David Maynor
Internet Security Systems

NX: How Well Does It Say NO to Attacker’s eXecution Attempts?

NX. It’s known by different names to different people. AMD calls it Enhanced Virus Protection, or EVP. Microsoft calls its support Data Execution Prevention, or DEP. After the press about how this new technology will stop hackers and worms in their tracks, many people call it a modern marvel. But this new technology has several layers of confusion surrounding it in regards to where it is implemented, how it protects and even when its on. This talk will unwrap the information while showing that at best NX is a speed bump and not a stop sign to malicious intruders.

David Maynor is a research engineer with the ISS Xforce R&D team where his primary responsibilities include reverse engineering high risk applications, researching new evasion techniques for security tools, and researching new threats before they become widespread. Before ISS Maynor spent the 3 years at Georgia Institute of Technology (GaTech), with the last two years as a part of the information security group as an application developer to help make the sheer size and magnitude of security incidents on campus manageable. Before that Maynor contracted with a variety of different companies in a widespread of industries ranging from digital TV development to protection of top 25 websites to security consulting and penetration testing to online banking and ISPs.
NX: How well does say No to an attackers eXecution attempts?

David Maynor
Research Engineer
X-Force Advanced R&D
dmaynor@iss.net

“HI! NICE TO MEET YOU!!”
You. NX. Blind Date. How awkward?

digital self defense
The journey of a thousand miles...

Start from the Beginning

The problem?
- Exploits, worms, shellcode, viruses, compromise, "root", "own"
- Becoming part of non-hackers vocabulary.
- Patch, update, new versions, Microsoft "Super Tuesday"
- Becoming part of non-admins daily conversation.

Why?
- Computer Security is no longer a movie subplot.
- It affects people where its most noticeable, the wallet.
- Worm stops ATM machines from working
- Worm could cost up to 50 billion dollars
  http://www.computing.co.uk/news/1156955

We have all seen buffer overflows, but here is a quick recap.

```c
#include <stdio.h>
#include <string.h>

void bob(char *badstr)
{
    char dest[5];
    strcpy(dest, badstr);
    printf("copy done\n");
}

int main(int argc, char **argv)
{
    bob(argv[1]);
    return 0;
}
```

digital self defense
What about buffer overflows are strange?
- Where code is being executed from.
- Code is being executed.
- Overwrite of return address
- Strange function calls
- Code on stack or heap

Ok, level with me, is this a hardware or software problem?
- Buffer overflows occur due to poor or rushed software design
- Hardware does have some blame, loose execution of code

What Is NX?
- A bit in the pagetable of certain microprocessors.
  - 63rd bit in the table.
  - 0 = executable, 1 = nonexec.
- Pagetable must be in PAE format
- The ability to set read/write/execute page permissions has been around since the 386, its just been all or nothing.
- By setting pages that contain no executable code to not executable code injection exploits like stack and heap based buffer overflows can be stopped!
- Intel calls their own version, XD, that is identical except for the name.
- You may see NX referred to as EVP, or Enhanced Virus Protection, in AMD docs.
Where can you get no-execute support?
AMD Athlon64 and Opteron chips
http://www.amdboard.com/amd_virus_protection.html
Inel

Is no-execute the magic bullet?
NX is not a stand alone solution, it requires Operating System support
CPU will just generate an exception.
Is it evadable or otherwise beatable?

What isn’t the no-execute technology?
Although its called EVP it doesn’t actually deal with viruses, just the exploitation vector.
This technology is designed to detect and stop buffer overflows.
If your attack doesn’t something other than code injection there is a great chance it will still work.
Only affects memory corruption exploits.
There is a lot more than buffer overflows in the world
Not a wide scale deployable solution yet.
There are many problems with the technology include Just-In-Time (JIT) compiled code and some copyright protection schemes.
Not a comprehensive solution yet.
More will be covered on the lack of comprehensive code coverage in a later section.

digital self defense
What might you confuse with NX technology?

Non-executable stack.
It sounds very similar but these are mostly seen on RISC chips that has their own version of a no-execute bit. This is most frequently associated with the “set noexec_user_stack = 1” argument found in the /etc/system file on Solaris systems.

API Hookers
Tools that overwrite the function prologue for functions that are highly coveted by hackers (things like WSAStartup(), connect(), CreateProcessA()) with a static jump into a runtime analysis engine to determine if the function was called legitimately.

Heap/Stack Canaries
Static values that are inserted into heap chunks or stack location before a function or a memory altering procedure is called. The value is then checked afterwards to make sure it hasn’t been overwritten.

The journey of a thousand miles…

If all of these methods of defeating buffer overflows have flaws what makes the no-execute technology that much better?

Nothing, It has it flaws just like all the other generic buffer overflow prevention systems.

Surely everybody can’t get it wrong, what is the problem?

