Memory Corruption Attacks
The (almost) Complete History

thinkst applied research
haroon meer - haroon@thinkst.com
25/06/2010

Table of Contents

Introduction

Memory Corruption attacks have monopolized the headlines in the security research community for the past 2 decades. While everybody is aware that Solar Designer did some early HEAP “stuff” and that the Morris Worm “happened in the 80’s”, a great deal of information is lost in between.

This problem is exacerbated because of some truly poor record keeping and a general disregard for attribution when publishing security research.

To their defense, some researchers have pointed out that online resources (like posts to mailing lists and forums) make for poor reference sources due largely to the transient nature of mail archives and forum postings.

A separate result of poor record keeping is the fact that techniques devised and published informally are often mis-credited in Academic journals (if they are indeed credited at all.) To combat this, a more formal, academic version of this paper is currently near completion which is aimed essentially at reading hacker folklore into the academic rolls.

It seems obvious that a central location where all of this information can be easily found, searched and annotated is an idea who’s time has come. It is instructive to be able to view all of key developments regarding memory corruption attacks and defenses in a single document. This makes trends and inflection points much easier to spot.

This document therefore aims at doing just that. I.e. presenting a reasonably complete timeline of memory corruption attacks and mitigations from the earliest documentation of them to the present day.

Even while compiling this document, new events have been added to the online timeline. This is a good indication of the fact that the live website will almost always trump the static written document. The reader is advised to make use of http://ilm.thinkst.com/folklore/ for the most up to date version of this timeline.

Caveats

A quick glance at this document (and indeed the current online timeline) will reveal a distinct western bias. That is to say that with only a few rare exceptions, the pieces of the folklore covered are based on English documents, and postings to english language forums. History has shown that there is a wealth of talent hidden from monoglots behind foreign alphabets and this is surely the case here too. My hope is that the online timeline will be useful enough for polyglots to fill in this void.

A 2nd caveat is to mention that there is obviously a great deal of important research on non x86 platforms (that inform attacks against x86) that have been omitted. This paper had to be delimited somehow, but the online timeline need not. Going forward they will be added online.

Background

A common enough response, when talk turns to memory corruption attacks is that they were a problem for the 90's, that are increasingly rare today. This paper should show that although public work on attacks and defenses did indeed have a local maximum during the 90's, the memory corruption problems are far from being hunted into extinction.

Memory Corruption Bugs

We have come a long way from the days when almost every memory corruption bug was called a “stack overflow” and when the purpose of every piece of shellcode was merely to spawn a shell. In the interests of giving this paper some boundaries we have made use of the following limits.

a) We choose to focus on attacks and mitigations on the x86 architecture against Win32 and Linux Systems exclusively.
b) We use the term “memory corruption attacks” to refer to attacks that allow an attacker to deterministically alter the execution flow of a program by submitting crafted input to an application.

OSVDB

The Open Source Vulnerability Database (OSVDB) is an open source database created by the security community in 2002 to provide “*accurate, detailed, current and unbiased technical information on security vulnerabilities*”. A quick scan of the DB shows that many (many) of the older entries lack critical information regarding the precise nature of documented bugs. This is often because such information was never reported by the original researcher or the vendor involved. In many cases however we are able to discover enough information on the bug (by doing a fairly exhaustive search of other sources) to be able to classify the bug as either “Memory Corruption” or “Non Memory Corruption” vulnerabilities. (We ignore Denial of Service (DoS) bugs for our purposes).

In some cases we are able to mass classify ranges of vulnerabilities due to their nature (eg: “SQL Injection in XXXX”, or “XSS in YYYY”), and in other cases the title of the reported bug itself aids in our classification even if the reported details do not. Using this rough classification over the entire database of vulnerabilities in the OSVDB leaves us with the following graph

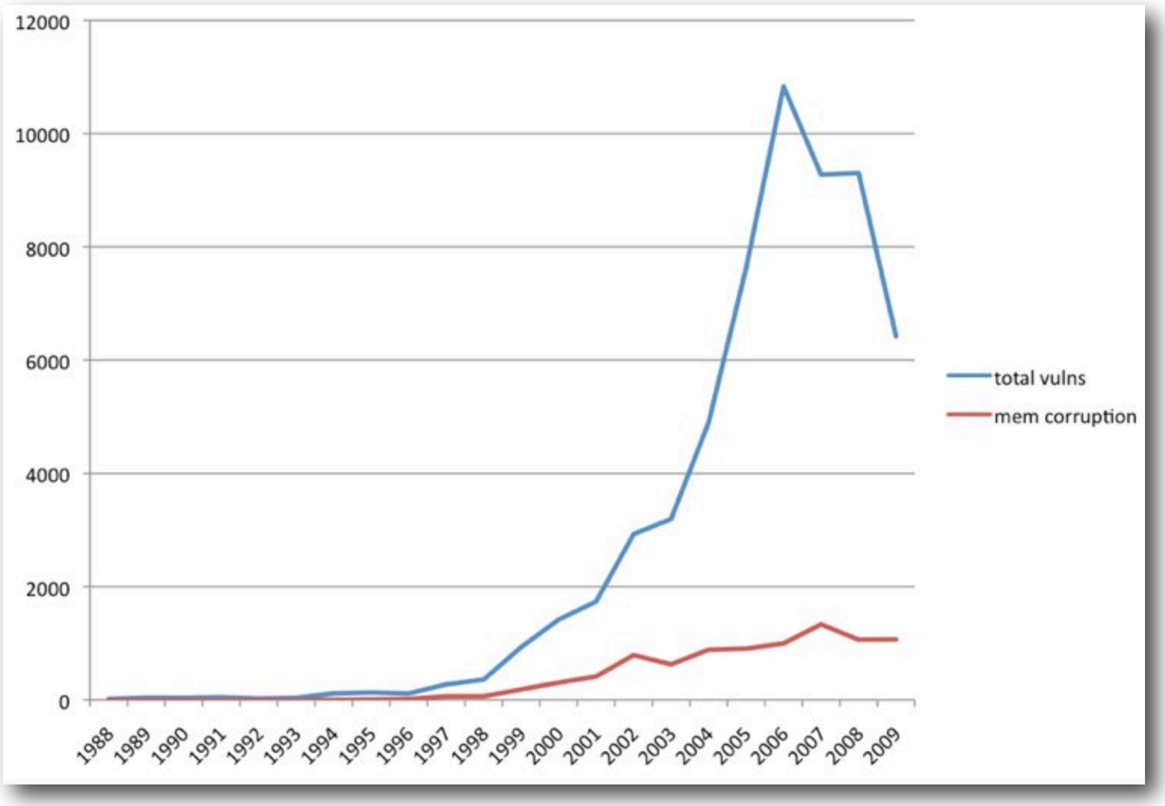


Figure 1 - Memory Corruption Bugs vs. Total Reported Bugs (OSVDB)

A quick scan of the graph could lead one to believe that memory corruption bugs are indeed disappearing, since the delta between such bugs and total reported bugs appears to be decreasing, but one would be wrong.

An examination of the percentages reveals the following:

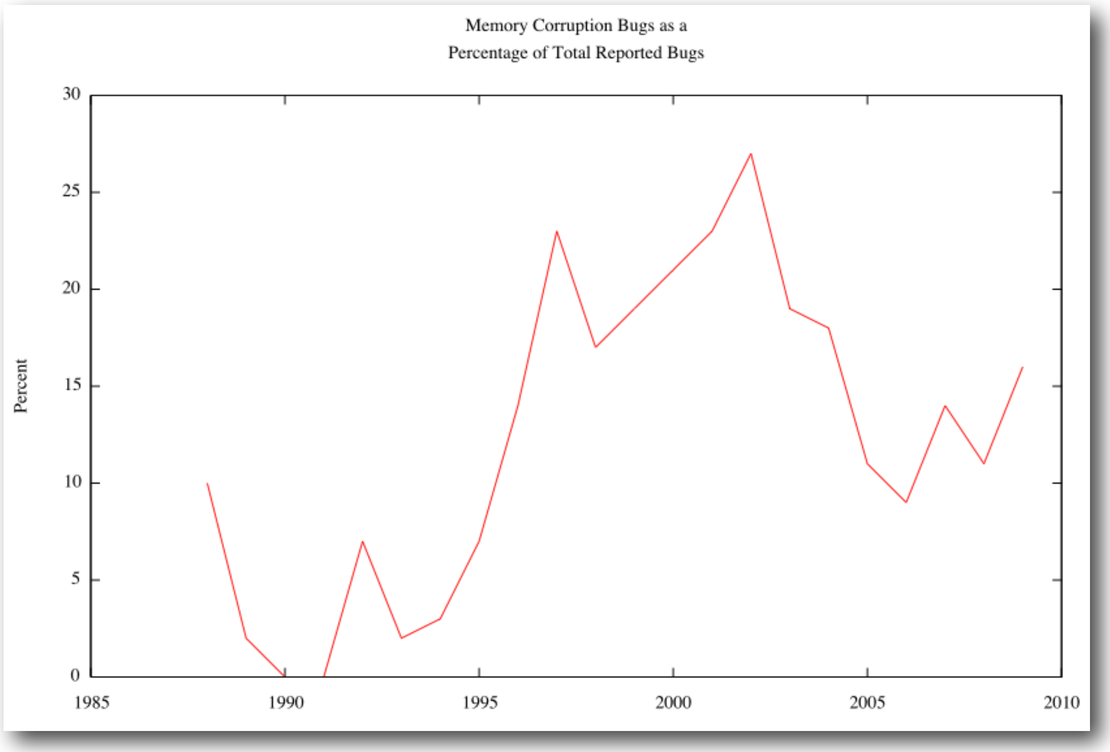


Figure 2 - Memory Corruption Bugs as a Percentage of Reported Bugs

It is worth noting that while the percentage dipped briefly between 2002 and 2006, the general trend is still upwards and to the right. (It is also worth noting that classes of bugs like Cross Site Scripting, which were not previously considered, and are incredibly easy to discover en masse now litter the databases.) Maintaining a positive growth rate in the face of such competition indicates that the bug class is indeed going nowhere in the very short term.

A closer examination of the OSVDB data is in order (and is currently being done), to determine splits of bug classes per vendor and one can safely intuit that for vendors we care about the results will speak even more in favor of memory corruption attacks as being a dominant attack vector. (This will for example focus on the major vendors like Microsoft and Apple while excluding vendors like Bob's PHP Gallery).

The Wiki-Timeline

Since the objective was to document memory corruption attacks and generic mitigation techniques, a timeline seemed to make perfect sense. An early attempt made it clear that we under estimated the scope of the project.

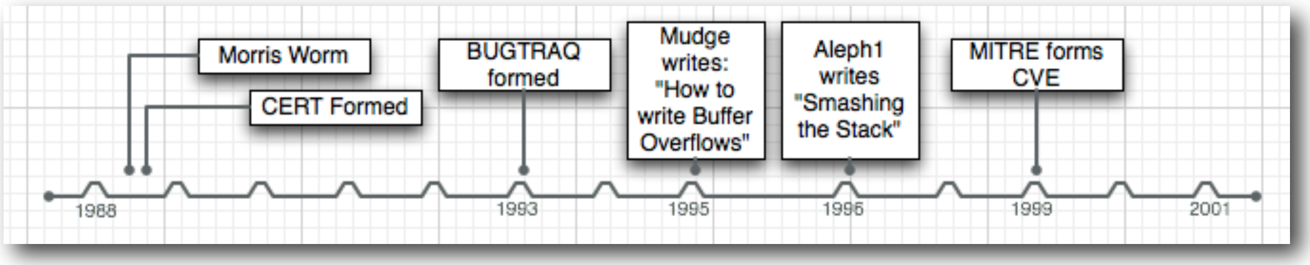
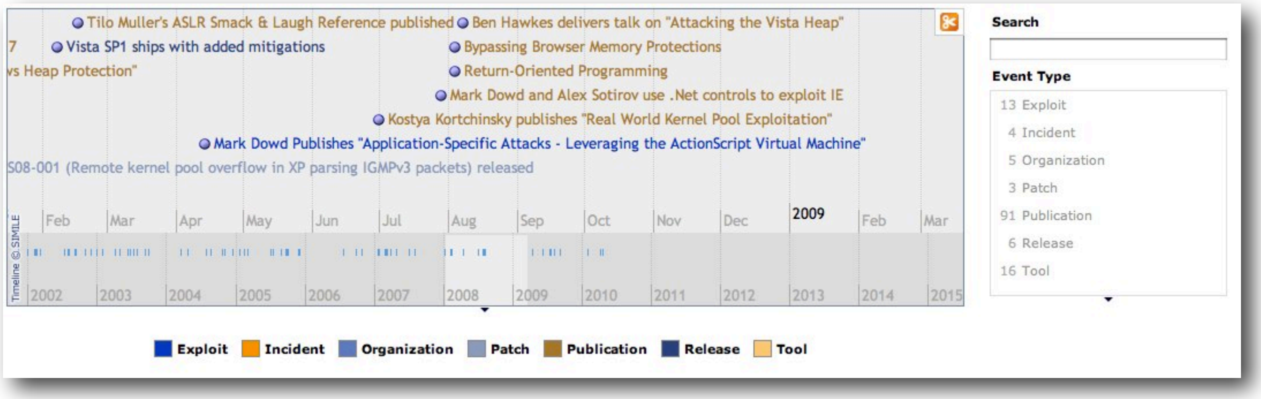

Morris Worm
CERT Formed
BUGTRAQ formed
Mudge writes: "How to write Buffer Overflows"
Aleph1 writes "Smashing the Stack"
MITRE forms CVE
1988
1993
1995
1996
1999
2001

Figure 3 - Early Version of the Timeline

As the number of events grew, it became increasingly obvious that a slightly better solution would be required.

