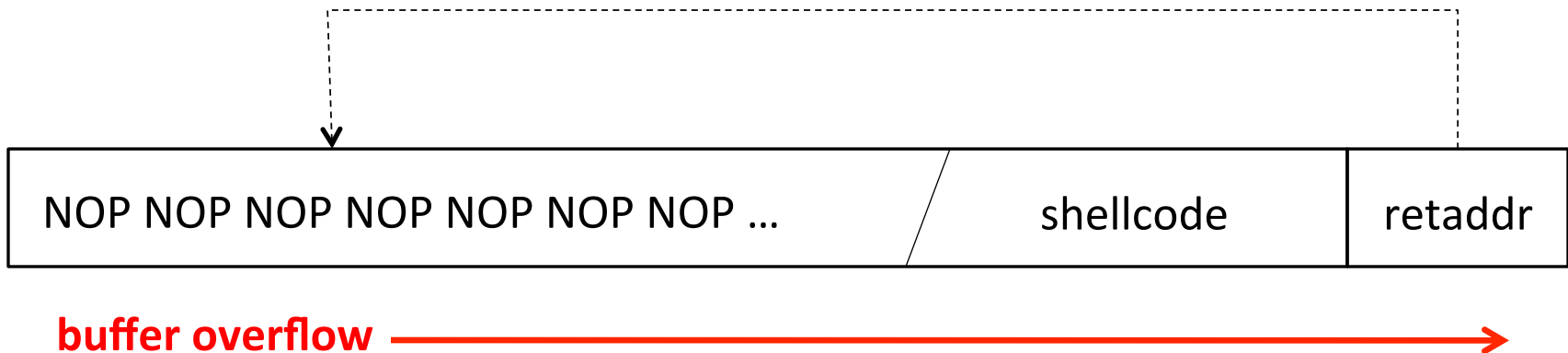


Part I

Exploiting basic stack overflows

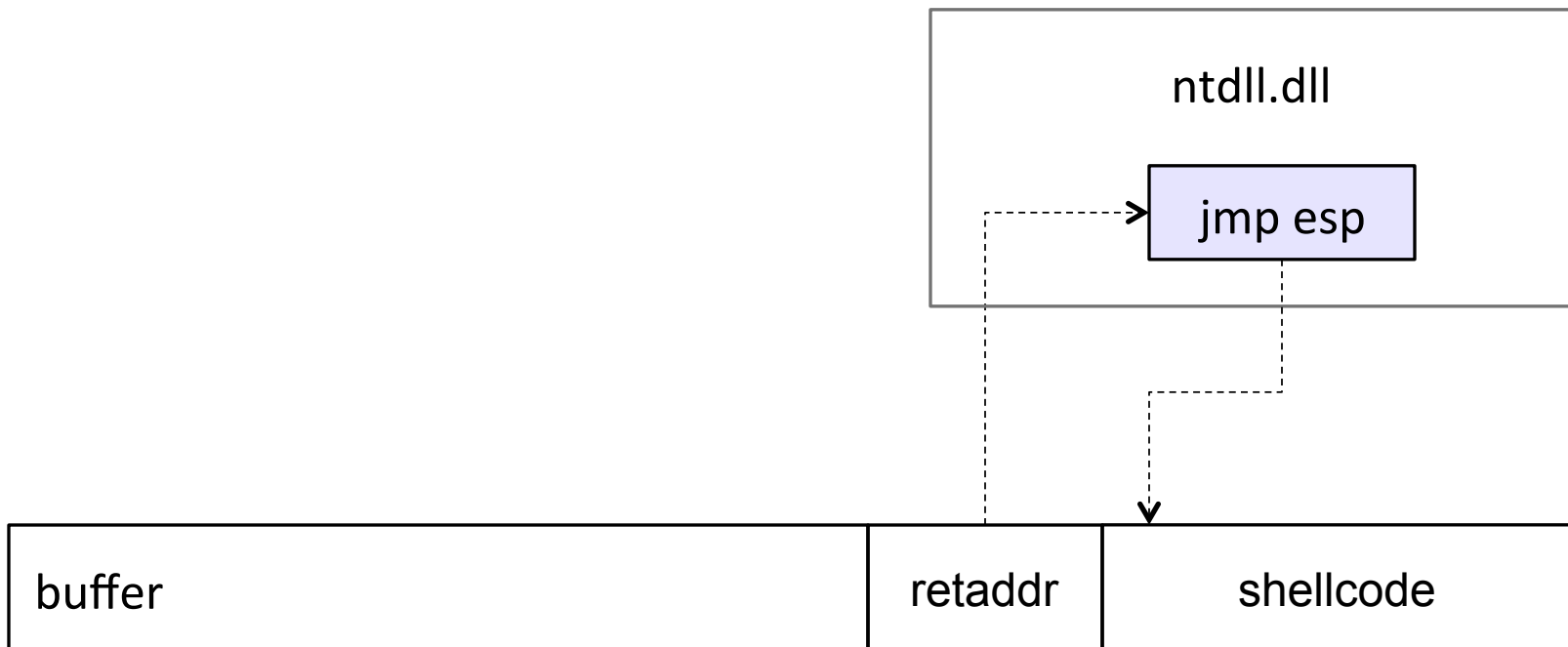
Exploiting stack overflows

Single-threaded applications with a static stack base address:



Exploiting stack overflows

Multi-threaded applications, ntdll.dll loaded at a static base address:



buffer overflow →



