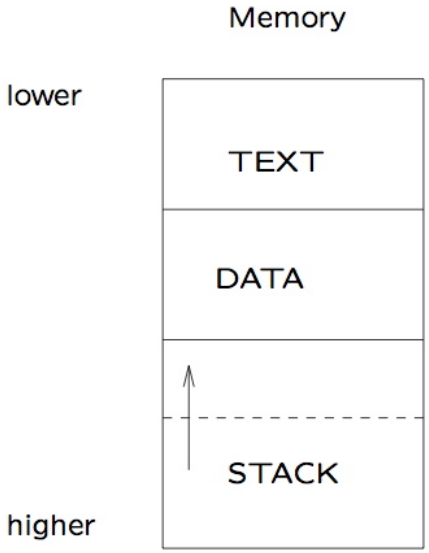
Buffer Overflow Vulnerabilities and Stack Smashing

Simon Foley

January 7, 2014

Security Risks of Buffer Overflows
Process Memory Organization



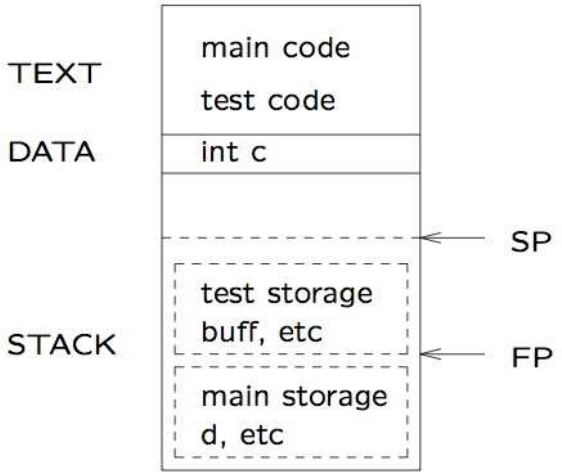
TEXT region stores executable code and constant data for the program. This is a read-only region (attempted writing will produce a segmentation fault).

DATA region stores global data.

STACK is LIFO, holds local variables, parameters and the storage necessary to manage the proper invocation/return of functions.

The STACK grows towards lower memory. If STACK/DATA space becomes exhausted, the process is blocked and the memory allocation enlarged.

Stack Frames




```
   int c;
   void test(int a, int b) {
      char buff[5];
-->   //currently here;
   }/*test*.

   void main() {
      int d=2;
      test(1,d);
   }/*main*/
```

When a function is invoked, a new stack frame is pushed onto the stack. The stack frame provides storage for local variables, etc. When the function exits, the frame is popped off the stack..

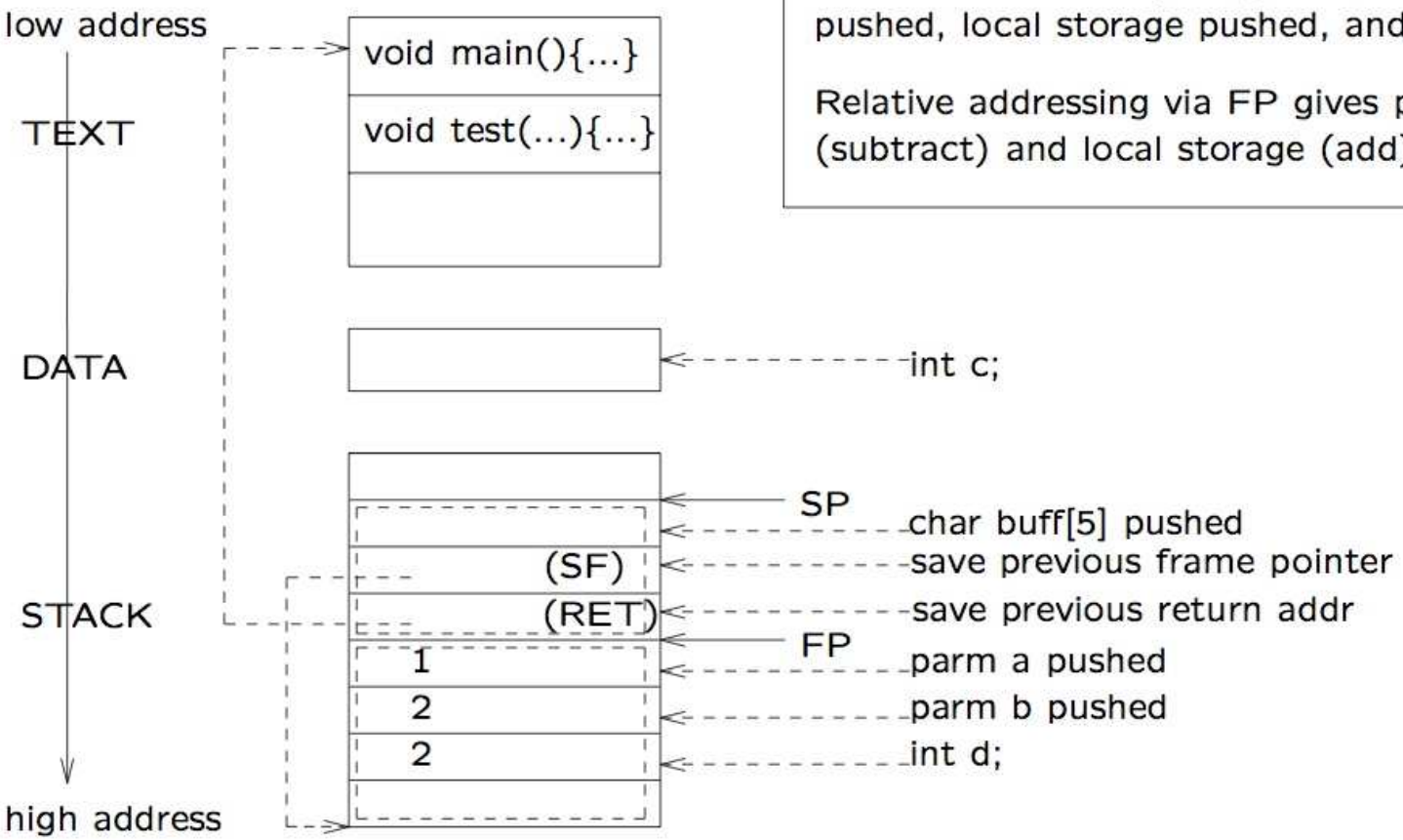
Stack Pointer SP points to the current top of the stack.