By making use of the awesome *Simile* project from MIT (http://simile.mit.edu/) (and a ghetto Google Docs back-end) we were able to throw together a quick web page to visualize our timeline online (http://ilm.thinkst.com/folklore/)



● Tilo Muller's ASLR Smack & Laugh Reference published ● Ben Hawkes delivers talk on "Attacking the Vista Heap"
7 ● Vista SP1 ships with added mitigations ● Bypassing Browser Memory Protections
vs Heap Protection" ● Return-Oriented Programming
● Mark Dowd and Alex Sotirov use .Net controls to exploit IE
● Kostya Kortchinsky publishes "Real World Kernel Pool Exploitation"
● Mark Dowd Publishes "Application-Specific Attacks - Leveraging the ActionScript Virtual Machine"
S08-001 (Remote kernel pool overflow in XP parsing IGMPv3 packets) released

Search

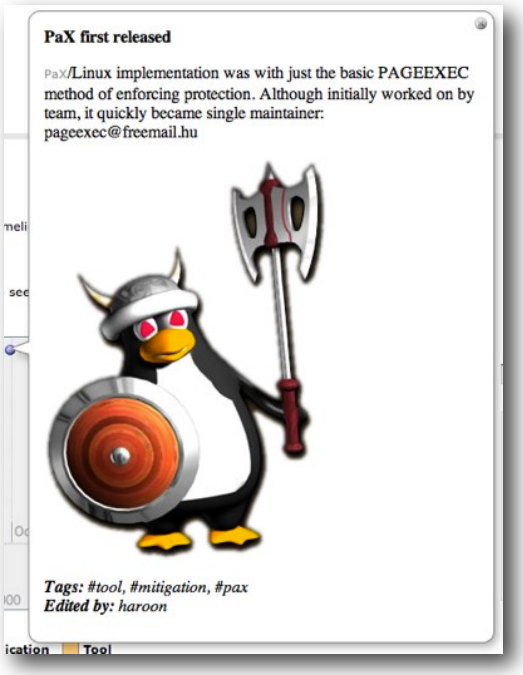
Event Type

13 Exploit
4 Incident
5 Organization
3 Patch
91 Publication
6 Release
16 Tool

Exploit Incident Organization Patch Publication Release Tool

Figure 4 - The timeline on http://ilm.thinkst.com/folklore

This allows us to capture event details fairly richly, and allows site visitors to upload events.

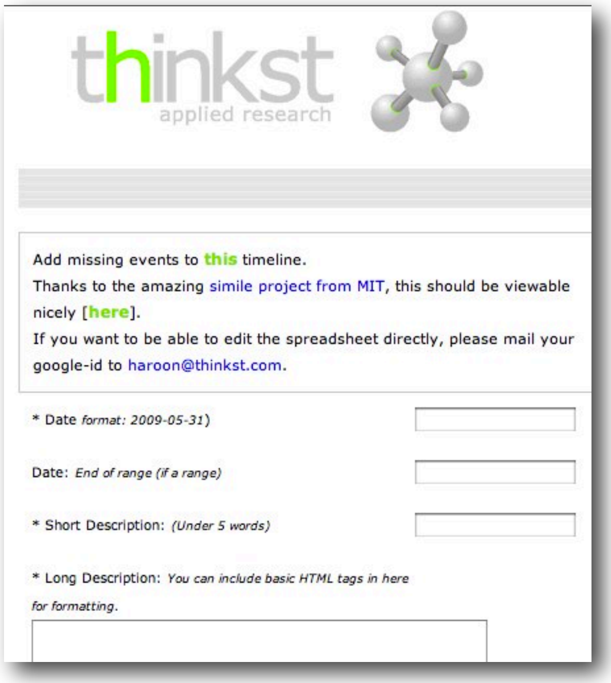


PaX first released

PaX/Linux implementation was with just the basic PAGEEXEC method of enforcing protection. Although initially worked on by team, it quickly became single maintainer: pageexec@freemail.hu

Tags: *#tool, #mitigation, #pax*
Edited by: *haroon*

Figure 5 - Rich Info on timeline



Add missing events to **this** timeline.
Thanks to the amazing simile project from MIT, this should be viewable nicely [**here**].
If you want to be able to edit the spreadsheet directly, please mail your google-id to haroon@thinkst.com.

* Date *format: 2009-05-31)*	
Date: *End of range (if a range)*	
* Short Description: *(Under 5 words)*	

* Long Description: *You can include basic HTML tags in here for formatting.*

Figure 6 - Add Events

The final version of this application will remain similar in principle but will allow for edits and will include other CRUD features. The timeline also allows us to store a local copy of referenced papers and posts which can be accessed via a perma-link. An example of its usage can be seen in the references section of this document.

At the time of this paper, the timeline hosts just under 150 events spanning from 1972 to the current day.

Timeline

What follows is the timeline of attacks and mitigations generated from the website, but edited somewhat to add detail where appropriate.

Key: *Attacks* 🟩 *Defense* 🟥

<u>10/31/1972 - **The First documented Overflow Attack**</u>

The Computer Security Technology Planning Study [1] compiled to present computer security requirements for the US Air Force documents the earliest known memory corruption attacks.

While discussing a program that handled pointers, it reads: *"By supplying addresses outside the space allocated to the users programs, it is often possible to get the monitor to obtain unauthorized data for that user, or at the very least, generate a set of conditions in the monitor that causes a system crash."*

It goes on to describe another operating system where *"the code performing this function does not check the source and destination addresses properly, permitting portions of the monitor to be overlaid by the user. This can be used to inject code into the monitor that will permit the user to seize control of the machine."*

Figure 7 - Air Force Study

<u>11/17/1985 - **Phrack Magazine 0x01 Published**</u>

<u>11/2/1988 - **The Morris Worm**</u>

*Figure 8
Robert Morris*

Robert Tappan Morris (Jr.) wrote and released the "Morris Worm" while still a student at Cornell University. Aside from being the first computer worm to be distributed via the Internet, the worm was the publics introduction to "Buffer Overflow Attacks", as one of the worms attack vectors was a classic stack smash against the fingerd daemon.

In his analysis of the worm, Eugene Spafford [2] writes the following: *"The bug exploited to break fingerd involved overrunning the buffer the daemon used for input. The standard C library has a few routines that read input without checking for bounds on the buffer involved. In particular, the gets call takes input to a buffer without doing any bounds checking; this was the call exploited by the Worm."*

In his email message to researcher Ben Hawkes, Morris recounts his thoughts on the overflow attack [3]: *"I had heard of the potential for exploits via overflow of the data-segment buffers overwriting the next variable. That is, people were worried about code like this:*

```
char buf[512];
int is_authorized;
main(){
      ...;
      gets(buf);
```

The idea of using buffer overflow to inject code into a program and cause it to jump to that code occured to me while reading fingerd.c"

<u>11/30/1988</u>
<u>CERT Formed</u>

<u>1/23/1989</u>
<u>“ZARDOZ Security Digest” Published</u>

<u>CERT Advisory for overflow in 4.3BSD bin/passwd.c</u> - <u>01/31/1989</u>
CERT published CA-1989-01 to document an overflow in passwd.c reported by Keith Bostic of UC-Berkley.

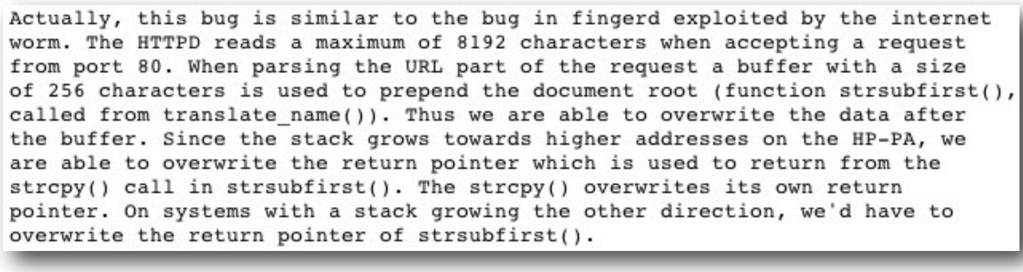
<u>6/23/1990</u>
<u>ZARDOZ becomes the Core Sec Mail list.</u>

<u>12/30/1990</u> - **<u>An empirical study of the reliability of UNIX Utilities</u>**
Barton Miller (et al) publish “An empirical study of the reliability of the UNIX Utilities in the ACM [4].” With relatively simply (by todays standards) fuzzing, they were "able to crash 25-33% of the utility programs on any version of UNIX that was tested".

<u>9/5/1993</u>
<u>BUGTRAQ Formed</u>

<u>9/30/1993</u>
<u>ISS Scanner Released.</u>

<u>Overflow in NCSA httpd</u> - <u>2/13/1995</u>
Thomas Lopatic made a posting to Bugtraq to report an overflow vulnerability in NCSA httpd 1.3. His posting clearly walked through the steps needed for successful exploitation and included an exploit that creates a file named ‘GOTCHA’ in the /tmp directory [5].



```
Actually, this bug is similar to the bug in fingerd exploited by the internet
worm. The HTTPD reads a maximum of 8192 characters when accepting a request
from port 80. When parsing the URL part of the request a buffer with a size
of 256 characters is used to prepend the document root (function strsubfirst(),
called from translate_name()). Thus we are able to overwrite the data after
the buffer. Since the stack grows towards higher addresses on the HP-PA, we
are able to overwrite the return pointer which is used to return from the
strcpy() call in strsubfirst(). The strcpy() overwrites its own return
pointer. On systems with a stack growing the other direction, we'd have to
overwrite the return pointer of strsubfirst().
```

Figure 09 - Lopatic’s post to Bugtraq

<u>10/19/1995</u>
<u>CERT publishes vulnerability in syslogd</u>

<u>3/4/1995</u>
<u>SATAN Released</u>

<u>10/20/1995 - **"How to Write Buffer Overflows"**</u>
Although only a private / internal release, Mudge (mudge@l0pht.com) published his document titled "How to write Buffer Overflows".[6]

Written primarily as a set of notes to himself, the document covers both the basics of the overflow and a rough introduction to writing shellcode. The document included an exploit for the syslogd [7] bug made public earlier.

Figure 10
Peiter Zatko (Mudge)

<u>12/3/1995 - **splitvt exploit published**</u>
DaveG and VicM of Avalon Research published an advisory (and exploit) for splitvt on Linux 2-3.x. The vulnerability was due to an unbounded sprintf call, which was exploited by an over long HOME environment variable.

<u>11/8/1996 - **Smashing the Stack Published**</u>

Figure 11
Elias Levy (Aleph One)

Aleph1 published what would become the most referenced paper on memory corruption attacks in Phrack49. [8]
From his introduction:
`smash the stack` [C programming] n. On many C implementations it is possible to corrupt the execution stack by writing past the end of an array declared auto in a routine. Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind.

<u>1/20/97</u>
<u>Stack Smashing Defenses Discussed</u>
Bugtraq hosts a discussion on defenses against stack smashing

<u>3/21/1997</u>
<u>Superprobe Exploit Published</u>
solar designer overwrites function pointers to hijack execution flow

<u>4/22/1997</u>
<u>DNS Poisoning QID Prediction</u>
<u>CORE and SNI report possible overflows due to bind ignoring MAXHOSTNAMELEN</u>

Figure 12 - Alexander Peslyak (Solar Designer)

<u>**Bypassing the non-exec Stack (ret-2-libc) -** 8/10/1997</u>
Solar Designer published the first known return-to-libc attack to overcome his own non-executable stack patch [9].
He demonstrated the technique using the lpr exploit against a Linux system with his non executable stack patch. His patch (for his patch) ensured that shared libraries are mmaped into regions containing a null byte to retard their use with unsafe string functions.

