



IT-Security (ITS) B1

DIKU, E2019



Today's agenda

Part 1: Some common and notable bugs

Part 2: A case story in buffer overflows



Lecture plan

Week	Date	Time	Lecture	Topic
----	----	-----	-----	-----
36	02 Sep	09-11	TL	Introduction, security concepts and the threat of hacking
	06 Sep	09-11	TL	Buffer overflow
37	09 Sep	09-11	CJ	Software security, Operating system security
	13 Sep	09-11	CJ	User authentication and access control
38	16 Sep	09-11	TJ	Malicious software
	20 Sep	09-11	CJ	Firewalls and denial-of-service attacks
39	23 Sep	09-11	CJ	Cloud and IoT
	27 Sep	09-11	TL	Cryptography
40	30 Sep	09-11	TL	Internet security protocols
	04 Oct	09-11	TL	Intrusion detection
41	07 Oct	09-11	TL	Forensics
	12 Oct	09-11	CJ	IT security management
42				Fall Vacation - No lectures
43	22 Oct	09-11	CJ	Privacy 1
	25 Oct	09-11	CJ	Privacy 2 - GDPR
44	29 Oct	15-16	Guest	TBA
		16-17	All	Recap and Q/A
45	06 Nov			Exam



Definitions

A vulnerability is a software flaw that can be exploited by an attacker.

(Bugs are plentiful. Not all bugs can be exploited.)

Not all vulnerabilities matter the same.

Vulnerabilities are exploited to run malware, in a nutshell.

A vulnerability

BlueKeep

From Wikipedia, the free encyclopedia

Not to be confused with [BlueBEEP](#).

BlueKeep ([CVE-2019-0708](#)^[]) is a [security vulnerability](#) that was discovered in Microsoft's [Remote Desktop Protocol](#) implementation, which allows for the possibility of [remote code execution](#).

First reported in May 2019, it is present in all unpatched Windows NT-based versions of Microsoft Windows from [Windows 2000](#) through [Windows Server 2008 R2](#) and [Windows 7](#). Microsoft issued a security patch (including an out-of-band update for several versions of Windows that have reached their end-of-life, such as [Windows XP](#)) on 14 May 2019. On 13 August 2019, related Bluekeep security vulnerabilities, collectively named **DejaBlue**, were reported to affect newer Windows versions, including [Windows 7](#) and all recent versions up to [Windows 10](#) of the operating system, as well as the older Windows versions.^[3]

Contents ^[hide]

- [History](#)
- [Mechanism](#)
- [Mitigation](#)
- [See also](#)
- [References](#)
- [External links](#)

BlueKeep



A logo created for the vulnerability, featuring a [keep](#), a [fortified tower](#) built within castles.

CVE identifier(s) [CVE-2019-0708](#)^[]

Date patched 14 May 2019; 3 months ago^[1]

Discoverer [UK National Cyber Security Centre](#)^[2]

Affected software [pre-Windows 8](#) versions of [Microsoft Windows](#)



Common Vulnerabilities Scoring System

Attack vector: Local vs. Remote

Attack complexity: High, Medium, Low

Authentication: Required vs. Not

Impact: C/I/A

CVSS v2.0 Severity and Metrics:

Base Score: 10.0 HIGH

Vector: (AV:N/AC:L/Au:N/C:C/I:C/A:C) (V2 legend)

Impact Subscore: 10.0

Exploitability Subscore: 10.0

Access Vector (AV): Network

Access Complexity (AC): Low

Authentication (AU): None

Confidentiality (C): Complete

Integrity (I): Complete

Availability (A): Complete

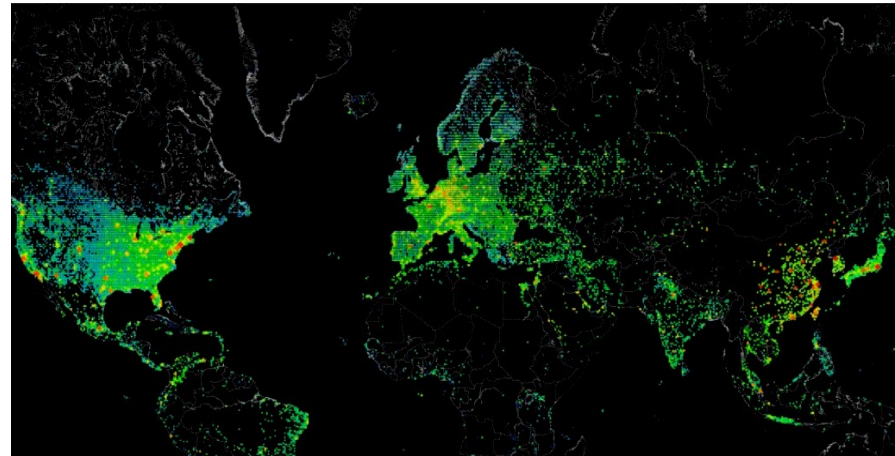
Additional Information:

Allows unauthorized disclosure of information

Allows unauthorized modification

Allows disruption of service

Shodan and BlueKeep





Another vulnerability

Vulnerability Summary for CVE-2010-2883

Original release date: 09/09/2010

Last revised: 08/04/2011

Source: US-CERT/NIST

Overview

Stack-based buffer overflow in CoolType.dll in Adobe Reader and Acrobat 9.x before 9.4, and 8.x before 8.2.5 on Windows and Mac OS X, allows remote attackers to execute arbitrary code or cause a denial of service (application crash) via a PDF document with a long field in a Smart INdependent Glyphlets (SING) table in a TTF font, as exploited in the wild in September 2010. NOTE: some of these details are obtained from third party information.

Impact

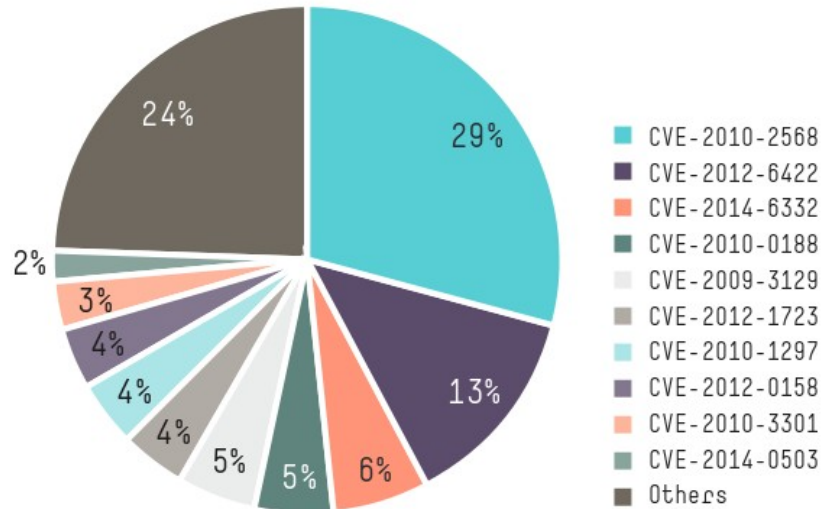
CVSS Severity (version 2.0):

CVSS v2 Base Score: 9.3 (HIGH) (AV:N/AC:M/Au:N/C:C/I:C/A:C) (legend)

Impact Subscore: 10.0

Exploitability Subscore: 8.6

Too old? Old exploits never die (2015)





2019 so far

	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Acrobat Reader Dc	Adobe	Application	341
2	Acrobat Dc	Adobe	Application	341
3	Debian Linux	Debian	OS	321
4	Cpanel	Cpanel	Application	315
5	Windows Server 2016	Microsoft	OS	280
6	Windows 10	Microsoft	OS	278
7	Windows Server 2019	Microsoft	OS	276
8	Windows Server 2008	Microsoft	OS	197
9	Windows 7	Microsoft	OS	196
10	Windows Server 2012	Microsoft	OS	195



Types of vulnerabilities, include:

Format string, overflow, over-read, load order, typos, dangling pointers, code injection, command injection, race conditions, use-after-free, and more.



Where's the bug?



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

```
int main () {  
    int i;  
    printf("Enter a value: ");  
    scanf("%d", &i);
```

```
    if (i < 0)  
        goto fail;  
    if (i > 100)  
        goto fail;  
        goto fail;  
    if (i%2 == 0)  
        goto fail;
```

```
    return;
```

```
fail:  
    printf("Fail\n");  
    return;  
}
```

```
$ ./a.out
```

```
Enter a value: 2  
Fail
```

```
$ ./a.out
```

```
Enter a value: 3  
Fail
```



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

```
int main () {  
    int i;  
    printf("Enter a value: ");  
    scanf("%d", &i);
```

```
    if (i < 0)  
        goto fail;  
    if (i > 100)  
        goto fail;  
    //goto fail;  
    if (i%2 == 0)  
        goto fail;
```

```
    return;
```


```
fail:  
    printf("Fail\n");  
    return;  
}
```

```
$ ./a.out  
Enter a value: 2  
Fail
```

```
$ ./a.out  
Enter a value: 3  
Fail
```