Frame Pointer FP points to 'bottom' of current frame. References to variables within the frame are done relative to FP.

Pushing Stack Frames

Before test is invoked, param values are pushed. When the new stack frame for test is created, the return address (main) and previous FP is pushed, local storage pushed, and new FP set.

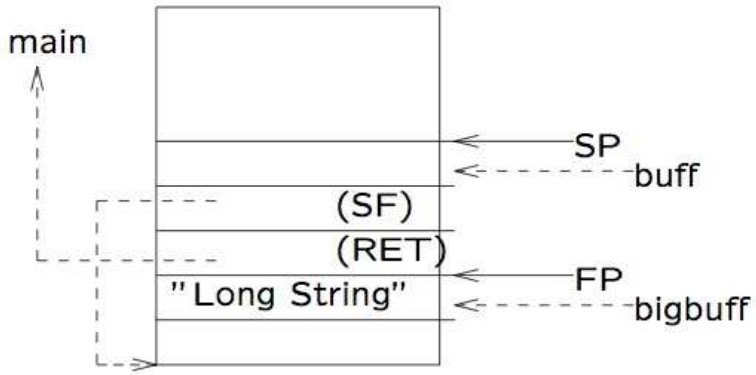
Relative addressing via FP gives parameters (subtract) and local storage (add).



Assembly Fragments for Test Program

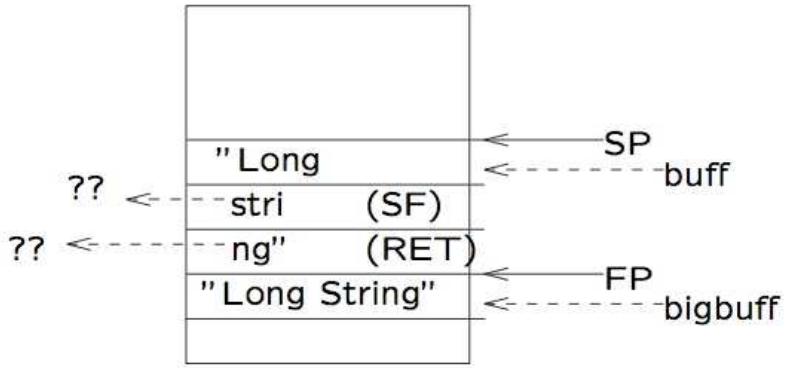
```
[...]
_test:
   pushl %ebp              // push current Frame pointer
   movl %esp,%ebp          // set new frame pointer
   subl $8,%esp            // allocate space for buff[]
   leave                   // pops frame and restores return address
   ret
[...]
_main:
   pushl %ebp              // push current Frame pointer
   movl %esp,%ebp          // set new frame pointer
   pushl $2                //push second parameter
   pushl $1                //push first parameter
   call _test              //call test--pushes instruction pointer
   addl $8,%esp            //restore framepointer
   leave
   ret
[...]
```

Buffer Overflows




```
   void test2(char *str){
-->   char buff[4];
      strcpy(buff,str);
   }/*test2*/
   void main(){
      char* bigbuff="Long String";
      test2(bigbuff);
   }/*main*/
```

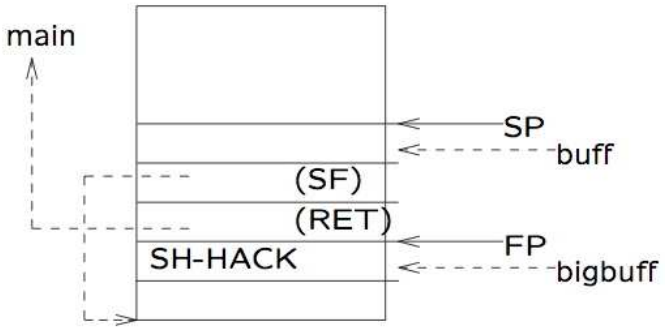
`strcpy` causes buff overflow into FP and RET. Segmentation fault:




```
   void test2(char *str){
      char buff[4];
-->   strcpy(buff,str);
   }/*test2*/
   void main(){
      char* bigbuff="Long String";
      test2(bigbuff);
   }/*main*/
```

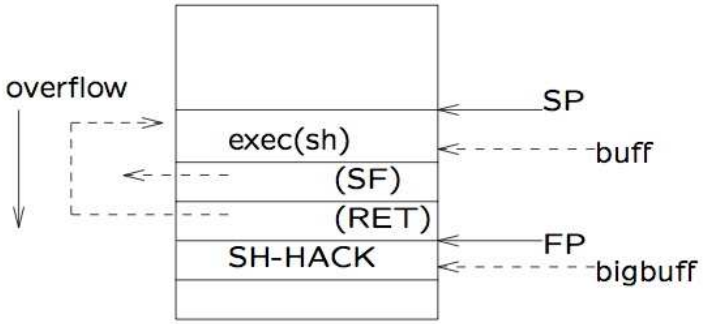
Should use `strncpy` to prevent this!