The problem is no-execute technology just like all the other technologies that were just discussed all have the same problem: they are named incorrectly.
The journey of a thousand miles...

If you wanted to stop this car from crossing the bridge, now is a bad time to start.

“Pre-dinner cocktails with the NX.”
Getting to know your blind date.

digital self defense
Pages and Page tables and Page Table Entries

What is a page?
A fixed amount of virtual memory that can be segmented and used for memory allocation purposes.
There can be large and small pages.
   On x86 and x64 the small page size is 4k.
   On x86 the large page size is 4MB, x64 the large page size is 2MB.
The building blocks of the memory system as you know it.
Since it's the smallest block that hardware protection can be applied to, it's the perfect place to implement no-execute protection.

Pages are either in a Committed, Reserved, or Free state.

VirtualAlloc and VirtualAllocEx are the functions used to do the commit and reserves manually. (win32, linux will be covered later)

Pages can be locked into memory and pages can be shared between processes.

What is a page table?
The virtual memory layout for a process is stored in page tables.
The are normally only modifiable by the kernel to keep a process from changing its layout during runtime.
Used for the translation of virtual memory into physical memory.

What is a page table entry (PTE)?
This is what actually contains the virtual to physical mapping of memory.
Each virtual address has one.

There is a lot more to pages like the page directory index, the page table index, how how to virtual memory lookup is actually done. That's out of our scope here.
I recommend reading Windows Internals, 4th edition for more information.
So how does all this relate to NX and what does NX do to it?

Normal x86 PTEs are too small. No-execute technology uses a PAE formatted 64bit wide PTE with the 63rd bit as the NX bit. When code attempts to execute in a page marked no-execute the CPU raises an exception. The error code is OS dependent. What happens after the exception is raised is OS dependent as well. The CPU does not enforce the no-execute rule directly.

How does the OS decide what is code and what is data to mark pages correctly?

The loader!!!
Flow of the loader
Where the enforcement is set/comes from.

How does this affect you?
3rd party loaders
cygwin
**Demo: Windows vs. Cygwin**

Windows has Cygwin installed on WinXP SP2 with DEP enabled for all programs.  
A bufferover run natively is caught. <Fig. 1>  
A buffer overflow from cygwin is not. <Fig. 2>  
Why?

---

**Fig. 1**

---

**digital self defense**
digital self defense
Lets look under the hood…

Again, why?
- Different loader?
- Pages are not marked correctly?
- Different Memory management?

Why do you care?
- No-execute technology isn’t extended to everything.
- Default settings for DEP?
- False sense of security.
- NX protection has to have OS support to work.
  - It can be selectively enabled/disable
  - VirtualProtect (for windows)

“The dinner”
Don’t order anything messy.

digital self defense
DEP In-depth
DEP is more than just the MS implementation of no-execute technology.
http://support.microsoft.com/kb/875352
   Explanation of DEP straight form MS.
Hardware vs. Software enforced
Can be a bit confusing as to when you are or are not protected.
Many features what hardware and state the OS is running with.
   PAE
   No-execute enabled chip
   Boot options

Hardware enforced DEP close up
This is what we have been discussing most of the talk
Requires no-execute enable chip
Protection/enforcement is done on a per virtual memory page basis
 Protects both 32bit and 64bit versions of the OS if correct chip is used.
32bit versions of the OS require PAE enabled
Protection is configurable via control panel and boot.ini options
**Software DEP close up**

Windows supports it…

Software only attempt to detect buffer overflows
Attempts to detect “Stupid SEH Tricks”
Uses SafeSEH
   - Before an exception is run SafeSEH makes sure it’s a valid handler
   - Makes sure it’s not in executable memory
Processor independent
A lot of popular exploits use SHE tricks to get execution
…Alot don’t.

---

**Boot.ini and you, a winning combination!**

```
/NoExecute=
```

Flag used in boot.ini to configure the system wide policy

```
OptIn
```

Default config. Covers system binaries only and other programs you define.

```
OptOut
```

Enabled for all processes except those you define

```
AlwaysOn
```

Covers everything!

```
AlwaysOff
```

Turns DEP off completely.

---

digital self defense
Problems with Windows?
MS maintains a list of software that DEP breaks
   Breaks java, copyright protection tools, games
Application compatibility toolkit
DisableNX
Device drivers
Kernel space

So now that you know all about it...why is no-execute bad?

Brand new for the first time ever released new 0day exploitation technique.
RETURN TO LIBC!!!!!

(its so cool)

Just kidding.

Although return-to-libc works its old and not sexy.

Best target of return-to-libc is VirtualProtect.

- Change page permissions
- Execute as normal

Chaining of return-to-libc

- Complex, tedious, prone to failure
- Not very portable

Return-to-Libc problems

- Makes David a sad panda
- Doesn’t help with heap overflows
**Demo:** example of return-to-llbc
Example of returning to VirtualProtect
Adding a user

**New hotness**
What is different?
Memory protection changed.
Did it all change?
How do we find out?
  Writing code
  Using a tool
What does this mean?
Evaluating the differences.