Apple iOS Goto Fail

```
1  static OSStatus
2  SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
3                                  uint8_t *signature, UInt16 signatureLen)
4  {
5      OSStatus      err;
6      ...
7
8      if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
9          goto fail;
10     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
11         goto fail;
12     goto fail;
13     if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
14         goto fail;
15     ...
16
17 fail:
18     SSLFreeBuffer(&signedHashes);
19     SSLFreeBuffer(&hashCtx);
20     return err;
21 }
```



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

```
int main () {
```

```
    char buf[20] = "http://www.diku.dk";
    char shh[30] = "mumstheword";
    char out[64];
    int chars;
```

```
    printf("Buffer contents: %s\n", buf);
```

```
    printf("Chars to copy: ");
    scanf("%d", &chars);
```

```
    memcpy(out, buf, chars);
```

```
    printf("Copied: ");
    fwrite(out, chars, 1, stdout);
    printf("\n");
```

```
$ ./a.out
```

```
Buffer contents: http://www.diku.dk
```

```
Chars to copy: 12
```


```
Copied: http://www.d
```

```
$ ./a.out
```

```
Buffer contents: http://www.diku.dk
```

```
Chars to copy: 50
```

```
Copied: http://www.diku.dk0LHmumstheword
```

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

```
int main () {
```

```
    char buf[20] = "http://www.diku.dk";
    char shh[30] = "mumstheword";
    char out[64];
    int chars;
```

```
    printf("Buffer contents: %s\n", buf);
```

```
    printf("Chars to copy: ");
    scanf("%d", &chars);
```

```
    if (chars > sizeof(buf)) chars = sizeof(buf);
    memcpy(out, buf, chars);
```

```
    printf("Copied: ");
    fwrite(out, chars, 1, stdout);
    printf("\n");
```

```
$ ./a.out
```

```
Buffer contents: http://www.diku.dk
```

```
Chars to copy: 12
```

```
Copied: http://www.d
```

```
$ ./a.out
```

```
Buffer contents: http://www.diku.dk
```

```
Chars to copy: 50
```


```
Copied: http://www.diku.dk0LHmumstheword
```

The HeartBleed Bug



Heartbeat – Normal usage





```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char **argv)
{


    printf("Current time: ");
    fflush(stdout);
    system("date");
    return 0;

}
```

```
$ ./a.out
Current time: Fri Sep  6 09:30:47 CEST 2019
```

```
$ export PATH=`pwd`: $PATH
$ echo -e '#!/bin/sh\nnecho "Hello"' > date
$ chmod 700 date
```

```
$ ./a.out
Current time: Hello
```



```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv)
{

    printf("Current time: ");
    fflush(stdout);
    system("/bin/date");
    return 0;

}
```

```
$ ./a.out
Current time: Fri Sep  6 09:30:47 CEST 2019
```

```
$ export PATH=`pwd`: $PATH
$ echo -e '#!/bin/sh\nnecho "Hello"' > date
$ chmod 700 date
```

```
$ ./a.out
Current time: Hello
```



Real-world example: PlugX


PlugX drops

- A legitimate NVIDIA file (NvSmart.exe)

- A malicious DLL (NvSmartMax.dll)

Normally, NvSmart.exe would load a legitimate NvSmartMax.dll

But if a (malicious) version the DLL file is located in the same directory, this will load instead



```
#!/usr/bin/perl

open(FH, $ARGV[0]);

while(<FH>)
{
    print $_;
}

close(FH);
```

```
$ ./code.pl code.pl
#!/usr/bin/perl

open(FH, $ARGV[0]);

while(<FH>)
{
    print $_;
}

close(FH);

$ ./code.pl 'ls -l code.pl'
-rwx----- 1 user user 79 Sep  1 10:45 code.pl
```



```
#!/usr/bin/perl
```

```
open(FH, "< ".$ARGV[0]); #force read open with '<'
```

```
while(<FH>)  
{  
    print $_;  
}
```

```
close(FH);
```

```
$ ./code.pl code.pl  
#!/usr/bin/perl
```

```
open(FH, $ARGV[0]);
```

```
while(<FH>)  
{  
    print $_;  
}
```

```
close(FH);
```


```
$ ./code.pl 'ls -l code.pl'  
-rwx----- 1 user user 79 Sep  1 10:45 code.pl
```