Manipulating Buffer Overflow




```
   void test2(char *str){
-->   char buff[4];
      strcpy(buff,str);
   }/*test2*/
   void main(){
      char* bigbuff=SH-HACK;
      test2(bigbuff);
   }/*main*/
```

SH-HACK encodes machine-code for `execve(/bin/sh)`, and RET address set up to itself and intended to replace existing RET:




```
   void test2(char *str){
      char buff[4];
-->   strcpy(buff,str);
   }/*test2*/
   void main(){
      char* bigbuff=SH-HACK;
      test2(bigbuff);
   }/*main*/
```

Instead of a segmentation fault, program provides a shell!

SH-HACK can include housekeeping so that `execve(/bin/sh)` returns to the main program when it is done. (Exercise: draw the links that achieve this).

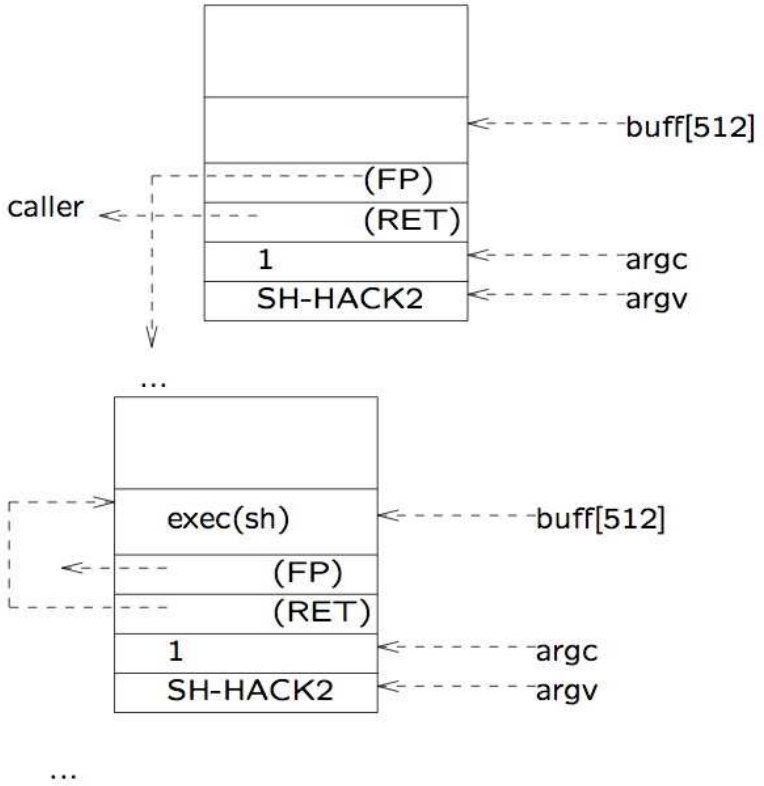
Special Permissions: SUID

When a program is invoked, it runs with the the user id of its invoking process.

When a program file has the setuid root permission set then during execution the user id of the invoking process becomes root.

```
$ ls −l /bin/sleep
−r−xr−xr−x 1 root  wheel  13964 Jan 30  2006 /bin/sleep
$ sleep 60 & ps −u | grep sleep
simon  6514  0.0  0.0  27244  340  p1  S  11:03AM  0:00.01  sleep 60
$
$ ls −l /usr/bin/passwd
−r−sr−xr−x 1 root  wheel  35572 Jan 12  2007 /usr/bin/passwd
$ passwd ... & ps −u | grep passwd
root  6523  0.0  0.0  27256  356  p1  S  11:06AM  0:00.01  passwd
```

Exploiting Buffer Overflow: Stack Smashing Attack




```
   //file vulnerable.c
   void main(int argc, char* argv[]){
      char buff[512];
-->
      if (argc==1)
         strcpy(buffer, argv[0]);
   }/*main*/
```

```
   //file vulnerable.c
   void main(int argc, char* argv[]){
      char buff[512];

      if (argc==1)
-->      strcpy(buffer, argv[0]);
   }/*main*/
```

By experimenting with the right parameter value (SH-HACK2) an attacker can overflow the buffer and force program to execute his own code.

If the program is an suid-root program the attacker gets a root shell! Hundreds of such exploits have been reported.

Example: Ping of Death

Internet Control Message Protocol

$C \rightarrow S$ `ICMP Echo Request`[*optional string*]

$S \rightarrow C$ `ICMP Echo Reply`

IP stack implementation on Server S did not do adequate bounds checking on optional string and an overflow occurs when message is greater than 64K

Attacker sends a specially formatted string which results in server executing some command.

Most implementations have been patched to include proper bounds checking.

Some older OS's do not have patches available for Ping of Death (eg Solaris 2.4, Win 95, MacOS 7, Novell Netware 3, . . .).

Sample Ping of Death Overflow String

The following is a (partial) example of of a 'HACK' string, which when passed to ping on an old unix platform will cause a buffer overflow and returns with a shell running at root (rootshell).

```
unsigned int code[]={0x4ffffb82, 0x4ffffb82, ... // large nr NOPs
        0x7c0802a6, 0x9421fbb0, 0x90010458, 0x3c60f019,
        0x60632c48, 0x90610440, 0x3c60d002, 0x60634c0c,
        0x90610444, 0x3c602f62, 0x6063696e, 0x90610438,
        0x3c602f73, 0x60636801, 0x3863ffff, 0x9061043c,
        0x30610438, 0x7c842278, 0x80410440, 0x80010444,
        0x7c0903a6, 0x4e800420, 0x0 };
```