9/1/1997
Nmap - Art of Portscanning
NMap is released in Phrack 51

12/18/1997 - **StackGuard Announced**

Figure 13
Crispin Cowan

Crispin Cowan announced StackGuard on Bugtraq with a link to his pre-released USENIX paper titled: *"StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks".* [10] Their invention was a GCC patch that makes use of a "canary" DWORD on the stack in front of the return address to determine if the return address had been modified.

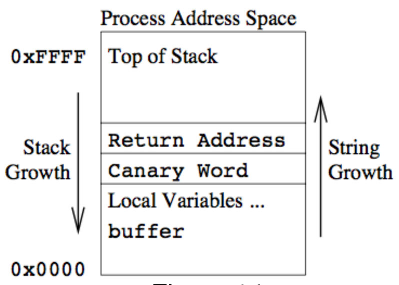


Figure 14
Canary on Stack

12/19/1997 - **StackGuard bypasses discussed**
Tim Newsham posts to Bugtraq [11] with possible weaknesses with StackGuard. He covers attacking local variables and possible leaks of the Canary value.

1/14/1998 - **IE4 Heap Overflow**
DilDog published his exploit and advisory for the mk:// exploit on Internet Explorer 4. The bug was a heap based overflow with the exploit code written for Windows 95 [12].

Figure 15
Christien Rioux (dildog)

4/16/1998 - **"The Tao of Windows Buffer Overflow"**
DilDog published his document on how to write Windows based Overflow exploits [13].
"The Tao" becomes the template that Win32 tutorials are based on for many a coming year.

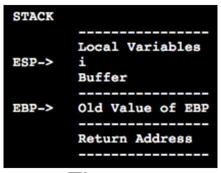


Figure 16
Stack Drawing from "The Tao"

1/31/1999 - **w00w00 on Heap Overflows**
Matt Conover and w00w00 published "w00w00 on Heap Overflows" [14]. The paper credited the previous work done on heap based overflows and documented possible exploitation using sample programs.
A good indication of the understanding of Heap based overflows at the time can be determined by this line from his paper:
"*Some people have actually suggested making a "local" buffer a "static" buffer, as a fix! This not very wise; yet, it is a fairly common misconception of how the heap or bss work.*"

Figure 17
Matt Conover (sh0k)

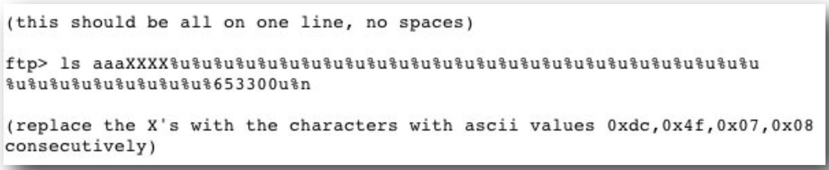
<u>5/1/1999</u>
<u>MITRE forms CVE Initiative</u>

Figure 18 Barnaby Jack (dark spyrit)

<u>Dark Spyrit Win32 Buffer Overflows</u> <u>- 9/9/1999</u>
dark spyrit AKA Barnaby Jack published "Win32 Buffer Overflows (Location, Exploitation and Prevention)" in Phrack 55 [15]. Using a publicly disclosed vulnerability in Seattle Labs Mail Server, he demonstrated how to locate the bug using tracing and debugging, and then walked through the shellcode challenges posed by Win32 systems (covering the use of trampoline calls) .

<u>9/9/1999 -</u> **<u>"The Frame Pointer Overwrite"</u>**
klog published "The Frame Pointer Overwrite" in Phrack 55 [16]. He showed how to gain execution by using a single byte overwrite to overwrite the last byte of %esp. In some situations this can result in the calling function retrieving its saved EIP from an attacker defined location resulting in altered execution flow.

<u>9/20/1999 -</u> **<u>Format String bug in proftpd 1.2.0pre6</u>**
In the first public disclosure of format string bugs, Tymm Twillman posted an email to bugtraq [17] after having notified the proftpd maintainers discussing remote hole in proftpd.



```
(this should be all on one line, no spaces)

ftp> ls aaaXXXX%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u
%u%u%u%u%u%u%u%u%u%653300u%n

(replace the X's with the characters with ascii values 0xdc,0x4f,0x07,0x08
consecutively)
```

Figure 19 - Format String Attack against proftpd

His POC pointed at the future of format string vulnerabilities.

<u>10/21/1999 -</u> **<u>"Advanced Buffer Overflow Exploits"</u>**
Taeh Oh publishes "Advanced Buffer Overflow Exploit" on advanced buffer overflows [18]. It describes techniques to bypass filtering restrictions, bypassing seteuid(getuid()) and breaking out of chroots.

<u>5/1/2000 -</u> **<u>Bypassing StackGuard and StackShield"</u>**
In Phrack #56, Bulba and Kil3r publish techniques to bypass StackGuard and StackShield (ab)using adjacent pointers on the stack. (Uses GOT overwrite for reliable exploitation) [19]

<u>5/1/2000 -</u> **<u>Smashing C++ VPTRS</u>**
In Phrack #56 rix published "Smashing C++ VPTRS" showing how overflows in C++ code can be used to overwrite VPTRS and VTABLEs to hijack execution flow [20].

5/1/2000 - **Exploiting non-terminated adjacent memory spaces**
In Phrack #56 twitch@vicar.org published "Exploiting Non-adjacent Memory Spaces". twitch showed the possibility of exploiting non terminated strings in adjacent memory spaces by (ab)using functions traditionally cited as safer alternatives to *strcpy* [21].
From his paper:
"The essence of the issue is that many functions that a programmer may take to be safe and/or 'magic bullets' against buffer overflows do not automatically terminate strings/ buffers with a NULL. That in actuality, the buffer size argument provided to these functions is an absolute size- not the size of the string."

6/24/2000 - **WuFTPD: Providing *remote* root since at least 1994**
The tf8 Wu-Ftpd remote format string exploit was released publicly bringing format string exploitation into the public eye [22].

6/24/2000 - **Format Bugs: What are they. Where they come from?**
Lamagra Argamal posts to bugtraq (linking to the previous WuFTPD thread), releasing his mini-paper titled "Format Bugs: What are they, Where did they come from, How to exploit them" [23].
His 200 line mini-paper, covered the essence of the bug class (both reading and writing to arbitrary memory using format string errors), included sample code and even pointed towards format string 0day in popular ftp servers.

7/25/2000 - **JPEG Com Marker vulnerability in Netscape**
Solar Designer makes a fairly low key posting titled "*JPEG COM Marker Processing Vulnerability in Netscape Browsers". *to the mailing lists [24]. The write-up of the JPEG bug is itself instructive (and is the harbinger of file format bugs that still plague us today), but much more interestingly he goes on to explain the generic method of gaining code execution through free() (and through unlink()) with heap based overflows.

Figure 20
Solar's "evil" image

9/9/2000 - **Format String Attacks**
Tim Newsham releases his paper on "Format String Attacks" on bugtraq [25]. The paper was the most comprehensive overview of the nature of the problem, and its exploitation and was worth reading for the introduction alone:
"*I know it has happened to you. It has happened to all of us, at one point or another. You're at a trendy dinner party, and amidst the frenzied voices of your companions you hear the words "format string attack"*.

Figure 21
Newsham's Paper

10/1/2000 - **PaX first Released**

PaX (a security patch which provides non-executable memory pages and full address space layout randomization (ASLR) for a wide variety of architectures) was first released for the Linux kernel
It was initially released with just the basic PAGEEXEC method of implementing non executable pages.
Although initially worked on by a team, it quickly became a single maintainer project: pageexec@freemail.hu

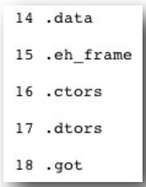
Figure 22
PaX Logo

11/30/2000 - **PaX adds MPROTECT**

PaX adds MPROTECT support to prevent the introduction of new executable code into a tasks address space. This is accomplished by restricting access to mmap() and mprotect() interfaces.

12/12/2000 - **Overwriting the .dtors section**

Juan M. Bello Rivas publishes "Overwriting the .dtors section" [26]. The paper explained how to make use of .ctors and .dtors as a means of gaining code execution. These sections are mapped into elf executables (by gcc) containing a list of functions to be executed before main() (.ctors / constructors) or after main() (.dtors / destructors).


```
14 .data
15 .eh_frame
16 .ctors
17 .dtors
18 .got
```
Figure 23
ELF Sections

2/18/2000 - **grsecurity released**

grsecurity (developed by Brad Spengler) was released initially as a port of Solar Designer's Openwall patch to the 2.4 series of Linux kernels.
It soon discovered the PaX project and the developers of the two projects have been working closely together ever since. grsecurity provides complementary protections that are outside of the scope of the PaX project. The project has expanded far beyond its humble roots and pioneered numerous protection mechanisms.

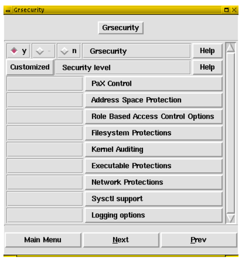


Figure 24
grsecurity config

6/18/2001 - **IIS .ida ISAPI filter Vulnerability**

eEye published an advisory for MS01-033, an overflow in the Index ISAPI extension. The exploit makes use of a heap spray to get execution on NT4 and Windows2000

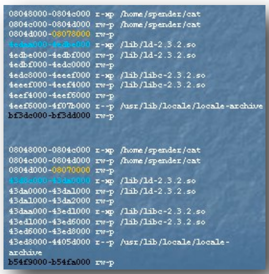
Figure 25 - eEye

7/13/2001 - **Code Red Worm in the Wild**

The Code Red worm was observed on the Internet. Discovered and researched by Marc Maiffret and Ryan Permeh of eEye Digital Security. (who named the worm). Interestingly, the worm made use of an overwritten SEH handler on the stack, a technique that became popular only a little while later.

Figure 26
Code Red

<u>7/31/2001 - **PaX introduces ASLR**</u>
PaX introduced Address Space Layout Randomization.
From the PaX aslr.txt document:
"The goal of Address Space Layout Randomization is to introduce randomness into addresses used by a given task. This will make a class of exploit techniques fail with a quantifiable probability and also allow their detection since failed attempts will most likely crash the attacked task."



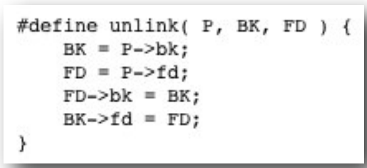
```
08048000-0804c000 r-xp /home/spender/cat
0804c000-0804d000 rw-p /home/spender/cat
0804d000-08078000 rw-p
4edaa000-4edbe000 r-xp /lib/ld-2.3.2.so
4edbe000-4edbf000 rw-p /lib/ld-2.3.2.so
4edbf000-4edc0000 rw-p
4edc8000-4eeef000 r-xp /lib/libc-2.3.2.so
4eeef000-4eef4000 rw-p /lib/libc-2.3.2.so
4eef4000-4eef6000 rw-p
4eef6000-4f07b000 r--p /usr/lib/locale/locale-archive
bf3dc000-bf3dd000 rw-p


08048000-0804c000 r-xp /home/spender/cat
0804c000-0804d000 rw-p /home/spender/cat
0804d000-08070000 rw-p
43d8c000-43da0000 r-xp /lib/ld-2.3.2.so
43da0000-43da1000 rw-p /lib/ld-2.3.2.so
43da1000-43da2000 rw-p
43daa000-43ed1000 r-xp /lib/libc-2.3.2.so
43ed1000-43ed6000 rw-p /lib/libc-2.3.2.so
43ed6000-43ed8000 rw-p
43ed8000-4405d000 r--p /usr/lib/locale/locale-
archive
b54f9000-b54fa000 rw-p
```

Figure 27
Two runs of an executable with ASLR

<u>8/13/2001</u>
<u>**StackGhost released**</u>
StackGhost presented at USENIX as a kernel modification to guard application return pointers [27]

<u>8/13/2001</u>
<u>**FormatGuard released**</u>
Crispin Cowan et al published:"FormatGuard: Automatic Protection From printf Format String Vulnerabilities" [28]

<u>**VUDO malloc tricks -** 11/8/2001</u>



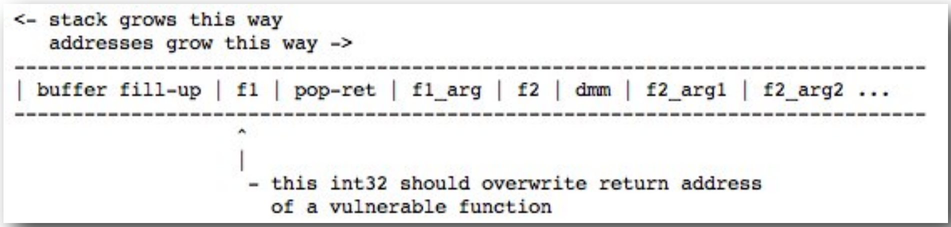
```
#define unlink( P, BK, FD ) {
    BK = P->bk;
    FD = P->fd;
    FD->bk = BK;
    BK->fd = FD;
}
```

Figure 28
Unlink macro

MaXX (Michel Kaempf) published Vudo Malloc Tricks in Phrack 57 [29]. The paper could have been sub-titled "How to smash the Heap for fun and profit". The paper documented techniques against libc's native Doug Lee's malloc and demonstrated the generic unlink() write4 technique against the published vulnerability in sudo-1.6.1-1. MaXX's article went on however to document the DLmalloc allocator in great detail.