Explanation

According to the Perl documentation

If filename ends with a "|", filename is interpreted as a command which pipes output



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

int main (int argc, char **argv)
{
    char buffer[64];
    strncpy(buffer, argv[1], sizeof(buffer));


    printf("You entered: ");
    printf(buffer);
    printf("\n");
}
```

```
$ ./a.out A
You entered: A
```

```
$ ./a.out %s
You entered: You entered:
```

```
$ ./a.out %x
You entered: 510a2000
```

```
$ ./a.out %x%x
You entered: 437a00041569e0
```



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

int main (int argc, char **argv)
{
    char buffer[64];
    strncpy(buffer, argv[1], sizeof(buffer));


    printf("You entered: ");
    printf("%s", buffer);
    printf("\n");
}
```

```
$ ./a.out A
You entered: A
```

```
$ ./a.out %s
You entered: You entered:
```

```
$ ./a.out %x
You entered: 510a2000
```

```
$ ./a.out %x%x
You entered: 437a00041569e0
```



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


```
void foo (char *bar)
{
    char c[12];
    strcpy(c, bar);
}
```

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

```
$ ./6.out A
```

```
$ ./6.out AAAAAAAAAAAAAAAAAA
```

```
$ ./6.out AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault
```



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

```
void foo (char *bar)
{
    char c[12];
    strncpy(c, bar, sizeof(c));
}
```

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

```
$ ./6.out A
```

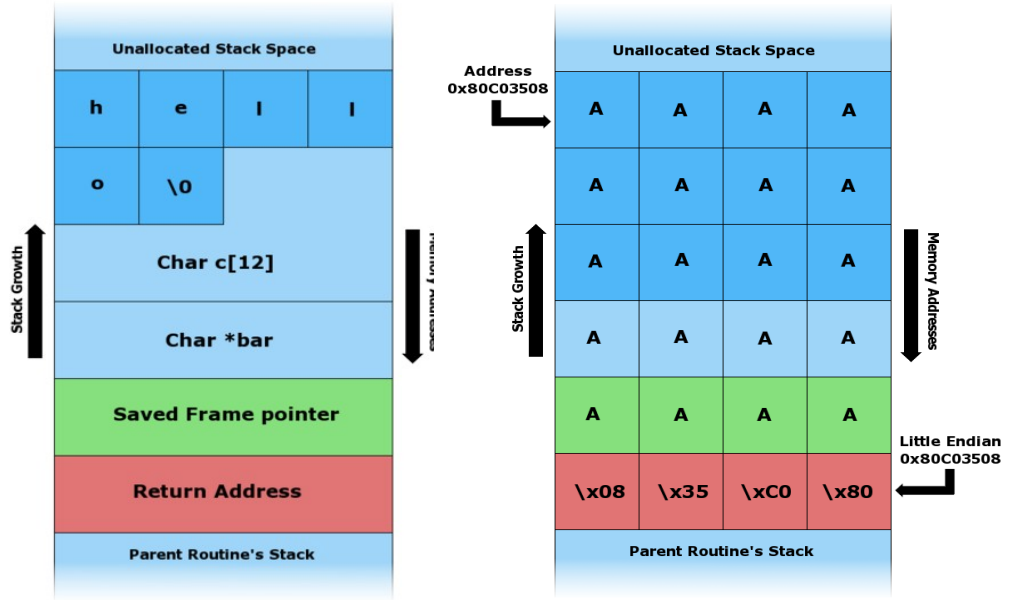
```
$ ./6.out AAAAAAAAAAAAAAA
```

```
$ ./6.out AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault
```

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

```
void foo (char *bar)
{
    char c[12];
    strcpy(c, bar);
}
```

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





Some countermeasures

Stack canaries

Check stack not altered when function returns

Data execution prevention (DEP)

Prevent the execution of data on the stack or heap

Address space layout randomization (ASLR)