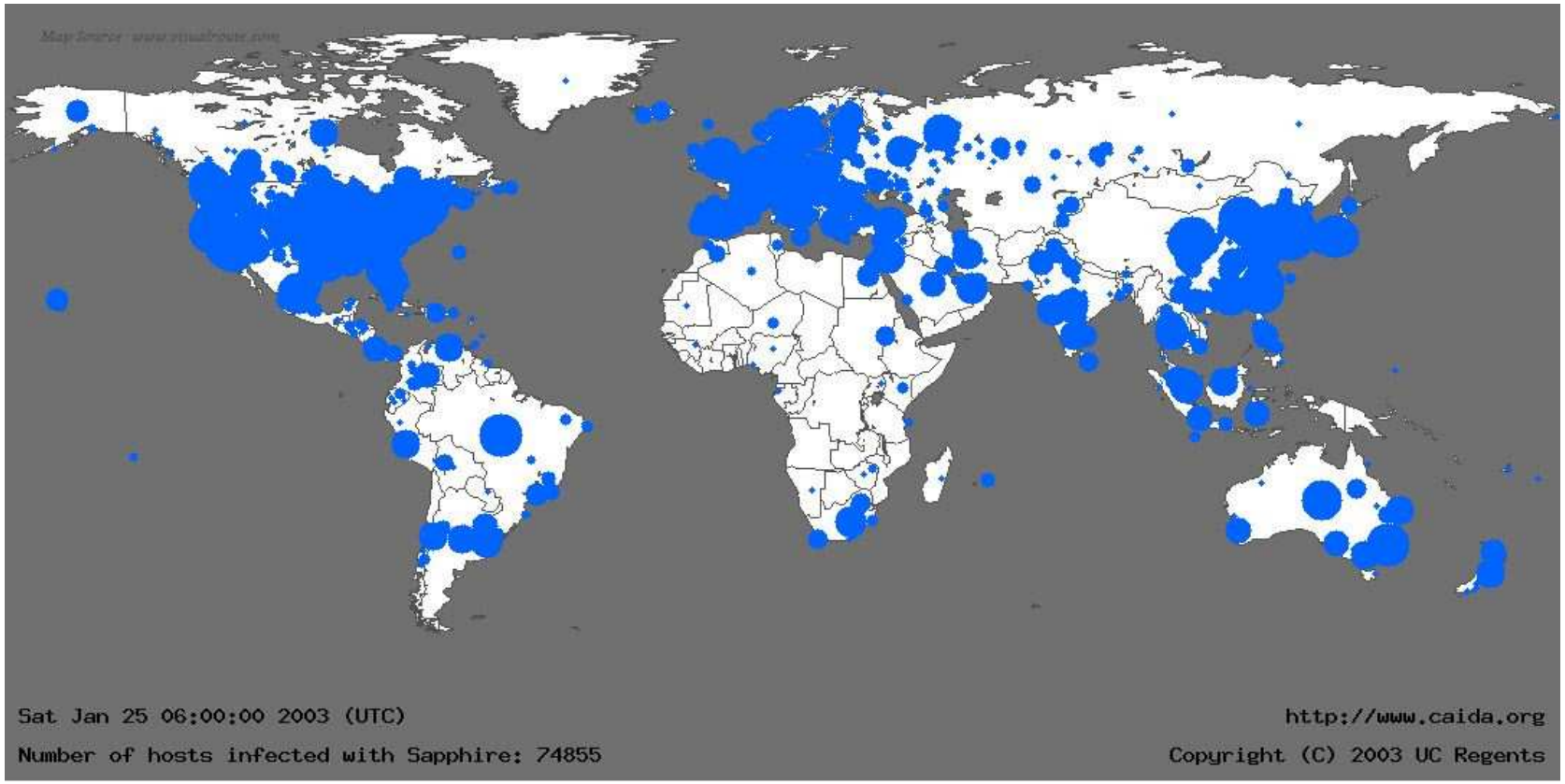
```
$ ls -l /sbin/ping
-r-sr-xr-x 1 root  wheel  33264 Oct 15 23:53 /sbin/ping
$ whoami
simon
$ pingme  # a program that invokes ping, passing above string
$ whoami
root
```

The SQL Slammer Worm [2003]

The SQL Slammer Worm caused a denial of service on some Internet hosts and dramatically slowed down general Internet traffic, starting at 05:30 UTC on January 25, 2003. It spread rapidly, infecting most of its 75,000 victims within 10 minutes. It exploited two buffer overflow bugs in Microsoft's SQL Server database management system.

- Get Inside. Send request to SQL Server causing stack smashing attack.
- Choose Victims at Random. Generate a random IP address, targeting another computer that could be anywhere on the Internet.
- Replicate. Slammer uses its own code as code to be executed from the stack smash.
- Repeat. After sending off the first tainted packet, Slammer loops around immediately to send another to a different computer.

SQL Slammer (Saphire) Worm after 30 mins



The diameter of each circle is a function of the logarithm of the number of infected machines, so large circles visually underrepresent the number of infected cases in order to minimize overlap with adjacent locations.

Stack Smashing: some more examples

Server-based application systems that do not have adequate bounds checking on input channel/port:

- SQL slammer worm (MSQLServ 2003);
- Code red worm (MS IIS 5.0, 2001); ...

Set-uid programs that may run at higher privilege than caller:

- lprm, lpr, crontab, xterm, libc, glibc, samba, ftp, ...

Compilers/interpreters that generated code that result in buffer-overflow:

- Perl, JVM, ...

Make sure your software is always patched and up to date! Be careful when using program libraries. Even if your own code is free of buffer-overflows, it may invoke library code that contains problems. glibc generally considered safer than libc.

Stack smashing also used on XBox, iPhone, ... to run unlicensed software.