<u>**Once upon a free()** - 11/8/2001</u>
In the very same issue of Phrack (57), anonymous wrote "Once upon a free()" as a gentle introduction to Heap based overflows using solar's unlink() technique [30].

<u>**Advanced return-2-libc -** 12/28/2001</u>
Nergal published Advanced return-into-lib(c) exploits (PaX case study) in Phrack 58. The article covered standard and advanced ret-2-libc attacks and included ret-2-libc chaining with retpop [31].



```
<- stack grows this way
   addresses grow this way ->
------------------------------------------------------------------------------
| buffer fill-up | f1 | pop-ret | f1_arg | f2 | dmm | f2_arg1 | f2_arg2 ...
------------------------------------------------------------------------------
                   ^
                   |
                   - this int32 should overwrite return address
                     of a vulnerable function
```

Figure 29 - nergals ret-into-libc

The 2nd half of the article focused on bypassing PaX with ret-2-libc

Figure 30 - Dave Aitel

Advantages of block based binary analysis - 2/4/2002

Dave Aitel publishes "The Advantages of Block-Based Protocol Analysis for Security Testing". The paper documented SPIKE his block based fuzzer and to some extent revived interest in fuzzing.

2/7/2002 - Third Generation Exploits

Halvar Flake presented "*Third Generation Exploits on NT/Win2k Platforms*" [32].
He covered the 4 byte write anywhere HEAP unlink attack and documented using SEH on windows as an attack vector.

Figure 31 - Thomas Dullien (Halvar Flake)

2/13/2002 - **Visual C++ adds /GS compiler protection**

Microsoft released their /GS (Buffer Security Check) with Visual C++ 7. The complier option places a cookie before the return address and calls __security_check_cookie during a function epilogue to detect stack corruption. Although /GS bears an uncanny resemblance to Crispin Cowan's StackGuard, Microsoft claims to have invented it independently. (Their claim is backed up below).

2/14/2002 - **Published flaw in /GS**

Cigital release a technical paper on Microsofts /GS compiler option and call it a "vulnerability seeder" [33]. They document a flaw with adjacent variables/parameter over writes and contrast it with StackGuard. Interestingly, their paper states:
"*The kinds of attack that Cigital made use of to defeat the Microsoft mechanism are neither novel nor do they require exceptional expertise. Had Microsoft studied the literature surrounding StackGuard, they would have been aware of the existence of such attacks.*"

3/5/2002 - **"Non-Stack based exploitation"**

David Litchfield (mnemonix) published "*Non-stack Based Exploitation of Buffer Overrun Vulnerabilities on Windows NT/2000/XP*" which essentially documents ret-2-libc style attacks on Win32 [34].

Figure 32 - David Litchfield (mnemonix)

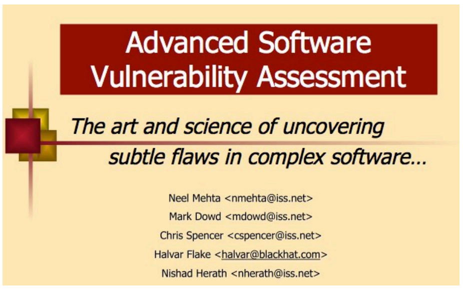
7/28/2002 - **Bypassing PaX ASLR Protection**

Tyler Durden publishes "*Bypassing PaX ASLR Protection*" in Phrack #59 [35].
He demonstrates using an information leak through a partial overwrite to obtain information on the running programs address space and discusses ***ret-to-ouput*** as an info leak vector.

7/28/2002 - **Advances in Format String Exploitation**
In Phrack #59, riq and gera publish "Advances in Format String Exploitation" [36].
From the article itself: *"it focused on "different tiny tricks that may help speeding up bruteforcing when exploiting format strings bugs, and ... about exploting (sic) heap based format strings bugs"*

*Figure 33
Gerardo Richarte (gera)*

7/30/2002 - **Integer Overflows Introduced to Public**
During the talk "*Professional Source Code Auditing*", the group consisting of Mark Dowd, Chris Spencer, Neel Metha, Nishad Herath and Halvar Flake discuss integer overflows publicly for the first time. Several examples are given that include the now infamous pre-malloc multiplications and roundup issues.


Advanced Software Vulnerability Assessment

The art and science of uncovering subtle flaws in complex software...

Neel Mehta <nmehta@iss.net>
Mark Dowd <mdowd@iss.net>
Chris Spencer <cspencer@iss.net>
Halvar Flake <halvar@blackhat.com>
Nishad Herath <nherath@iss.net>

Figure 34 - Title Slide from "Advanced Software Vuln Assessment"

7/31/2002
PaX Advances
PaX introduces non-relocatable executable file randomization and vma mirroring.

8/1/2002
Syscall Proxying
Maximiliano Cáceres of CORE publishes "Syscall Proxying - Simulating Remote Execution" [37]

8/3/2002
grsecurity gets Learning Mode
A learning mode was added automatically generates policies for individual subjects by exercising normal system behavior

4 tricks to bypass StackGuard and StackShield - 9/4/2002
Gera (Gerardo Richarte) published "*Four different tricks to bypass StackShield and StackGuard protection*" [38]. The paper focused on bypassing compiler level protections StackGuard, StackShield and Microsoft's /GS.

Slapper worm targets Apache/mod_SSL - 9/13/2002
The Slapper Worm was discovered in the wild exploiting a previously disclosed bug in OpenSSL which was released in July. 2002.
The worm was interesting in that it made use of a memory leak to reliably exploit a remote heap overflow. *(Picture from Peter Szors awesome writeup [39])*

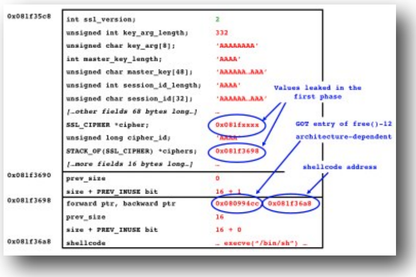

```
0x081f35c8  int ssl_version;                       2
            unsigned int key_arg_length;           332
            unsigned char key_arg[8];              'AAAAAAAA'
            int master_key_length;                 'AAAA'
            unsigned char master_key[48];          'AAAAAA…AAA'
            unsigned int session_id_length;        'AAAA'
            unsigned char session_id[32];          'AAAAAA…AAA'
            [...other fields 68 bytes long...]     …
            SSL_CIPHER *cipher;                    0x081fxxxx
            unsigned long cipher_id;               'AAAA'
            STACK_OF(SSL_CIPHER) *ciphers;         0x081f3698
            [...more fields 16 bytes long...]      …
0x081f3690  prev_size                              0
            size + PREV_INUSE bit                  16 + 1
0x081f3698  forward ptr, backward ptr              0x080994cc  0x081f36a8
            prev_size                              16
            size + PREV_INUSE bit                  16 + 0
0x081f36a8  shellcode                              … execve("/bin/sh") …
```

Figure 35 - Memory leaked through SSL Handshake

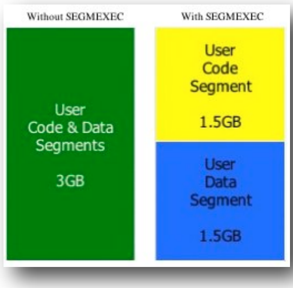


Figure 36
PaX SEGMEXEC

PaX releases SEGMEXEC - 10/31/2002

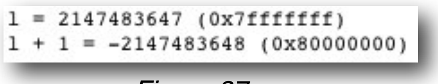
PaX introduces SEGMEXEC to implement the non-executable page feature by using the processors segmentation logic. SEGMEXEC splits the address space with the bottom half for data and the top half for code, effectively incurring no performance hit.

PaX releases kernel stack randomization - 10/31/2002

PaX introduces RANDKSTACK to introduce randomness into the kernel stack.

grsecurity adds /dev/mem and /dev/kmem protection - 12/1/2002

A feature preventing writes to kernel memory through /dev/mem and /dev/kmem, while allowing legitimate writes to certain ranges by XFree86 and others, was added. This was developed mainly in response to sd and devik's Phrack 58 paper on runtime kernel patching [40] (though the techniques of /dev/[k]mem abuse go back as far as Silvio Cesare's work in November 1998 available at http://vxheavens.com/lib/vsc07.html [41]).



```
1 = 2147483647 (0x7fffffff)
1 + 1 = -2147483648 (0x80000000)
```

Figure 37
Integer Wrapping from article

"Basic Integer Overflows" published - 12/28/2002

blexim publishes "Basic Integer Overflows" in Phrack60 [42].

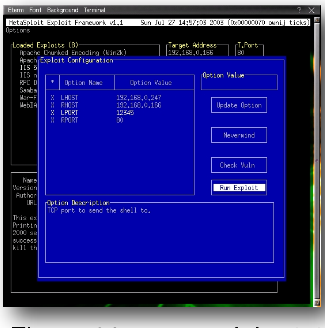
He discussed both widthness and arithmetic overflows.

"pax-future.txt" released - 1/26/2003

This often-overlooked document detailed many attacks and preventions that were either rediscovered years later, or not yet realized in a software implementation. The document was meant to describe the current weaknesses in a system implementing NX/ASLR properly and their future solutions. The design notes in section 1 categorizes all attacks against NX/ASLR that have been published and have yet to be published.
Section c.1 discusses variants of return to code techniques (including jmp instructions). Several methods of preventing all forms of these techniques were described. It also first introduced (in section b.1) the concept of what would later be called KERNEXEC [43].

grsecurity adds RBAC - 4/6/2003

In grsecurity's 2.0 release, a Role Based Access Control (RBAC) system was added. The previous ACL system had subject/object abstractions, and the new RBAC system added a user/group/special role hierarchy behind those abstractions. It was a complete rewrite on both the kernel and user-land end.



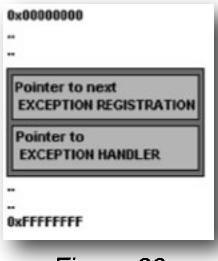
```
Eterm  Font  Background  Terminal
MetaSploit Exploit Framework v1.1      Sun Jul 27 14:57:03 2003 (0x00000070 ownij ticks)
Options

Loaded Exploits (8)                               Target Address    T.Port
  Apache Chunked Encoding (Win2k)                 192.168.0.166     80
  Apach  Exploit Configuration
  IIS 5
  IIS n    *  Option Name    Option Value          Option Value
  RPC D
  Samba    X  LHOST          192.168.0.247          Update Option
  War-F    X  RHOST          192.168.0.166
  WebDA    X  LPORT          12345                  Nevermind
           X  RPORT          80
                                                    Check Vuln

   Name                                             Run Exploit
Version
 Author    Option Description
    URL    TCP port to send the shell to.

This ex
Printin
2000 se
success
kill th
```

Figure 38 - metasploit v1.1

<u>Metasploit.com opened to the public -</u> <u>6/14/2003</u>
The metasploit.com website goes live.
The open-source, free framework is written in Perl and aims to fill the niche occupied by paid for tools like Immunity CANVAS and CORE Impact.
Version 1.0 was released in October with 11 exploits and an ncurses based interface.