Diving into Internet Explorer



Part II

Exploitation Mitigations

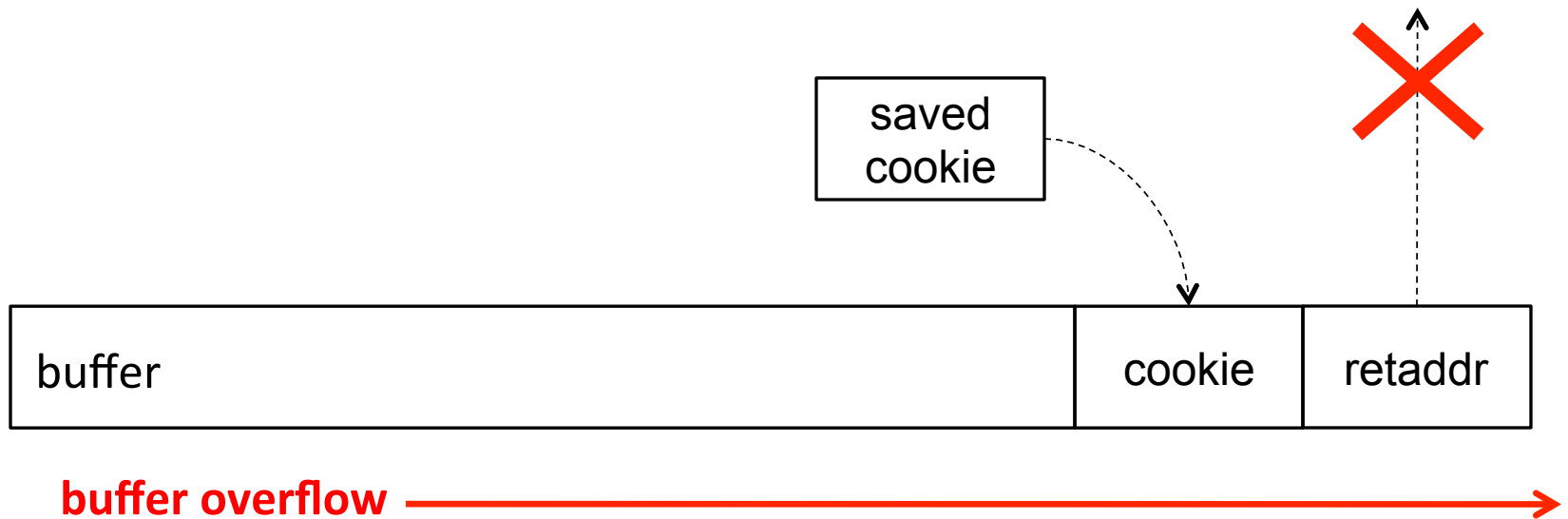
GS stack cookies



GS prevents the attacker from using an overwritten return addresses on the stack:

- adds a stack cookie between the local variables and the return address
- checks the cookie at the end of the function