50% of home computers are unpatched [Symantec, March, 2006]

p706-eisenberg.pdf (page 1 of 4)
Previous Next Zoom Move Text Select Sidebar Search

The Cornell Commission: On Morris and the Worm

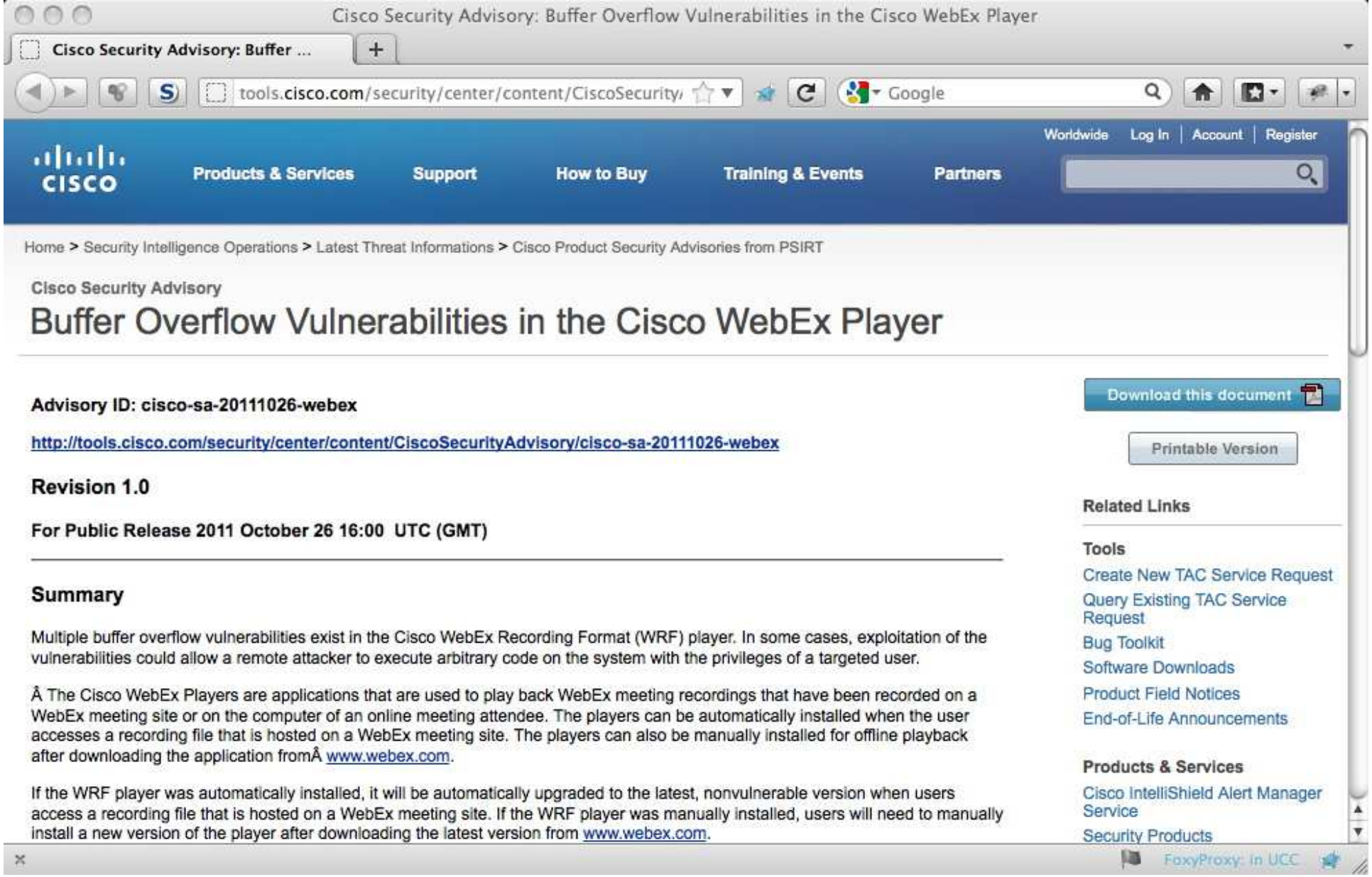
After careful examination of the evidence, the Cornell commission publishes its findings in a detailed report that sheds new light and dispels some myths about Robert T. Morris and the Internet worm.