<u>4/30/2003</u>
<u>PaX introduces non-executable kernel pages</u>

<u>7/10/2003 -</u> **<u>"Variations in Exploit methods between Linux and Windows" published</u>**
David Litchfield publishes "Variations in Exploit methods between Linux and Windows" [44].
He references the "long known technique" of overwriting the SEH record on the stack on Windows (which is actually the first known documentation of the attack technique)


0x00000000
..
..
Pointer to next EXCEPTION REGISTRATION
Pointer to EXCEPTION HANDLER
..
..
0xFFFFFFFF
Figure 39
Exception Handlers on the Stack

<u>8/2/2003 -</u> **<u>"Win32 device drivers communication vulnerabilities" published</u>**
Sec-Labs team releases paper regarding memory corruption in windows device drivers and an exploit for Norton Antivirus

<u>9/8/2003 -</u> **<u>"Defeating the Stack Based Buffer Overflow Prevention Mechanism of MS Windows 2003 Server" Published</u>**
David Litchfield publishes "Defeating the Stack Based Buffer Overflow Prevention Mechanism of MS Windows 2003 Server" [45].
He bypassed registered SEH protection by noticing that unregistered exception handlers that are out of the range of loaded DLLS were considered valid.

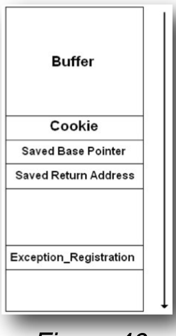


Figure 40
Exception Handlers on the Stack

<u>9/30/2003 -</u> **<u>/SAFESEH introduced into Visual Studio</u>**
Microsoft introduced /SAFESEH in Visual Studio 2003 in an attempt to limit exploitation through overwritten SEH handlers.

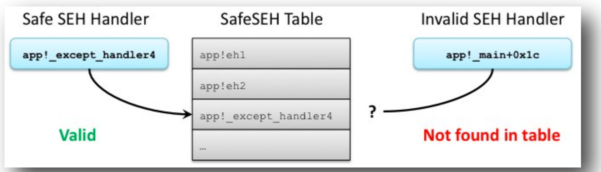


Figure 41 - /SAFESEH introduced in VS2003

/GS now moves buffers to prevent local overwrites too

10/2/2003
MOSDEF Released
Immunitysec releases MOSDEF, a
100% Python re-targetable compiler for C->shellcode

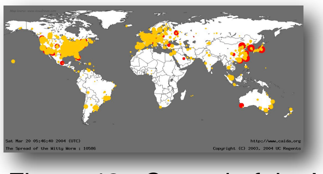


Figure 42 - Spread of the Witty Worm (caida.org)

Witty Worm in the Wild - 3/19/2004
An Internet worm that exploits a vulnerability in software from a specific security vendor (ISS) propagates.

Figure 43 - Title Slide for "Reliable Windows Heap Exploits"

"Reliable Windows Heap Exploits" Presented - 4/21/2004
Matt Conover (sh0k) and Oded Horovitz present "*Reliable Windows Heap Exploits*" at CansecWest [46].
They give an overview of reliable heap exploits pre-XP-SP2 and discuss the new challenges posed by XP-SP2 for reliable heap exploitation.

"Windows Heap Overflows" Presented - 7/28/2004
David Litchfield presents "*Windows Heap Overflows*" at BlackHat USA 2004 [47]. He covers 4 byte write anywhere on free attack and discusses heap repair and targets for overwrite

XP-SP2 Ships - 8/25/2004

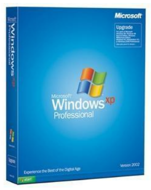


Figure 44 Windows XP

Microsoft ships XP-SP2. Windows itself is now compiled with /GS and /SAFESEH.
DEP is supported by hardware and or software. Heap cookie validation and Safe Unlinking is now included.
PEB and TEB are randomized.
Pointers are encoded. Both the Stack and Heap are marked non executable.

10/25/2004 - **"On the effectiveness of ASLR" published**
Shacham et al Publish "*On the effectiveness of address-space randomization*" at the 11th ACM Conference on Computer and Communications Security [48].
They showed that the utility of ASLR on 32-bit platforms is limited due to the number of bits available for randomization, and demonstrate an attack that is able to use brute force to determine the layout of the stack (by returning into sleep()) before launching a full return-2-libc exploit on systems with PaX or W^X.

Figure 45
Barend-Jan Wever (skylined)

"Heap Spraying" against Internet Explorer is demonstrated - 11/2/2004
Barend-Jan Wever (SkyLined) released the exploit for ned's Internet Explorer IFRAME src&name parameter overflow. His exploit included "Internet Exploiter" which made use of "Heap Spraying" as an attack vector.

12/17/2004 - **Unsafe unlinking of the lookaside list is exploited**
Matt Conover (sh0k) & Oded Horovitz demo the Unsafe unlinking of the lookaside list at SyScan [49].
They exploit the lack of the unlink check on the lookaside to hijack execution flow.

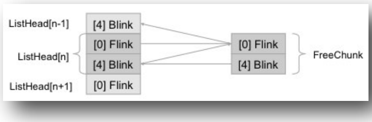


Figure 46 - heap chunks

1/21/2005 - **"Defeating Microsoft Windows XP SP2 Heap protection and DEP bypass" published**
Alexander Anisimov publishes "*Defeating Microsoft Windows XP SP2 Heap protection and DEP bypass*" [50].
He abuses the non existent checks on the look-aside list to gain execution

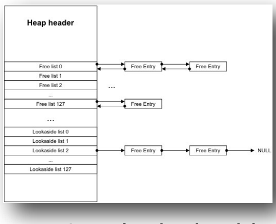
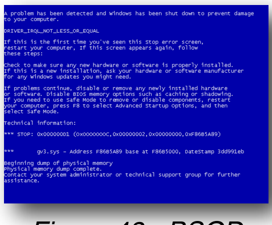


Figure 47 - the lookaside list

2/17/2005 - **"Remote Windows Kernel Exploitation" published**
Barnaby Jack publishes his paper titled "*Remote Windows Kernel Exploitation, Step into the Ring 0*" [51].He uses for purposes of illustration a previously disclosed bug in the Symantec line of personal firewalls. His sample shellcode is a remote keystroke logger and a kernel loader that allows one to plug-in and run any userland shellcode.



```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)


***       gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
```

Figure 48 - BSOD

3/5/2005 - **PaX privilege escalation bug published**
The author of PaX (pageexec) publishes an escalation of privilege bug in PaX. He calls the bug "a *spectacular fuckup*" that "*pretty much destroys what PaX has always stood and been trusted for*" and tries to leave the project. (he gives an in depth explanation of the bug on the DailyDave Mailing List.)

7/6/2005 - **Process Stalker Released**
Pedram Amini releases Process Stalker, an open source software package to combine run-time profiling, state mapping and tracing.

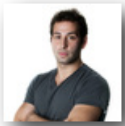


Figure 49
Pedram Amini

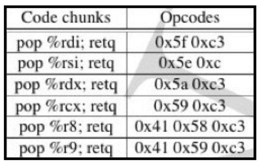
<u>7/20/2005 - "Windows Kernel Pool Overflow Exploitation" published</u>
SoBelt presents on "Windows Kernel Pool Overflow Exploitation" at XCon 2005 [52]. He covered the internals of the kernelpool, and showed its analogies to a traditional heap write4. He targeted the the KiDebugRoutine Pointer which is called by KiDispatchException to gain execution.

<u>8/31/2005 - "Critical Section Heap Exploit Technique" published</u>
Nicolas Falliere of Symantec publishes a paper detailing a new exploitation technique against XP-SP2's heap protection mechanisms [53]. He leverages an overwrite in the critical section doubly linked list.

<u>9/25/2005 - Format string vulnerability published in a Perl Application</u>
Jack Louis publishes format string vulnerability in a Perl Application. The publication sparked discussion on the exploitability of such bugs

Figure 50 - Jack Louis

<u>9/28/2005 - "Borrowed Code Chunk Exploitation Technique" published</u>
Sebastian Krahmer publishes his "*Borrowed Code Chunk Exploitation Technique*" [54] specifically to target x86-64 based systems.
The paper effectively describes what will later become called ROP to bypass DEP/NX/AVP



Code chunks	Opcodes
pop %rdi; retq	0x5f 0xc3
pop %rsi; retq	0x5e 0xc
pop %rdx; retq	0x5a 0xc3
pop %rcx; retq	0x59 0xc3
pop %r8; retq	0x41 0x58 0xc3
pop %r9; retq	0x41 0x59 0xc3

Figure 51
6 important code chunks (for ROP)

<u>10/5/2005 - Technique published to bypass hardware DEP</u>
In Uninformed Journal 2, Matt Miller (skape) and Ken Johnson (skywing) published a technique to bypass hardware enforced DEP by using a ret-2-libc style attack to return into functions that disable DEP. [55]

Figure 52
Matt Miller (skape)

<u>11/30/2005 - Microsoft ships Visual Studio 2005 with GS v2.</u>
Visual Studio 2005 ships with strict GS pragma. /GS is now in version 2 and makes a shadow copy of argument parameters.

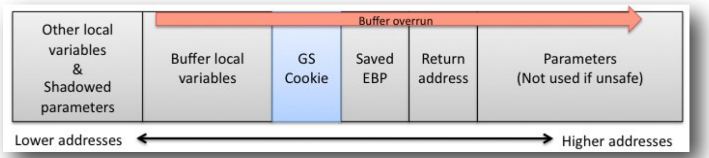


Figure 53 - Improved /GS layout

It also introduces C++ operator::new integer overflow detection [56]

<u>12/1/2005 - Format String integer wrap vulnerability Published</u>
Jack Louis publishes format string integer wrap vulnerability [57]. The bug in Perl's internals required a vulnerable perl program to be exploitable

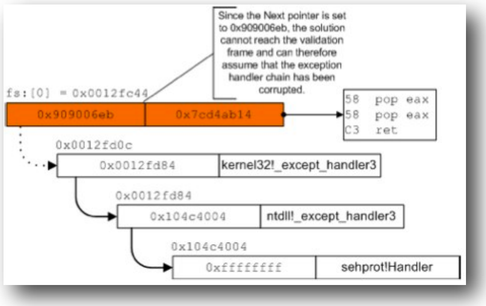
12/2/2005 - **"Format String Vulnerabilities in Perl Programs" published**
Steve Christey publishes "*Format String Vulnerabilities in Perl Programs*" [58]. He covered format string problems in the Perl interpreter and Perl programs, along with a timeline of format string attacks.

12/7/2005 - **Technique published to exploit Freelist[0] on XP-SP2**
Brett Moore publishes technique to exploit overflows in Freelist[0] on XP-SP2 [59]. It takes advantage of the fact that processes continue to run despite detected heap corruption.

Figure 54
Brett Moore
(antic0de)

7/26/2006 - **PaX Team releases UDEREF/x86**
The PaX Team killed NULL pointer dereferences as a class of vulnerabilities (as well as the larger class of invalid userland accesses) before the vulnerability class ever became public/popular. The first Linux kernel NULL pointer dereference published in 2007 was developed by Brad Spengler in 2006 about two weeks after the grsecurity release with UDEREF. The class of NULL ptr dereferences was known to the PaX/grsecurity team as being exploited privately in the kernel which provided the motivation to develop the feature. To get the protection in the hands of users before everyone jumped on the NULL pointer deref bandwagon, no PaX announcement was issued for the feature. The grsecurity announcement was also intentionally vague about how such vulnerabilities are exploited. A thorough write-up on UDEREF was released later

9/30/2006 - **"Preventing the Exploitation of SEH Overwrites" Published**
Matt Miller (skape) publishes the Uninformed Journal article titled *"Preventing the Exploitation of SEH Overwrites"* [60]. The paper (first) gives an excellent overview of SEH based exploitation.
His solution is to place an exception record as the last exception (as a form of exception canary). The exception stack can be walked prior to dispatch to ensure that the "validation frame" can be reached.
The solution kills the attractiveness of pop/pop/ret instructions in SEH based overflows


Since the Next pointer is set to 0x909006eb, the solution cannot reach the validation frame and can therefore assume that the exception handler chain has been corrupted.
fs:[0] = 0x0012fc44
0x909006eb
0x7cd4ab14
58 pop eax
58 pop eax
C3 ret
0x0012fd0c
0x0012fd84
kernel32!_except_handler3
0x0012fd84
0x104c4004
ntdll!_except_handler3
0x104c4004
0xffffffff
sehprot!Handler
Figure 55 - SEH Overwrite Protection

9/30/2006 - **Unusual Bugs Presentation**
Ilja van Sprundel gives a presentation on many published, but not widely known attacks. Most interestingly it brings up NULL dereferences as exploitable and shows a NULL pointer dereference exploit from 1994 [61].