Rearrange memory positions to make successful exploitation more difficult

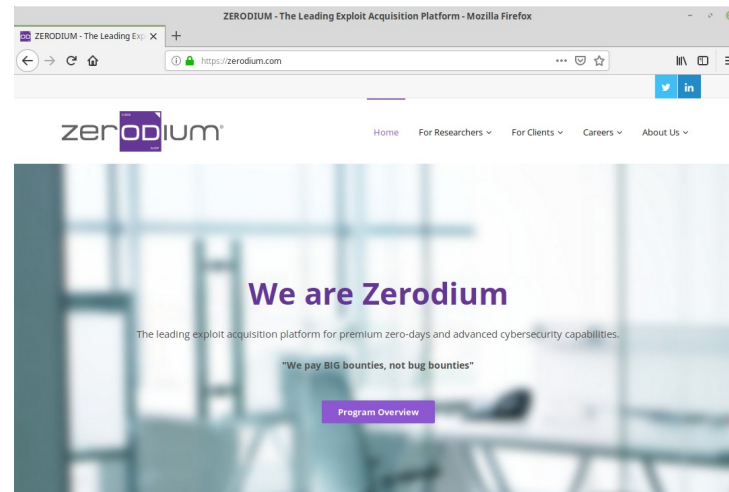
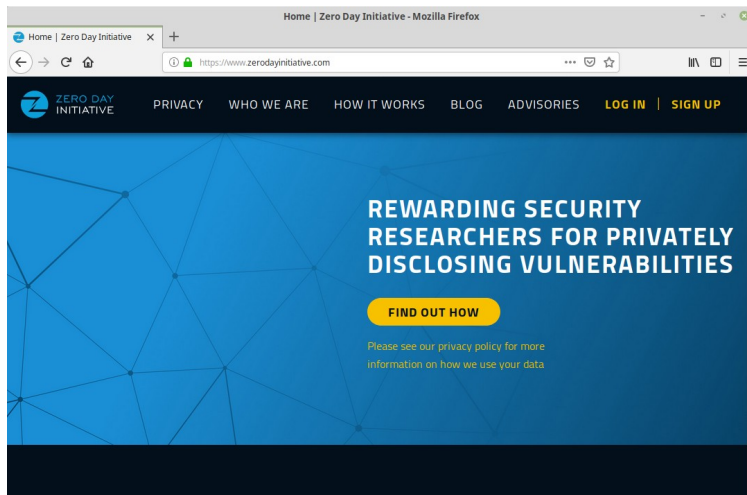


Okay, so you've found a bug

Options



Options



Zerodium

ZERODIUM Payouts for Mobiles*

ZERODIUM Payouts for Mobiles*

Up to \$2,500,000

Up to \$2,000,000

Up to \$1,500,000

Up to \$1,000,000

Up to \$500,000

Up to \$200,000

Up to \$100,000

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

IOS

Android

Any OS

1.001

Android FCP
Zero Click

Android

1.002

iOS FCP
Zero Click

iOS

2.001

WhatsApp
RCE+LPE
Zero Click

iOS/Android

2.002

iMessage
RCE+LPE
Zero Click

iOS

2.003

WhatsApp
RCE+LPE

iOS/Android

2.004

SMS/MMS
RCE+LPE

iOS/Android

3.001

Persistence

iOS

2.005

WeChat
RCE+LPE

iOS/Android

2.006

iMessage
RCE+LPE

iOS

2.007

FB Messenger
RCE+LPE

iOS/Android

2.008

Signal
RCE+LPE

iOS/Android

2.009

Telegram
RCE+LPE

iOS/Android

2.010

Email App
RCE+LPE

iOS/Android

4.001

Chrome
RCE+LPE

Android

4.002

Safari
RCE+LPE

iOS

5.001

Baseband
RCE+LPE

iOS/Android

6.001

LPE to
Kernel/Root

iOS/Android

2.011

Media Files
RCE+LPE

iOS/Android

2.012

Documents
RCE+LPE

iOS/Android

4.003

SBX
for Chrome

Android

4.004

Chrome RCE
w/o SBX

Android

4.005

SBX
for Safari

iOS

4.006

Safari RCE
w/o SBX

iOS

7.001

Code Signing
Bypass

iOS/Android

5.002

WiFi
RCE

iOS/Android

5.003

RCE
via MitM

iOS/Android

6.002

LPE to
System

Android

8.001

Information
Disclosure

iOS/Android

8.002

[k]ASLR
Bypass

iOS/Android

9.001

PIN
Bypass

Android

9.002

Passcode
Bypass

iOS

9.003

Touch ID
Bypass

iOS

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

■ iOS
■ Android
■ Any OS

* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.

2019/09 © zerodium.com



“A patch is out”

Patching challenges

Magento Flaw Exploited in the Wild Within 24 Hours After Disclosure

By [Eduard Kovacs](#) on April 24, 2015

[Tweet](#)



Malicious actors are attempting to hijack online shops by exploiting a recently disclosed critical [vulnerability in Magento](#), the popular e-commerce platform owned by eBay.