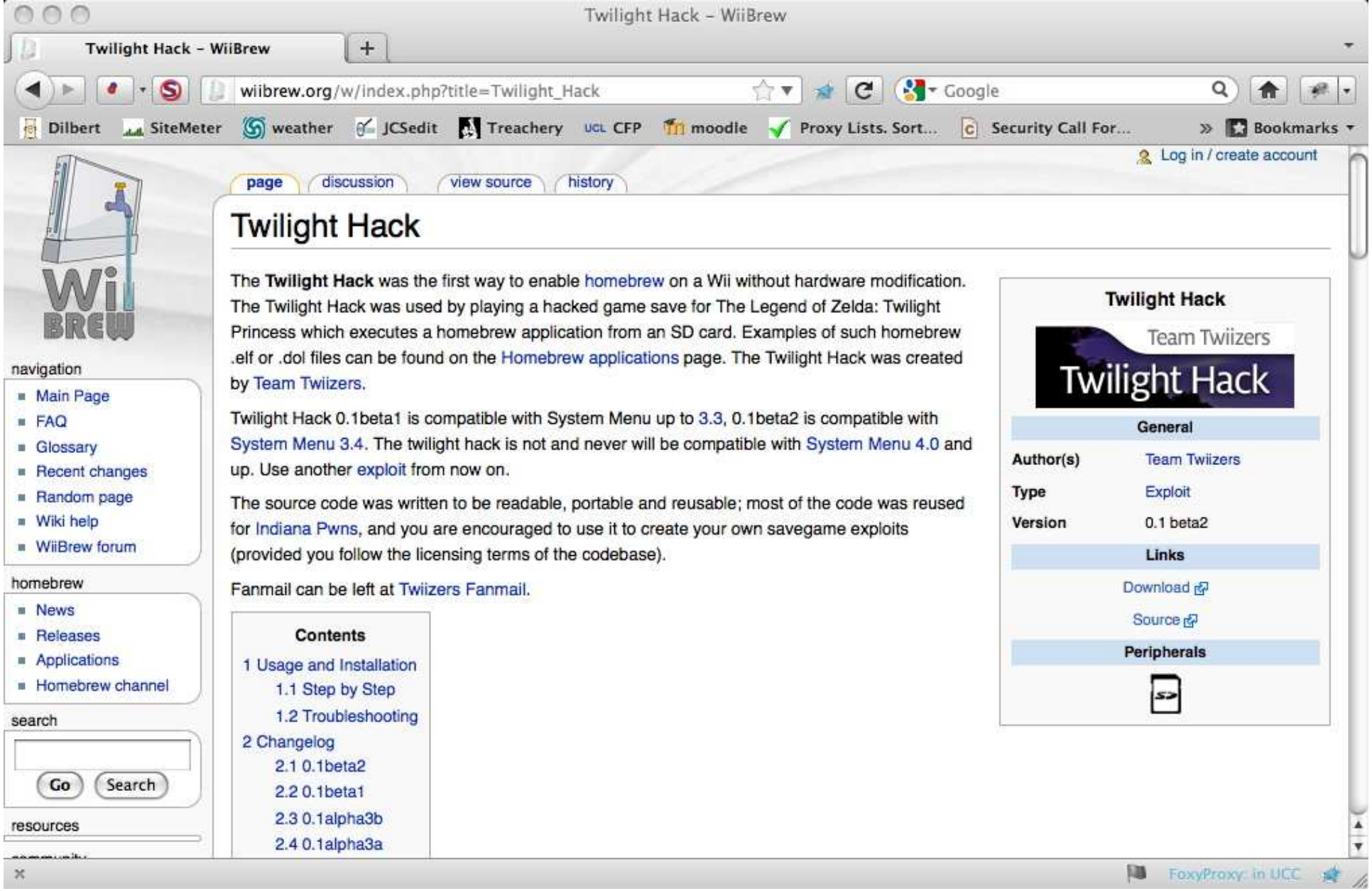
Ted Eisenberg, David Gries, Juris Hartmanis, Don Holcomb, M. Stuart Lynn, Thomas Santoro

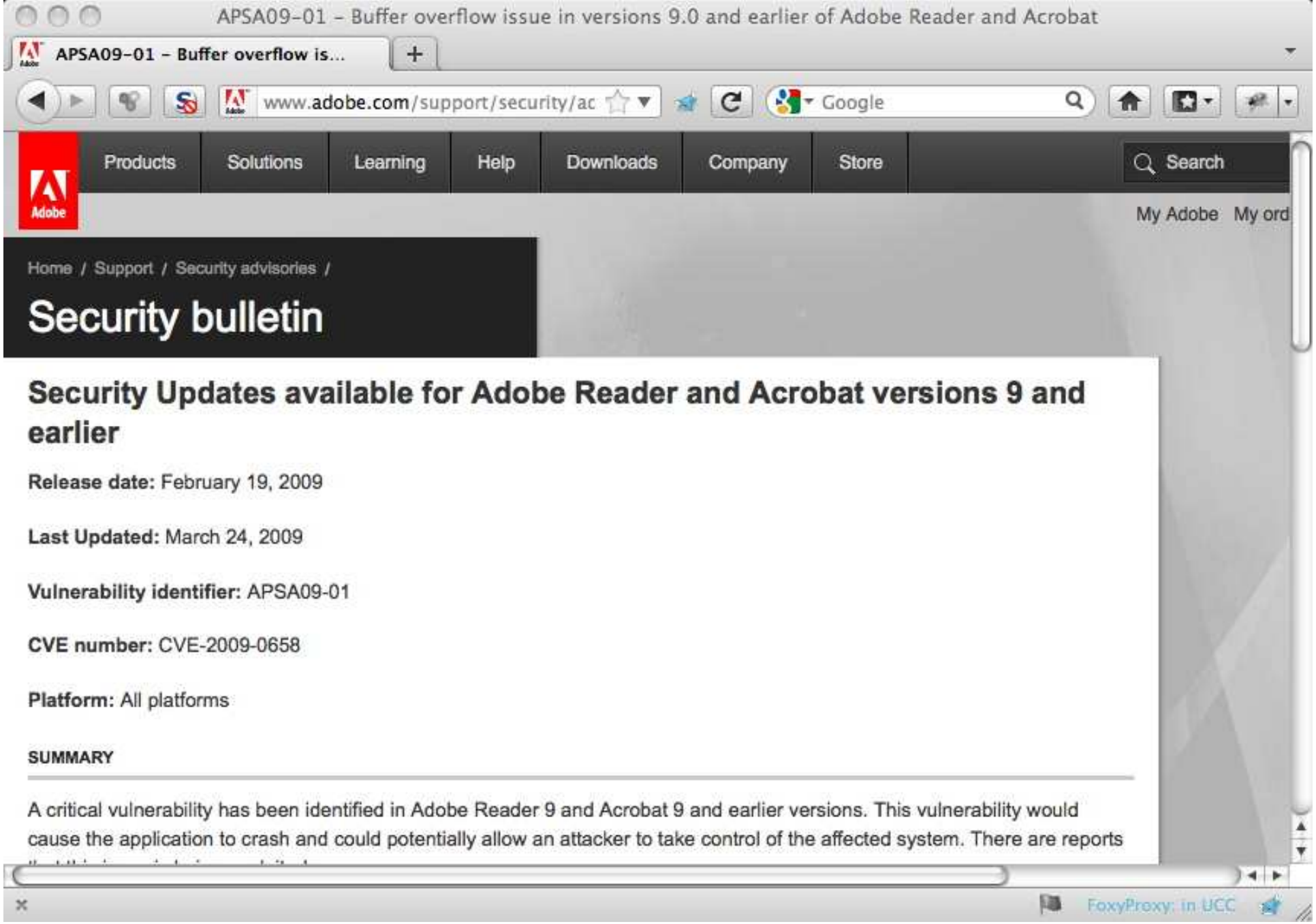
Robert Tappan Morris, Jr. worked alone in the creation and spread of the Internet worm computer program that infected approximately 6,000 computers nationwide last November. That principal conclusion comes from a report issued last April 3, by an internal investigative commission at Cornell University, Ithaca, NY.