<u>10/31/2006 - **"Memory Retrieval Vulnerabilities" Published**</u>
Derek Soeder of eEye publishes "*Memory Retrieval Vulnerabilities*" [61].
He covers using format strings, non-terminating string functions, uninitialized memory and information leaks to retrieve information from a process

Memory Retrieval
Vulnerabilities

October 2006

Derek Soeder
Senior Software Engineer

eEye Digital Security®

Figure 56 - eEye presentation on memory retrieval vulns.

<u>1/19/2007 - **"Double Free Vulnerabilities" Published**</u>
Matthew Conover (sh0k) publishes "*Double Free Vulnerabilities*" on the Symantec blog describing how double free vulnerabilities can be used on Win32 based systems [62].

Figure 57 Symantec

<u>1/30/2007 - **Microsoft ships Windows Vista with ASLR**</u>
Microsoft ships Windows Vista with full Address Space Layout Randomization. The heap is further hardened:
(Removal of lookaside lists and array lists, Block metadata encryption, Header cookie scope extended, validated in more places, Dynamic change of heap allocation algorithms (LFH), Terminate on heap corruption (default for system apps), RtlDeleteCriticalSection technique mitigated by RtlSafeRemoveEntryList, FreeList[0] technique mitigated by RtlpFastRemoveFreeBlock) [63]

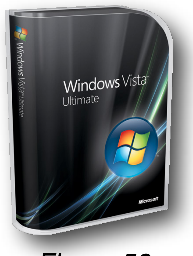


Figure 58 Windows Vista

<u>3/1/2007 - **"GS and ASLR in Windows Vista" Presented**</u>
Ollie Whitehouse presents on "GS and ASLR" in windows Vista at BlackHat Europe. He documented the workings of GS and ASLR on Vista and documented weaknesses in the randomness for HeapAlloc, image and PEB randomization.

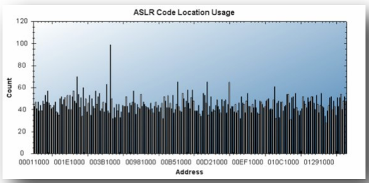


Figure 59 - ASLR Analysis

<u>3/3/2007 - **First NULL ptr deref exploit for Linux kernel released**</u>
Though the actual exploit was developed on August 10th, 2006, it was released to the public on March 3rd 2007. Brad Spengler (spender) announced the exploit to the Daily Dave mailing list [64]. The attack payload disabled SELinux and all other LSM modules atomically through code-scanning heuristics (no symbols required).

<u>3/13/2007 - **OpenBSD gets its second remote exploit in 10 years**</u>
Alfred Ortega of Core Security writes the second remote exploit for OpenBSD in 10 years, a buffer overflow in the IPv6 networking code [65].

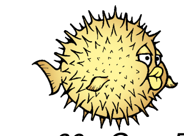


Figure 60 - OpenBSD

3/27/2007 - "Heap Feng Shui in JavaScript" Published
Alex Sotirov (solar eclipse) presented "*Heap Feng Shui in JavaScript*" at BlackHat Europe [66].
His technique results in precise control of the browser heap by intelligent JavaScript allocation and garbage collection.

Figure 61
Alex Sotirov
(solar eclipse)

3/29/2007 - Microsoft issued Security Advisory (935423) (ANI bug)
After public reports of exploitation in the wild, Microsoft released a patch and advisory to remediate the animated cursor overflow. The bug was reported the previous year by Alexander Sotirov. The bug could be reliably exploited on all versions of Windows, including Vista since the structure overflowed lacked a string, which led /GS heuristics to skip the use of a cookie.

Figure 62
Windows Cursor

5/1/2007 - "Reducing the effective Entropy of gs cookies" Published
Matt Miller (skape) publishes "*reducing the effective entropy of GS cookies*" in Uninformed Volume 7 [67]. He showed that the GS cookie could be successfully derived for local attacks.

7/6/2007 - "Understanding and Bypassing Windows Heap Protection" Presented
Nicolas Waisman presents "*Understanding and Bypassing Windows Heap Protection*" [68]. He declared the death of the write4 primitive, in favor of application specific attacks. he distinguished between hard and soft mem leaks and showed off immunity debuggers !HEAP utility functions

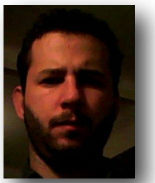


Figure 63
Nicolas
Waisman

8/4/2007 - Immunity Debugger Released
Nico Waisman of Immunity announced the v1.0 release of Immunity Debugger. Built on Ollydbg but with a fully scriptable python engine, and scripts/plugins tailored to exploit development

1/8/2008 - Remote kernel pool overflow in XP parsing IGMPv3 packets advisory
MS08-001 (Remote kernel pool overflow in XP parsing IGMPv3 packets) advisory was released crediting Alex Wheeler and Ryan Smith of IBM-ISS [69]. Initially considered "unlikely" exploitable by Microsoft. ImmunityInc released a working exploit 10 days later [70].

2/8/2008 - Vista SP1 ships with added mitigations
Vista SP1 ships with Structured Exception Handling Overwrite Protection (SEHOP). While SAFESEH from VS2003 required a recompile, SEHOP can be enabled without one. SEHOP inserts a symbolic exception registration record as the last exception registration record, allowing the SEH list to be examined for corruption.[71]

<u>2/17/2008 - “ASLR Smack & Laugh Reference” published</u>

The paper written by Tilo Müller provides in-depth discussion of attacks against Address Space Layout Randomization implementations [72]. The summary of the paper lists the attacks mentioned: "*dos, brute force, ret2text, ret2bss, ret2data, ret2heap, string and function pointer redirecting, stack stethoscope and formatted information, ret2ret, ret2pop, ret2esp, ret2eax and finally ret2got. Furthermore I pointed at integers, off-by-ones and dtors that are still exploitable under special circumstances (i.e. in a combination with one of the ASLR smashing methods listed above). Some of these techniques like ret2text (especially borrowed code), string and function pointer redirecting or ret2got are also useful to bypass a nonexecutable stack.*"

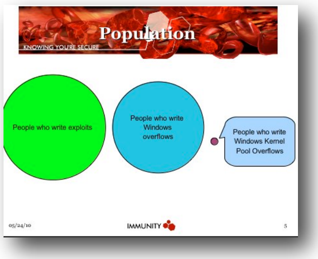
<u>4/14/2008 - "Application-Specific Attacks - Leveraging the ActionScript Virtual Machine" published</u>

Mark Dowd (duke) Publishes "*Application-Specific Attacks - Leveraging the ActionScript Virtual Machine*" [73]. Other than presenting the opportunity to compromise most machines in the world, Dowd managed to turn a difficult to exploit Integer Overflow (and a null pointer dereference) into a reliable write4 primitive, which he then used to reduce the built in verification of the Flash Virtual Machine, allowing his code to run

Figure 64
Mark Dowd (duke)

<u>"Real World Kernel Pool Exploitation" published - 7/1/2008</u>

Kostya Kortchinsky publishes "*Real World Kernel Pool Exploitation*" at SysCan 08 [74]. He points out the lack of a pool cookie (or safe unlinking) in the retail build. Added additional possible target pointers.



Population
KNOWING YOU'RE SECURE

People who write exploits

People who write Windows overflows

People who write Windows Kernel Pool Overflows

05/24/10 IMMUNITY 5

Figure 65 - Kernel Pool Overflows

<u>.Net controls used to exploit IE - 7/29/2008</u>

Mark Dowd and Alex Sotirov publish a paper at BlackHat USA titled “*Bypassing Browser Memory Protections*” [75]. Amongst many other attacks they make use of .Net controls embedded in a page to load shellcode into executable sections of memory. From the paper: “*Since the .NET binaries have the same basic format as PE files, the CLR maps them into memory as images. This means that the kernel parses the PE header and loads all PE sections in memory the same way it does for normal executables or DLLs. In doing this, it sets the page permissions for each section according to the flags in the PE header. If the binary contains an executable section, it will be loaded in memory and its pages will be marked executable.*”

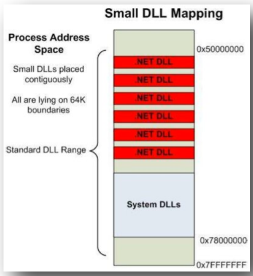


Figure 66
.NET spraying on the Heap

They also covered a number of different “spraying” techniques to bypass ASLR, including .Net spraying.

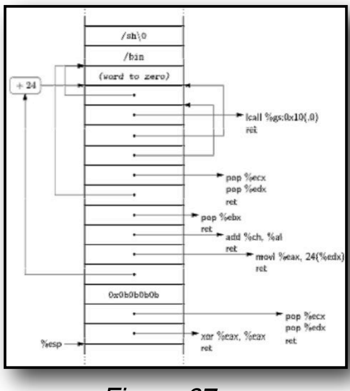


Figure 67
Return Oriented Programming

<u>"Return-Oriented Programming" paper published</u> - 8/4/2008
Hovav Shacham, Erik Buchanan, Ryan Roemer and Stefan Savage introduce Return Oriented Programming, a generalization of return-into-libc that allows an attacker to undertake arbitrary, Turing-complete computation without injecting code, therefore bypassing defenses like DEP and W^X The talk is based on the paper they delivered in 2007 titled "*the geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86).* [76]" and leaned heavily on the previous return-into-libc work done by solar designer, nergal, barnaby jack and scut.

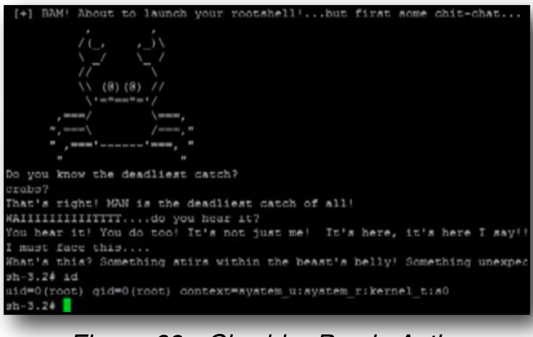
<u>"Attacking the Vista Heap" published</u> - 8/8/2008

Figure 68
Ben Hawkes

Ben Hawkes delivers talk on "*Attacking the Vista Heap*" at BlackHat USA 2008 [77]. His talk covered application specific double-frees. He effectively used heap spraying to spray heap structures to overwrite function pointers in use by the heap management routines (like commitHook). He also abused overflows in the Low Fragmentation Heap.

<u>PAX_USERCOPY released -</u> 4/18/2009
PAX_USERCOPY was a feature invented and initially developed by Brad Spengler, then improved upon and added to PaX by the PaX Team. The purpose of the feature is described in its configuration help: "[PAX_USERCOPY causes the kernel to] enforce the size of heap objects when they are copied in either direction between the kernel and userland, even if only a part of the heap object is copied. Specifically, this checking prevents information leaking from the kernel heap during kernel to userland copies (if the kernel heap object is otherwise fully initialized) and prevents kernel heap overflows during userland to kernel copies." It also includes limited stack infoleaking protection. The protection has proved itself useful by discovering unfixed information leaks in the Linux kernel.

<u>grsecurity adds limited integer-overflow defense</u> - 6/8/2009
In July of 2009, Brad Spengler (spender) added a limited defense against integer overflows in the Linux kernel. Specifically, the feature focuses on integer overflows that occur during the evaluation of arguments passed to any kernel allocation function that accepts a size argument. (The method of detection came from Felix von Leitner's paper "*Catching Integer Overflows in C*" [78])



```
 [+] BAM! About to launch your rootshell!...but first some chit-chat...
           ,       ,
          /(_,    ,_)\
          \ _/    \_ /
          //        \
          \\ (@)(@) //
           \'="=="='/
       ,===/        \===,
      ",===\        /===,"
      " ,==='------'===, "
       "                "
Do you know the deadliest catch?
crabs?
That's right! MAN is the deadliest catch of all!
WAIIIIIIIIIIITTT....do you hear it?
You hear it! You do too! It's not just me!  It's here, it's here I say!!
I must face this....
What's this? Something stirs within the beast's belly! Something unexpec
sh-3.2# id
uid=0(root) gid=0(root) context=system_u:system_r:kernel_t:s0
sh-3.2#
```

Figure 69 - Cheddar Bay in Action

<u>Cheddar Bay kernel exploit released</u> <u>- 7/16/2009</u>
Brad Spengler (spender) released the amusing Cheddar Bay exploit against a NULL pointer dereference vulnerability in the /dev/net/tun driver of the Linux kernel. What made the vulnerability interesting was that from a reading of the source code, it would seem to be un-exploitable for code execution. Due to the kernel compiling without -fno-delete-null-pointer-checks, any dereference of a NULL pointer causes any future NULL checks against the pointer to be removed. The exploit also revealed a vulnerability in SELinux by which it overrode the mmap_min_addr LSM hook and then provided weaker default security for several distributions.