GS stack cookies



Breaking GS

The function might use overwritten stack data before the cookie is checked:

callee saved registers

copy of pointer and string buffer arguments

local variables

string buffers

exception handler record

gs cookie

saved frame pointer

return address

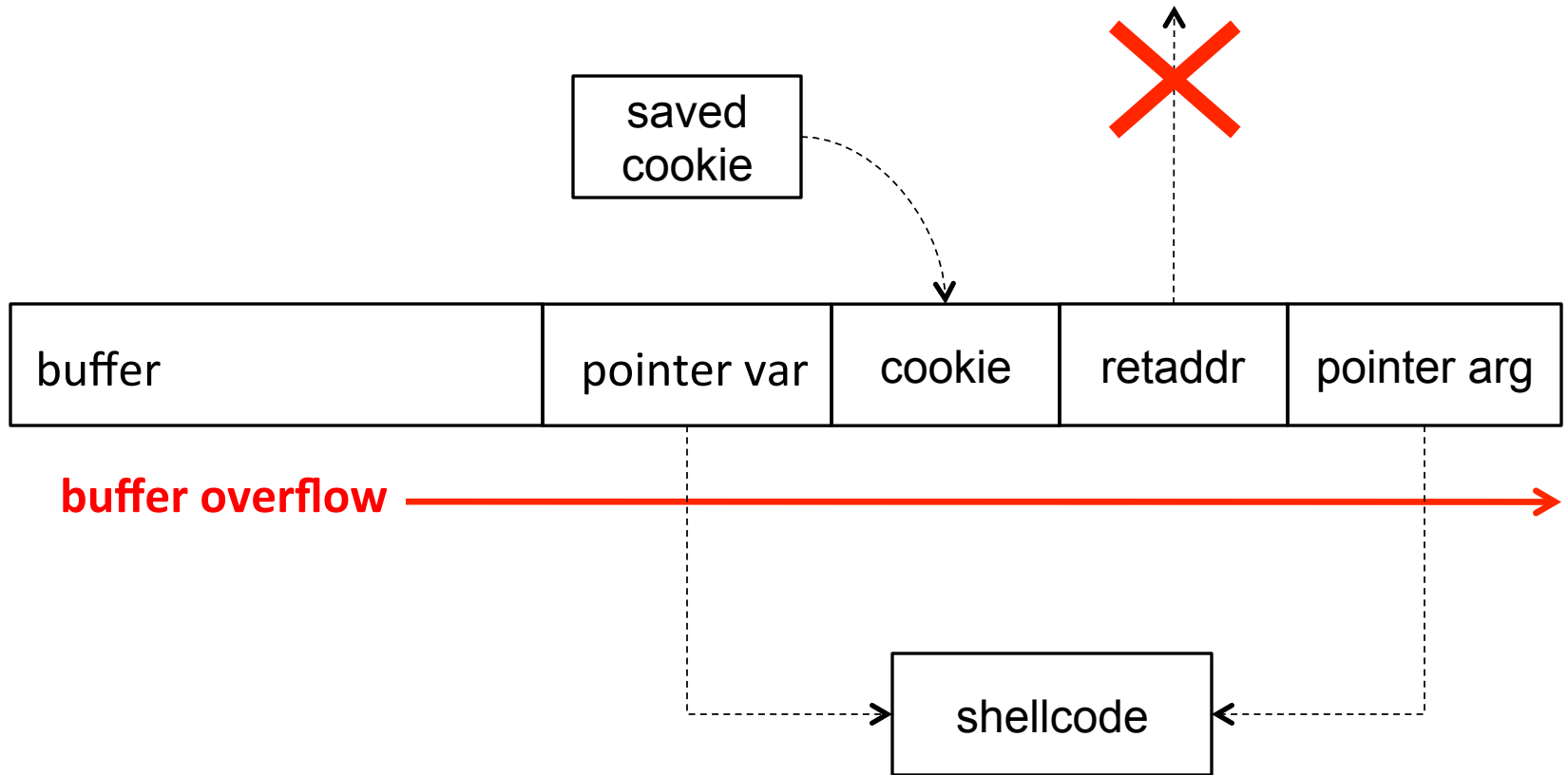
arguments

stack frame of the caller

o
v
e
r
f
l
o
w



Breaking GS



GS variable reordering



Prevents the attacker from overwriting other local variables or arguments:

- string buffers go above other variables
- arguments are copied below the local variables

source code

```
void vuln(char* arg)
{
    char buf[100];
    int i;
    strcpy(buf, arg);
    ...
}
```

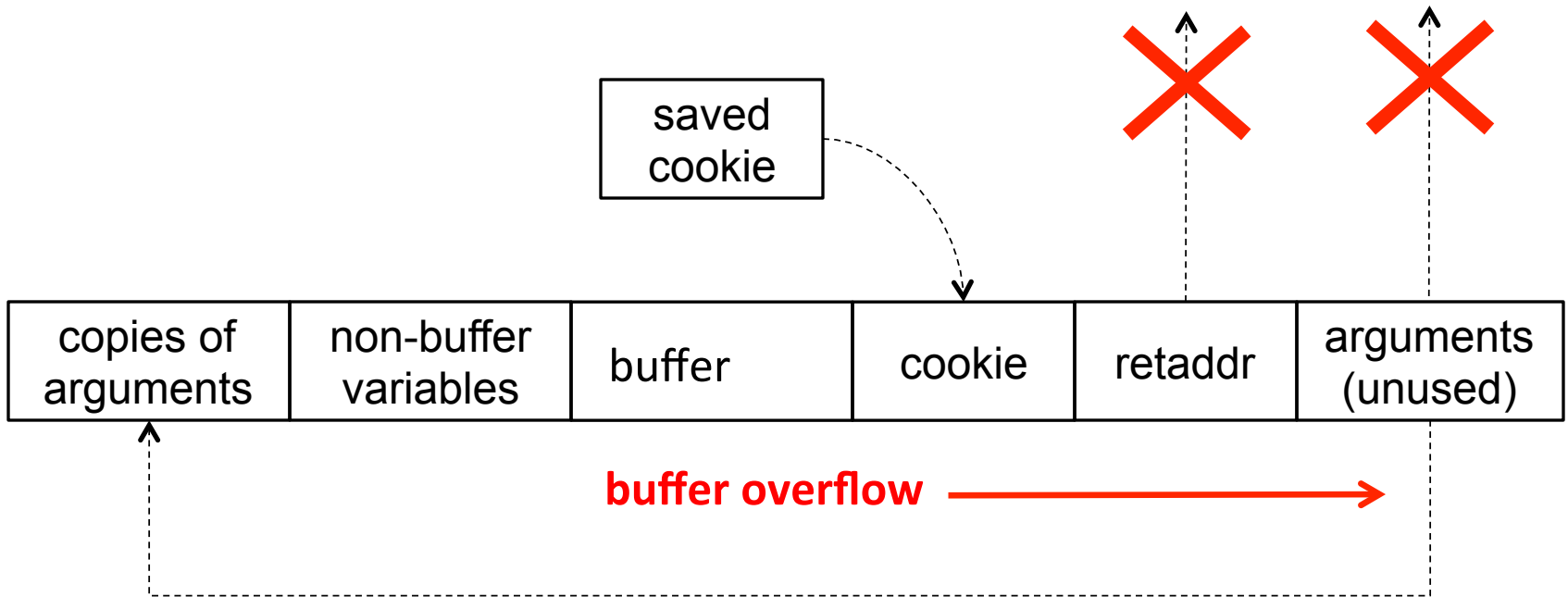
standard stack frame

```
buf
i
return address
arg
```

stack frame with /GS

```
copy of arg
i
buf
stack cookie
return address
arg (unused)
```

GS variable reordering



pointer arguments are copied
before the other variables

Breaking GS, round 2

Some function still use overwritten stack data before the cookie is checked:

callee saved registers

copy of pointer and string buffer arguments

local variables

string buffers

gs cookie

exception handler record

saved frame pointer

return address

some arguments

stack frame of the caller

o
v
e
r
f
l
o
w



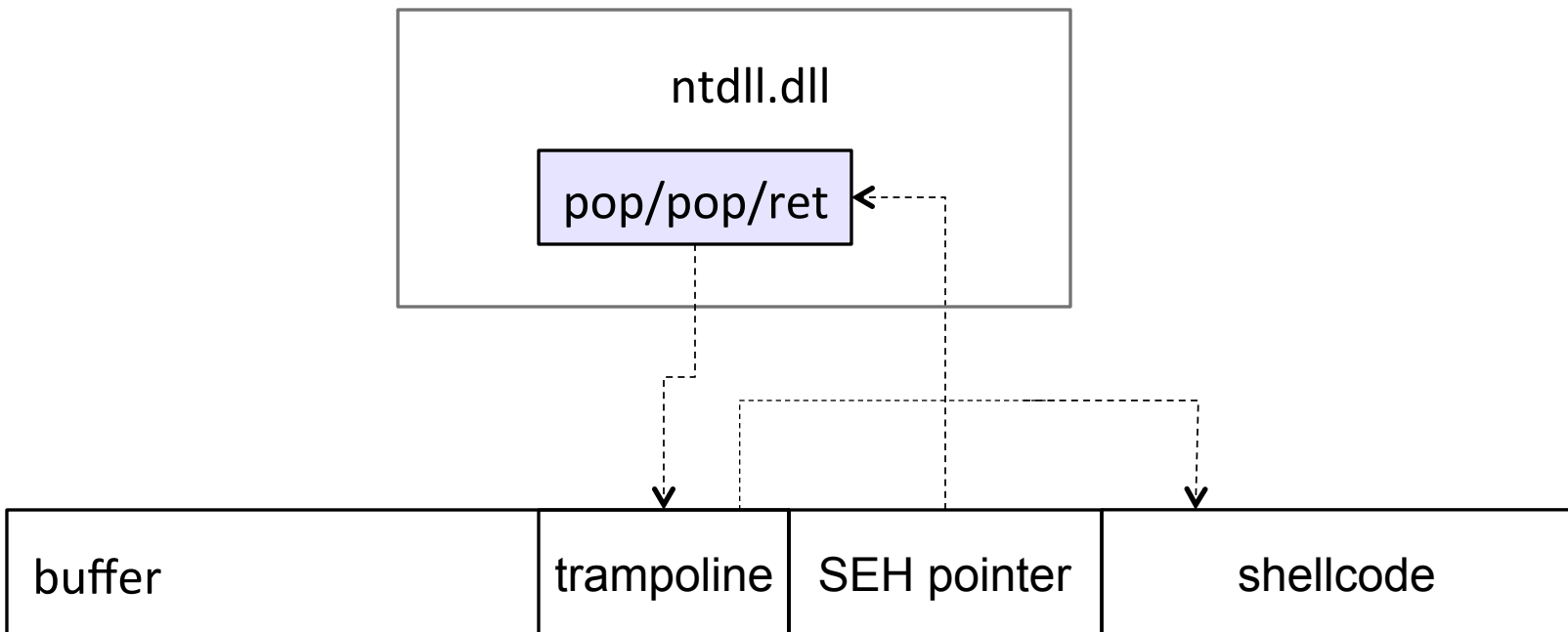
Structured Exception Handling



- Pointers to exception handlers are stored in the stack frame of each function
- All exception handler records are linked in a linked list, with head at `fs:[0]`
- The exception handler dispatcher calls the the first handler on the list.

Overwriting exception handlers

Windows SEH pointer overwrite followed by access violation before the function returns:



pop/pop/ret



- When the exception handler is called, the third word on the stack points to the SEH record on the stack
- A sequence of two POP and a RET instruction will return to our shellcode on the stack.

Bypassing GS with SEH



Triggering an exception will give us control of the program execution before the GS cookie check.

- overwrite a pointer or counter variable
- overflow to the top of the stack
- application specific exceptions

SEH records on the stack are not protected by GS.

SafeSEH



- Validates that each SEH handler is found in the SafeSEH table of the DLL
- Prevents the exploitation of overwritten SEH records

Breaking SafeSEH



- Requires that all DLLs in the process are compiled with the new /SafeSEH option
- A single non-compatible DLL is enough to bypass the protection
- Control flow modification is still possible

SafeSEH: DLL without SafeSEH



If DEP is enabled:

- Find a DLL without a SafeSEH table
- Point the SEH handler to code in the DLL

We can use ActiveX to load third-party DLLs in Internet Explorer.