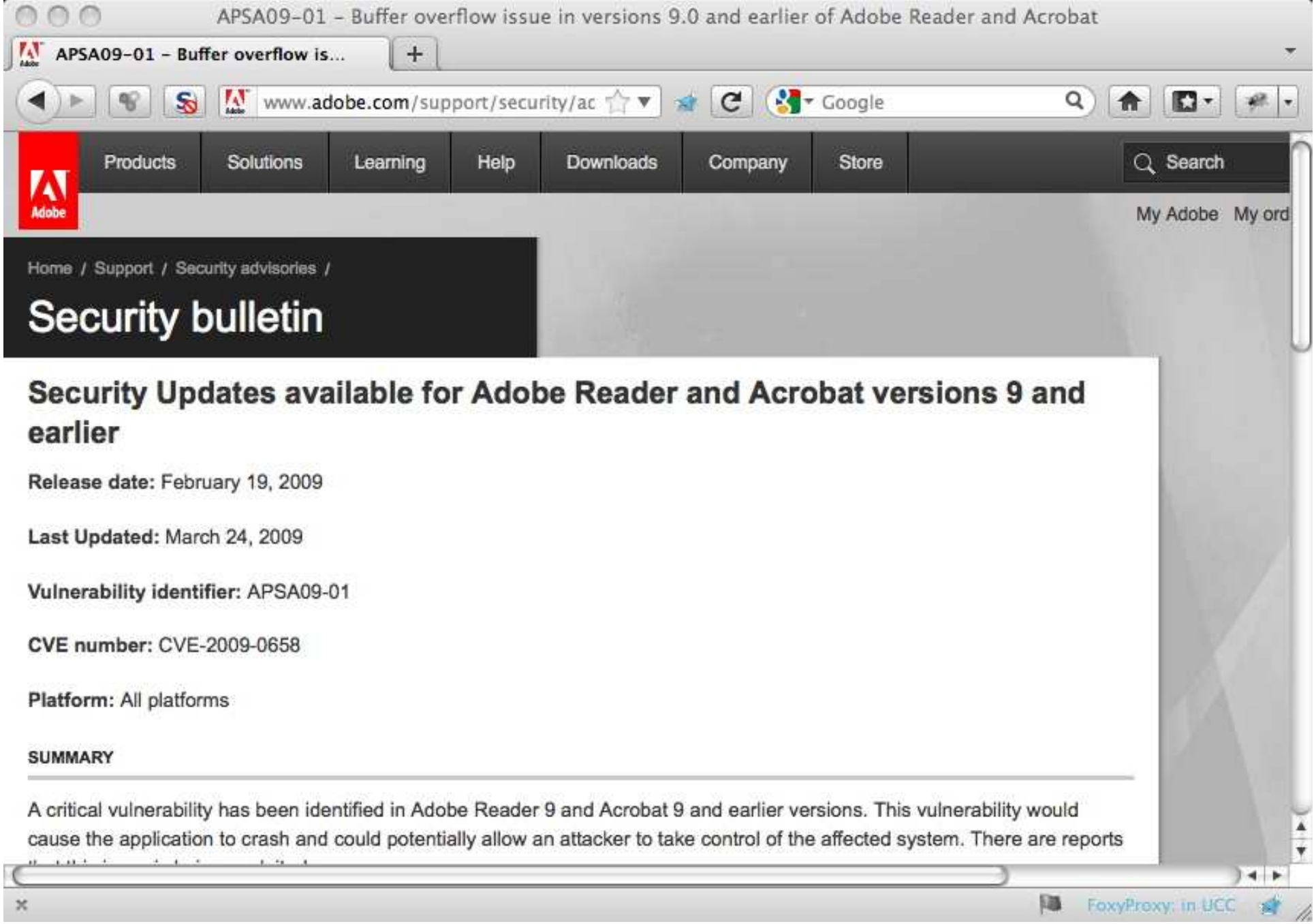
The report labeled Morris' behavior "a juvenile act that ignored the clear potential consequences." Of the graduate student's intentions in releasing the virus, the commission claims: "It may simply have been the unfocused intellectual meandering of a hacker completely absorbed with his creation and unharnessed by considerations of explicit purpose or potential effect."