<u>7/23/2009 -</u> **<u>0day discovered in the wild performing heap spray through flash</u>**
CVE-2009-1862 is released to cover the vulnerability in Adobe Reader and Adobe Flash. The captured zero day is found to be using Flash to spray the heap [78].

<u>8/12/2009 -</u> **<u>"Nozzle, a Defense Against Heap Spraying Code Injection Attacks" published</u>**
At the 18th annual Usenix Security Symposium, Microsoft presented "Nozzle" [79], to counter memory spraying attacks. Nozzle monitors heap utilization and flags when a high fraction of the heap contains suspicious objects.

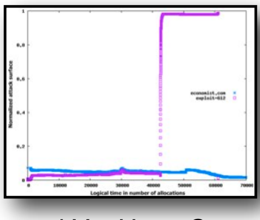
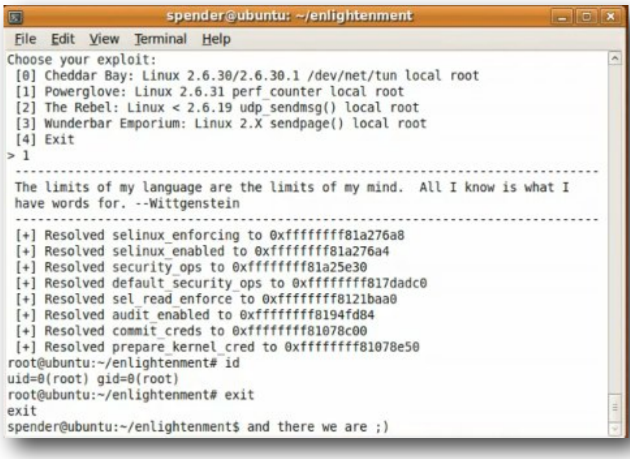


Figure 70 - Normal Vs. Heap Spray Activity

<u>9/9/2009 -</u> **<u>Enlightenment Linux kernel exploitation framework released</u>**
Brad Spengler(spender) released an extensive exploitation framework [80] for the Linux kernel including exploits for all the major NULL pointer dereference vulnerabilities in 2009. The framework is highly-weaponized: it disables LSM, SELinux, AppArmor, IMA, Tomoyo, works under Xen, fakes the security status of SELinux, bypasses SELinux execmod protections, grants full root and capabilities on all 2.4 and 2.6 kernels (including those with the new credentials system), and includes every public bypass method for mmap_min_addr



```
Choose your exploit:
 [0] Cheddar Bay: Linux 2.6.30/2.6.30.1 /dev/net/tun local root
 [1] Powerglove: Linux 2.6.31 perf_counter local root
 [2] The Rebel: Linux < 2.6.19 udp_sendmsg() local root
 [3] Wunderbar Emporium: Linux 2.X sendpage() local root
 [4] Exit
> 1
 ------------------------------------------------------------------------------
 The limits of my language are the limits of my mind.  All I know is what I
 have words for. --Wittgenstein
 ------------------------------------------------------------------------------
 [+] Resolved selinux_enforcing to 0xffffffff81a276a8
 [+] Resolved selinux_enabled to 0xffffffff81a276a4
 [+] Resolved security_ops to 0xffffffff81a25e30
 [+] Resolved default_security_ops to 0xffffffff817dadc0
 [+] Resolved sel_read_enforce to 0xffffffff8121baa0
 [+] Resolved audit_enabled to 0xffffffff8194fd84
 [+] Resolved commit_creds to 0xffffffff81078c00
 [+] Resolved prepare_kernel_cred to 0xffffffff81078e50
root@ubuntu:~/enlightenment# id
uid=0(root) gid=0(root)
root@ubuntu:~/enlightenment# exit
exit
spender@ubuntu:~/enlightenment$ and there we are ;)
```

Figure 71 - Enlightenment in Action

<u>2/3/2010 - Pointer Inference and JIT Spray</u>
Dion Blazakis presents a talk on bypassing DEP and ASLR on Windows Vista using implementation details of the Flash Player virtual machine and Pointer Inference [81]. “Pointer inference” uses an indirect method (ordering of a Dictionary iteration) to disclose heap addresses of runtime objects. JIT spraying uses predictable code generation patterns to construct shellcode in executable memory bypassing DEP.

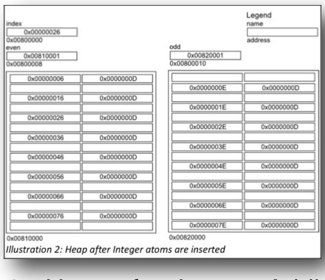


Figure 72 - Heap after Integer Addition to Dictionary.

<u>4/9/2010 - PaX Team releases UDEREF/amd64</u>
The PaX Team released an implementation of UDEREF for the 64-bit Intel Architecture. The implementation is drastically different, as it was not possible to use the expand-down segment method used by UDEREF/x86.

<u>4/10/2010 - “zero allocation vulnerabilities” uncovered</u>
Julien Vanegue publishes zero-sized heap allocation vulnerabilities and a way to partially automate finding of such bugs using theorem proving techniques. [82]

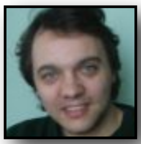


Figure 73
Julien Vanegue

Conclusion ?

Clearly there isn’t one.

Great strides have been made in generic exploit mitigations, and the execution of reliable memory corruption attacks are far more complex today than they were 10 years ago. This ray of sunshine is clouded out however by the fact that applications are far more complex and this added set of rich functionality also serves to give attackers fine grained control of the memory space of targeted applications.

What the mitigations take away, the added functionality gives back. In the near future we are likely to see much more research into memory leakage areas, where attackers are able to reliably deduce the state of the running system since this largely mitigates the obfuscation added by ASLR, while DEP and anti-DEP tricks are likely to continue being a cat and mouse game for some time.

Right now researchers are hard at work on both sides of the fight. As new attacks and mitigations come to light, they will find a mention on: http://ilm.thinkst.com/folklore/

Drop by, add an event..

Acknowledgments

The first acknowledgment needed is all the researchers who have done the awesome work documented. While finding another bug, in another daemon is fairly common place today, the researchers mentioned in this document have truly advanced the art.

Secondly, i would like to thank Barry Irwin and Bradley Cowie from Rhodes Universities SNRG for data mangling and help. Much coffee is owed!

Finally, i would like to thank the individuals who provided feedback (and events) to the timeline. According to http://www.useit.com/alertbox/participation_inequality.html, in any online community, there tends to be a 90-9-1 rule:

- 90% of users are lurkers (i.e., read or observe, but don't contribute),
- 9% of users contribute from time to time,
- 1% of users participate a lot and account for most contributions.

So it has been too with the timeline. Despite heavy usage, and literally thousands of views only a handful of people have added events, or contributed changes. For this, I am grateful for the feedback and updates from all of the following people (in no particular order): *Brad Spengler (spender), PaX Team, Halvar Flake, icesurfer, Nate Lawson, Chris Wysopal, Saumil Shah, Matt Miller, Ollie Whitehouse, Dennis Groves, Ivan Arce, Mario Vilas, Tyler Shields, Dion Blazakis, georgie, Ben Nagy and the Grugq*.

References

Note: *All references have an additional link to the timeline, where a copy of referenced material (and related information) is stored*

1.James P Anderson. “Planning Study”, ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford, MA (Oct. 1972) [NTIS AD-758 206]; Volume I [http://ilm.thinkst.com/perm/1008]
2.Eugene Spafford. “The Internet Worm Program: Analysis”. Computer Communication Review, (January 1989) [http://ilm.thinkst.com/perm/1002]
3.Ben Hawkes. “A History of Corruption”, KiwiCon III, https://www.kiwicon.org/presentations/#Hawkes, (28 September 2009) [http://ilm.thinkst.com/perm/1141]
4. B.P. Miller, L. Fredriksen, and B. So. "An Empirical Study of the Reliability of UNIX Utilities", Communications of the ACM 33, (12 December 1990). [http://ilm.thinkst.com/perm/1060]
5.Thomas Lopatic. “Vulnerability in NCSA HTTPD 1.3”, Posting to Bugtraq Mailing List, http://seclists.org/bugtraq/1995/Feb/109, (13 Feb 1995), [http://ilm.thinkst.com/perm/1012]
6. mudge. “how to write overflows”, http://l0pht.com/advisories/bufero.html, (20 Oct 1995) [http://ilm.thinkst.com/perm/1005]
7.7. CERT. “CERT® Advisory CA-1995-13 Syslog Vulnerability, http://www.cert.org/advisories/CA-1995-13.html (19 Oct 1995), [http://ilm.thinkst.com/perm/1008]
8. Aleph One. “Smashing The Stack For Fun And Profit.”, Phrack, 7(49), (November 1996) [http://ilm.thinkst.com/perm/1004]
9. Solar Designer. “Getting around non-executable stack (and fix)”, Posting to Bugtraq Mailing List,http://seclists.org/bugtraq/1997/Aug/63 (10 August 1997) [http://ilm.thinkst.com/perm/1007]
10.Crispan Cowan, Calton Pu, Dave Maier, Jonathan Walpole, Peat Bakke, Steve_Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. “StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks”,Proceedings of the 7th USENIX Security Symposium (26 Jan 1998) [http://ilm.thinkst.com/perm/1027]
11.Tim Newsham. “Re: StackGuard: Automatic Protection From Stack-smashing Attacks”, Posting to Bugtraq Mailing List, http://seclists.org/bugtraq/1997/Dec/123 (19 Dec 1997) [http://ilm.thinkst.com/perm/1029]
12.DilDog. “L0pht Advisory MSIE4.0(1)”, Posting to Bugtraq Mailing List, http://seclists.org/bugtraq/1998/Jan/57 (14 Jan 1998) [http://ilm.thinkst.com/perm/1090]
13.DilDog. “The Tao of Windows Buffer Overflow”,http://www.cultdeadcow.com/cDc_files/cDc-351/ (16 April 1998) [http://ilm.thinkst.com/perm/1088]
14.Matt Conover and w00w00. “w00w00 on Heap Overflows”,http://www.w00w00.org/files/articles/heaptut.txt (31 Jan 1999) [http://ilm.thinkst.com/perm/1024]
15.dark spyrit AKA Barnaby Jack. “Win32 Buffer Overflows (Location, Exploitation and Prevention)”, Phrack 55 (09 Sep 1999) [http://ilm.thinkst.com/perm/1031]
16.klog. “The Frame Pointer Overwrite” Phrack 55 (09 Sep 1999) [http://ilm.thinkst.com/perm/1126]
17.Tymm Twillman. “Exploit for proftpd 1.2.0pre6”, Posting to Bugtraq Mailing List, http://seclists.org/bugtraq/1999/Sep/328 (20 Sep 1999) [http://ilm.thinkst.com/perm/1139]
18.Taeh Oh. "Advanced Buffer Overflow Exploit",http://www.packetstormsecurity.org/papers/unix/adv.overflow.paper.txt (21 Oct 1999) [http://ilm.thinkst.com/perm/1030]
19.Bulba and Kil3r. “Bypassing Stackguard and Stackshield”, Phrack 56 (01 May 2001) [http://ilm.thinkst.com/perm/1033]
20.rix. “Smashing C++ VPTRS” Phrack 56 (01 May 2001) [http://ilm.thinkst.com/perm/1127].
21. twitch. "Exploiting Non-adjacent Memory Spaces", Phrack 56 (01 May 2001) [http://ilm.thinkst.com/perm/1128]
22. zolo.”WuFTPD: Providing *remote* root since at least1994”, Posting to Bugtraq Mailing List, http://seclists.org/bugtraq/2000/Jun/297 (23 June 2000) [http://ilm.thinkst.com/perm/1140]
23.Lamagra Argamal. “format bugs, in addition to the wuftpd bug”, Posting to Bugtraq Mailing List, http://archives.neohapsis.com/archives/bugtraq/2000-06/0256.html (24 Jun 2000) [http://ilm.thinkst.com/perm/1091]