*digital self defense*
Writing code to find the differences?

VirtualQuery


Requires a address to test, MBI (Memory Basic Information) struct, and a length to test for as an args.

Example of for loop

```
for(i=0x0;i<=0x80000000;i+=0x1000){
    VirtualQuery((int *)i, mbi, 0x1000);
    switch(mbi->Protect){
        case 0x10:
        case 0x20:
        case 0x40:
        case 0x80:
            printf("%X has Execute permission", i);
            break;
        default:
            break;
    }
}
```

The return values you are interested in are defined in WINNT.H

Basically 0x10, 0x20, 0x40, 0x80.

Anything that means the page is executable.

The ideal value is 0x40, Read/write/execute.

If there are any pages that are still writeable and executable you can inject code into that address and execute.

My example does 4k chunks up to 0x7FFFFFFF, 0x80000000 starts kernel space.
Windows supports it...

Demo: Example of a run of my program on protected vs. non-protected machines

An easier way?

PrcView

digital self defense
Now we know the lay of the land...

- Code coverage
- Heap tricks
  - Turning a heap overflow into a stack overflow
- Timing
- Execution Paths

Fake Server demo
Using the examples just discussed we own a server on Win XP SP2 with /NoExecute=AlwaysOn

How did we do it?
“Desert”
I hope you saved room.

“GO GO INGO GO!”

LINUX!
Linux supports it.
http://kerneltrap.org/node/3240
Linux has supported it since June 2, 2004
Linux support was written by Ingo Molnar
You may remember him from such hits as the O(1) scheduler, work on RAID, and some wacky in kernel stuff like FTP and HTTP servers.
As in the case of Windows, Linux has prereqs that you must have before the support will be active.
Code Coverage.
Page stuff with out friends kscand and bdflush
What is needed to make Linux do the noexecute dance?

You need a CPU that supports the NX bit in it's PTEs.

PT_GNU_STACK

This allows you to mark stacks as no-exec.

readelf -l will tell you if you got it. You want your stack to be RW.

If you are covered you will see a message in your bootup

NX (Execute Disable) protection: active

No-execute support can be disabled by passing noexec=off via the commandline.
This protection is included in most popular distributions.
User land vs. Kernel space
The NX protection provides userland support
Kernel space overflows are also protected against

Problems?
Many opensource tools are not known to follow standards very well.
In homegrown or custom built apps stacks have to manually be
marked as non-executable
Things that are dynamically generated at runtime can cause serious
problems.

Demo: Stack based overflow caught!
Simple stack overflow
Other security tools like libc address randomization have been
disabled.
Fedora Core 3 64bit.
Funny thing, RAX instead of EAX.
Demo: Process with and without NX support.
- What is marked what?
- Are there places I can still get to?

Return-to-libc
- Less effective because of address randomization
- Ours is disabled for NX demos
- 64bit registers

What is actually done when the kernel catches something bad?
- printk
- Allow or disallow?

Where do the page permissions get set?
- Loader
- kernel

digital self defense
How does this affect the heap?
Heap de-armoring
Arbitrary overwrites vs. execution
Playboys and their pointers
Dereferencing NULL?
Arrays.

Demo: Owning Linux
Server that checks and returns file status remotely
Stack and heap based overflow
Other security measures besides NX have been disabled
b00m
"GO GO INGO GO!"

How did this happen?
Say hello to my little friend!
Things kernel developers don’t think about but security researchers do.

“Good night kiss”
This is assuming you haven’t baled

digital self defense
Was this a good idea?

It's not the end all security solution.
Major compatibility issues with existing code.
Code Coverage.
Alternate flow of data.
Only helps with memory corruption bugs.
  Demo: Change my UID
Should not replace any current security tools but is a nice addition.

Why isn't this the silver bullet?

The problem is very complex.
  Airport metal detectors and fingernail clippers.
A patch on top of a band aid on the outside of a bandage wrapped around a cast.
  It's time to start fresh.
Can you blame anyone for NX not living up to the hype?
  It's less about programming errors and more about the architecture being to open.
Can it be fixed?

Not without a major redesign of the architecture.
The examples shown are not the only ways to avoid this protection.
Adopt a layered approach and don’t count on one single security feature.
   NX+Libc Addres randomazation+API hooking+host based IPS+Network Based IPS
   It may still have flaws but the more the layers the harder it is to trivially evade.

What is the future?

More research in trusted OS models.
Sandboxing everything.
Whatever the future may bring I can assure you we will pick it apart as well.
Thanks to Jeff Moss and Black Hat, ISS, and especially XFRD.

QUESTIONS? dmaynor@iss.net