Morris is currently on leave of absence from Cornell, and the university is prohibited by federal law from commenting further on his academic status. Morris was not interviewed by the commission, a decision he made under advice of his attorney. According to Cornell Provost Robert Barker, both the federal prosecutors and Morris' defense attorney asked that the release of the report be


Cisco Security Advisory: Buffer Overflow Vulnerabilities in the Cisco WebEx Player
Cisco Security Advisory: Buffer ...
tools.cisco.com/security/center/content/CiscoSecurity/
Google
Worldwide
Log In
Account
Register
Products & Services
Support
How to Buy
Training & Events
Partners
Home > Security Intelligence Operations > Latest Threat Informations > Cisco Product Security Advisories from PSIRT
Cisco Security Advisory
Buffer Overflow Vulnerabilities in the Cisco WebEx Player
Advisory ID: cisco-sa-20111026-webex
http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20111026-webex
Revision 1.0
For Public Release 2011 October 26 16:00 UTC (GMT)
Summary
Multiple buffer overflow vulnerabilities exist in the Cisco WebEx Recording Format (WRF) player. In some cases, exploitation of the vulnerabilities could allow a remote attacker to execute arbitrary code on the system with the privileges of a targeted user.
Â The Cisco WebEx Players are applications that are used to play back WebEx meeting recordings that have been recorded on a WebEx meeting site or on the computer of an online meeting attendee. The players can be automatically installed when the user accesses a recording file that is hosted on a WebEx meeting site. The players can also be manually installed for offline playback after downloading the application fromÂ www.webex.com.
If the WRF player was automatically installed, it will be automatically upgraded to the latest, nonvulnerable version when users access a recording file that is hosted on a WebEx meeting site. If the WRF player was manually installed, users will need to manually install a new version of the player after downloading the latest version from www.webex.com.
Download this document
Printable Version
Related Links
Tools
Create New TAC Service Request
Query Existing TAC Service Request
Bug Toolkit
Software Downloads
Product Field Notices
End-of-Life Announcements
Products & Services
Cisco IntelliShield Alert Manager Service
Security Products
FoxyProxy: In UCC


Twilight Hack - WiiBrew
Twilight Hack - WiiBrew
wiibrew.org/w/index.php?title=Twilight_Hack
Google
Dilbert
SiteMeter
weather
JCSedit
Treachery
CFP
moodle
Proxy Lists. Sort...
Security Call For...
Bookmarks
Log in / create account
page
discussion
view source
history
Twilight Hack
The Twilight Hack was the first way to enable homebrew on a Wii without hardware modification. The Twilight Hack was used by playing a hacked game save for The Legend of Zelda: Twilight Princess which executes a homebrew application from an SD card. Examples of such homebrew .elf or .dol files can be found on the Homebrew applications page. The Twilight Hack was created by Team Twiizers.
Twilight Hack 0.1beta1 is compatible with System Menu up to 3.3, 0.1beta2 is compatible with System Menu 3.4. The twilight hack is not and never will be compatible with System Menu 4.0 and up. Use another exploit from now on.
The source code was written to be readable, portable and reusable; most of the code was reused for Indiana Pwns, and you are encouraged to use it to create your own savegame exploits (provided you follow the licensing terms of the codebase).
Fanmail can be left at Twiizers Fanmail.
Contents
1 Usage and Installation
1.1 Step by Step
1.2 Troubleshooting
2 Changelog
2.1 0.1beta2
2.2 0.1beta1
2.3 0.1alpha3b
2.4 0.1alpha3a
Twilight Hack
Team Twiizers
Twilight Hack
General
Author(s) Team Twiizers
Type Exploit
Version 0.1 beta2
Links
Download
Source
Peripherals
navigation
Main Page
FAQ
Glossary
Recent changes
Random page
Wiki help
WiiBrew forum
homebrew
News
Releases
Applications
Homebrew channel
search
Go
Search
resources
FoxyProxy: in UCC


APSA09-01 - Buffer overflow issue in versions 9.0 and earlier of Adobe Reader and Acrobat
APSA09-01 - Buffer overflow is...
www.adobe.com/support/security/ac
Google
Products
Solutions
Learning
Help
Downloads
Company
Store
Search
My Adobe
My ord
Home / Support / Security advisories /
Security bulletin
Security Updates available for Adobe Reader and Acrobat versions 9 and earlier
Release date: February 19, 2009
Last Updated: March 24, 2009
Vulnerability identifier: APSA09-01
CVE number: CVE-2009-0658
Platform: All platforms
SUMMARY
A critical vulnerability has been identified in Adobe Reader 9 and Acrobat 9 and earlier versions. This vulnerability would cause the application to crash and could potentially allow an attacker to take control of the affected system. There are reports
FoxyProxy: in UCC


APSA09-01 - Buffer overflow issue in versions 9.0 and earlier of Adobe Reader and Acrobat
APSA09-01 - Buffer overflow is...
www.adobe.com/support/security/ac
Google
Products
Solutions
Learning
Help
Downloads
Company
Store
Search
My Adobe
My ord
Home / Support / Security advisories /
Security bulletin
Security Updates available for Adobe Reader and Acrobat versions 9 and earlier
Release date: February 19, 2009
Last Updated: March 24, 2009
Vulnerability identifier: APSA09-01
CVE number: CVE-2009-0658
Platform: All platforms
SUMMARY
A critical vulnerability has been identified in Adobe Reader 9 and Acrobat 9 and earlier versions. This vulnerability would cause the application to crash and could potentially allow an attacker to take control of the affected system. There are reports
FoxyProxy: in UCC

Stuxnet

Some Defenses against Stack Smashing

Stack smashing is difficult to get 'right': we need to find the vulnerable buffer, find the position of the RET, etc. Once a buffer vulnerability has been identified the exploit is often implemented as a script that can be used by a relative novice ('script kiddie'). Tools like metasploit provide a range of off the shelf exploit scripts.

Avoiding stack smashing: bound check your arrays!

Don't use C, use a type-safe language such as Java. However some JVM implementations contain errors in type-checking systems that can be exploited.

If you use C then use a patched version of C compiler that provides bound checking. Has performance implications.

Stackguard: a gcc extension/option that puts a random 'canary word' in front of the RET value on the stack. This is checked just before the function returns and if different then the program exits. Can be bypassed if attacker can guess the canary word and place it on the stack
Stackguard/ProPolice is now default for gcc in most linux distributions.