24.Solar Designer. “JPEG COM Marker Processing Vulnerability in Netscape Browsers”, Posting to Bugtraq Mailing List, http://www.openwall.com/advisories/OW-002-netscape-jpeg (25 July 2000) [http://ilm.thinkst.com/perm/1020]
25.Tim Newsham. "Format String Attacks", Posting to Bugtraq Mailing List, http://seclists.org/bugtraq/2000/Sep/214 (09 Sep 2000) [http://ilm.thinkst.com/perm/1117]
26.Juan M. Bello Rivas. "Overwriting the .dtors section", Posting to Bugtraq Mailing List, http://seclists.org/bugtraq/2000/Dec/175 (09 Sep 2000) [http://ilm.thinkst.com/perm/1125]
27.Mike Frantzen, Mike Shuey. "StackGhost: Hardware Facilitated Stack Protection." 2001. USENIX Security Symposium (2001) [http://ilm.thinkst.com/perm/1026]
28.Crispin Cowan, Matt Barringer, Steve Beattie, Greg Kroah-Hartman, Mike Frantzen, Jamie Lokier. ”FormatGuard: Automatic Protection From printf Format String Vulnerabilities”, Proceedings of the 10th conference on USENIX Security Symposium (2001) [http://ilm.thinkst.com/perm/1033]
29.MaXX (Michel Kaempf). “Vudo Malloc Tricks” Phrack (08 Nov 2001) [http://ilm.thinkst.com/perm/1021]
30.anonymous. “Once upon a free()” Phrack (08 Nov 2001) [http://ilm.thinkst.com/perm/1022]
31.nergal. “Advanced return-into-lib(c) exploits (PaX case study)”. Phrack 58 (28 Dec 2001) [http://ilm.thinkst.com/perm/1023]
32.Halvar Flake. "Third Generation Exploits on NT/Win2k Platforms" BlackHat Windows Briefings (07 Feb 2002) [http://ilm.thinkst.com/perm/1045]
33.Chris Ren, Michael Weber, and Gary McGraw. “Microsoft Compiler Flaw Technical Note”, http://www.cigital.com/news/index.php?pg=art&artid=70 (14 Feb 2002) [http://ilm.thinkst.com/perm/1033]
34.David Litchfield. "Non-stack Based Exploitation of Buffer Overrun Vulnerabilities on Windows NT/2000/XP", http://www.ngssoftware.com/papers/non-stack-bo-windows.pdf (5 Mar 2003) [http://ilm.thinkst.com/perm/1130]
35.Tyler Durden. "Bypassing PaX ASLR Protection", Phrack 59 (28 Jul 2002) [http://ilm.thinkst.com/perm/1056]
36.riq, gera. "Advances in Format String Exploitation" Phrack 59 (28 Jul 2002) [http://ilm.thinkst.com/perm/1123]
37.Maximiliano Cáceres. "Syscall Proxying - Simulating Remote Execution" http://www.coresecurity.com/content/syscall-proxying-simulating-remote-execution (1 Aug 2002) [http://ilm.thinkst.com/perm/1122]
38. Gerardo Richarte. “Four different tricks to bypass StackShield and StackGuard protection” (9 April 2002) [http://ilm.thinkst.com/perm/1025]
39.Frédéric Perriot and Peter Szor. "An Analysis of the Slapper Worm Exploit", http://www.symantec.com/avcenter/reference/analysis.slapper.worm.pdf, (13 Nov 2002) [http://ilm.thinkst.com/perm/1068]
40.sn, devik. "Linux on-the-fly kernel patching without LKM", Phrack 58 (28/12/2001) [http://ilm.thinkst.com/perm/1142]
41.Silvio Cesare. "Runtime kernel kmem patching", http://vxheavens.com/lib/vsc07.html (Nov 1998) [http://ilm.thinkst.com/perm/1143]
42.blexim. "Basic Integer Overflows", Phrack 60 (28/12/2002) [http://ilm.thinkst.com/perm/1033]
43.paxteam. "pax-future.txt", http://pax.grsecurity.net/docs/pax-future.txt (28 Dec 2002) [http://ilm.thinkst.com/perm/1108]
44.David Litchfield. "Variations in Exploit methods between Linux and Windows", BlackHat (28 Jul 2003) [http://ilm.thinkst.com/perm/1129]
45.David Litchfield. "Defeating the Stack Based Buffer Overflow Prevention Mechanism of MS Windows 2003 Server", http://www.ngssoftware.com/papers/defeating-w2k3-stack-protection.pdf (8 Nov 2003) [http://ilm.thinkst.com/perm/1035]
46.Matt Conover, Oded Horovitz. "Reliable Windows Heap Exploits" CanSecWest (2004) [http://ilm.thinkst.com/perm/1048]
47.David Litchfield. "Windows Heap Overflows", BlackHat USA 2004 (28 Jul 2004) [http://ilm.thinkst.com/perm/1046]
48.Hovav Shacham, Matthew Page, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, Dan Boneh. "On the effectiveness of address-space randomization", 11th ACM Conference on Computer and Communications Security (25 Oct 2004) [http://ilm.thinkst.com/perm/1105]
49.Matt Conover, "Reliable Windows Heap Overflows", Syscan (2004) [http://ilm.thinkst.com/perm/1063]

50.Alexander Anisimov. "Defeating Microsoft Windows XP SP2 Heap protection and DEP bypass", http://www.ptsecurity.com/download/defeating-xpsp2-heap-protection.pdf (21 Jan 2005) [http://ilm.thinkst.com/perm/1047]
51.Barnaby Jack. “Remote Windows Kernel Exploitation, Step into the Ring 0”, http://research.eeye.com/html/papers/download/StepIntoTheRing.pdf (17 Feb 2005) [http://ilm.thinkst.com/perm/1121]
52.SoBeIt. "Windows Kernel Pool Overflow Exploitation", XCon 2005 (20 Jul 2005) [http://ilm.thinkst.com/perm/1072]
53.Nicolas Falliere. "A new way to bypass Windows heap protections", http://www.symantec.com/connect/articles/new-way-bypass-windows-heap-protections (31 Aug 2005) [http://ilm.thinkst.com/perm/1054]
54.Sebastian Krahmer. "x86-64 buffer overflow exploits and the borrowed code chunks exploitation technique", http://www.suse.de/~krahmer/no-nx.pdf, (28 Sep 2005) [http://ilm.thinkst.com/perm/1070]
55.Matt Miller, Ken Johnson. “Bypassing Windows Hardware-enforced Data Execution Prevention”, Uninformed Journal 2 (2 Oct 2005), [http://ilm.thinkst.com/perm/1070]
56.Michael Howard. “Integer Overflow and operator::new”, http://blogs.msdn.com/b/michael_howard/archive/2005/12/06/500629.aspx (6 Dec 2005) [http://ilm.thinkst.com/perm/1042]
57.Jack Louis. “Webmin miniserv.pl format string vulnerability”, Post to DailyDave Mailing List http://lists.immunitysec.com/pipermail/dailydave/2005-November/002685.html (29 Nov 2005) [http://ilm.thinkst.com/perm/1119]
58.Steven Christey. "Format String Vulnerabilities in Perl Programs", Post to DailyDave Mailing List http://lists.immunitysec.com/pipermail/dailydave/2005-November/002694.html (29 Nov 2005) [http://ilm.thinkst.com/perm/1118]
59.Brett Moore. “Exploiting Freelist[0] On XP Service Pack 2”, http://www.security-assessment.com/files/whitepapers/Exploiting_Freelist[0]_On_XPSP2.zip (7 Dec 2005) [http://ilm.thinkst.com/perm/1055]
60.Matt Miller. "Preventing the Exploitation of SEH Overwrites", Uninformed Journal 5(Sep 2006) [http://ilm.thinkst.com/perm/1065]
61.Derek Soeder. "Memory Retrieval Vulnerabilities", http://research.eeye.com/html/papers/download/eeyeMRV-Oct2006.pdf, (31 Oct 2006) [http://ilm.thinkst.com/perm/1057]
62.Matthew Conover."Double Free Vulnerabilities", http://www.symantec.com/connect/blogs/double-free-vulnerabilities-part-1 (19 Jan 2007) [http://ilm.thinkst.com/perm/1106]
63.Adrian Marinescu. “Windows Vista Heap Management Enhancements”, Black Hat USA. (28 Aug 2006.) [http://ilm.thinkst.com/perm/1043]
64.Brad Spengler. “On exploiting null ptr derefs, disabling SELinux, and silently fixed Linux vulns”, http://lists.immunitysec.com/pipermail/dailydave/2007-March/004133.html (3 Mar 2007) [http://ilm.thinkst.com/perm/1093]
65.Alfred Ortega. “OpenBSD's IPv6 mbufs remote kernel buffer overflow”, http://www.coresecurity.com/content/open-bsd-advisorie (13 Mar 2007) [http://ilm.thinkst.com/perm/1139]
66.Alex Sotirov. “Heap Feng Shui in JavaScript” BlackHat Europe (27 Mar 2007) [http://ilm.thinkst.com/perm/1049]
67.skape. "reducing the effective entropy of GS cookies" Uninformed Volume 7 (1 May 2007) [http://ilm.thinkst.com/perm/1059]
68.Nicolas Waisman. "Understanding and Bypassing Windows Heap Protection", http://www.immunitysec.com/downloads/Heap_Singapore_Jun_2007.pdf (6 Jul 2007) [http://ilm.thinkst.com/perm/1073]
69.Microsoft TechNet, “Microsoft Security Bulletin MS08-001 – Critical”, http://www.microsoft.com/technet/security/bulletin/ms08-001.mspx (8 Jan 2008) [http://ilm.thinkst.com/perm/1074]
70.ImmunitySec. “CANVAS Professional 6.32 includes Immunity's MS08-001 Windows IGMPv3 exploit”, http://www.immunitysec.com/news-latest.shtml (18 Jan 2008) [http://ilm.thinkst.com/perm/1074]
71.Microsoft TechNet. “Preventing the Exploitation of Structured Exception Handler (SEH) Overwrites with SEHOP”, http://blogs.technet.com/b/srd/archive/2009/02/02/preventing-the-exploitation-of-seh-overwrites-with-sehop.aspx (2 Feb 2009) [http://ilm.thinkst.com/perm/1044]
72.Tilo Müller. “ASLR Smack & Laugh Reference”, http://www-users.rwth-aachen.de/Tilo.Mueller/ASLRpaper.pdf (17 Feb 2008) [http://ilm.thinkst.com/perm/1109]
73.Mark Dowd. "Application-Specific Attacks - Leveraging the ActionScript Virtual Machine", http://documents.iss.net/whitepapers/IBM_X-Force_WP_final.pdf (14 April 2008) [http://ilm.thinkst.com/perm/1069]

74.Kostya Kortchinsky. "Real World Kernel Pool Exploitation", SyScan 08 (11 June 2008) [http://ilm.thinkst.com/perm/1071]
75.Mark Dowd, Alex Sotirov “Bypassing Browser Memory Protections”, BlackHat USA (29 July 2008) [http://ilm.thinkst.com/perm/1064]
76.Hovav Shacham. “The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls (on the x86)”, Proceedings of ACM CCS 2007 (30 Oct 2007) [http://ilm.thinkst.com/perm/1144]
77.Ben Hawkes. "Attacking the Vista Heap" BlackHat USA 2008 (28 Jul 2008) [http://ilm.thinkst.com/perm/1067]
78.CVE. “CVE-2009-1862 (UNDER REVIEW)”,http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-1862, (23 Jul 2009) [http://ilm.thinkst.com/perm/1052]
79. Ben Livshits, Ben Zorn. "Nozzle: Counteracting Memory Exploits", Proceedings of the 18th annual Usenix Security Symposium (Aug 2009) [http://ilm.thinkst.com/perm/1053]
80.Brad Spengler. “Enlightenment Linux kernel exploitation framework”, http://grsecurity.net/~spender/enlightenment.tgz (9 Sep 2009)[http://ilm.thinkst.com/perm/1094]
81.Dion Blazakis. “Interpreter Exploitation: Pointer Inference and JIT Spraying” BlackHat-DC (03 Feb 2010) [http://ilm.thinkst.com/perm/1112]
82.Julien Vanegue. “Zero-sized heap allocation vulnerabilities”, hackitoergosum (20 Apr 2010) [http://ilm.thinkst.com/perm/1115]
83.Tim Burrell, Peter Beck. “” EuSecWest 2009 (27 May 2009) [http://ilm.thinkst.com/perm/1145]