In the Boxing Ring September 2020

Network Box Technical News

from Mark Webb-Johnson
Chief Technology Officer, Network Box

Welcome to the September 2020 edition of In the **Boxing Ring**

This month, we are talking about **Bugs, Crashes, and Vulnerabilities**. Unfortunately, in modern computing, an application or operating system crash seems to be the norm. We have all become used to frequently saving our work, hitting that 'Relaunch' button, and carrying on from where we were interrupted. On pages 2 to 3, we look at what causes an application to crash and highlight the security implications of it.

In other news, Network Box Germany has partnered with **F-Secure** and **Computer Insider GmbH** to expand security offerings. And in this month's Media Coverage, Network Box was featured in the **SCMP**, **Asia Times**, and Network Box has been nominated in the 2020 **Funkschau Readers Choice Awards** for *IT Product of the Year*.

Mark Webb-Johnson
CTO, Network Box Corporation Ltd.
September 2020

Stay Connected

You can contact us here at Network Box HQ by email: **nbhq@network-box.com**, or drop by our office next time you are in town. You can also keep in touch with us by several social networks:

https://twitter.com/networkbox

https://www.facebook.com/networkbox
https://www.facebook.com/networkboxresponse

https://www.linkedin.com/company/network-box-corporation-limited/

https://www.youtube.com/user/NetworkBox

In this month's issue:

Page 2 to 3

Bugs, Crashes, and Vulnerabilities

Usually, programs crash due to some unexpected input, and there is good reason to be concerned, as that crash can be the route into your system for a malicious hacker. In our featured article, we discuss this in further detail.

Page 4

Network Box Highlights:

- Network Box Germany F-Secure and Computer Insider GmbH partnerships
- **Network Box Media Coverage:**
  - South China Morning Post
  - Asia Times
  - Funkschau Read Choice Awards 2020

NOTE: With effect from January 2020 we have switched to a quarterly Patch Tuesday cycle for Network Box 5. However, essential security fixes will continue to be released out-of-cycle, if necessary.

Bugs, Crashes, and Vulnerabilities

An application or operating system crash seems to be the norm with modern computers. We have all become used to frequently saving our work, hitting that 'Relaunch' button, and carrying on from where we were interrupted.

But have you ever wondered about what causes the crash, and if there are any more significant implications? Well, there is good reason to be concerned. Usually, programs crash due to some unexpected input, and that crash can be the route into your system for a malicious hacker.

What causes applications to crash?

There can be many reasons, but the vast majority is because the program did something that the operating system did not like. Either some system function was called with invalid parameters, or the application tried to access resources it should not have access to. The most common is an 'out of bounds' memory access, where the application is attempting to access memory out of it's permitted ownership.

It is simple for a programmer to ask for some input (such as a user name), and then copy that input to a storage variable for later use. In the popular 'C' programming language, it looks like this:

```
char username[32];
strcpy(username,input);
```

The strcpy() function is historically very commonly used, and copies a string from the second parameter (`input`) to the area of memory pointed to by the first (`username`). It stops the copy when it reaches an end-of-string marker (the byte zero).

But, what if the input provided is bigger than 32 characters (or 31, as the zero terminating byte, takes up one)? The answer is that strcpy() doesn't care and will carry on copying. Whatever comes after the username will be overwritten. If the username variable is at the top of memory, that will result in a 'memory out of bounds' crash. Even if that is not the case, other things may get overwritten and mess up the program flow or cause other problems.

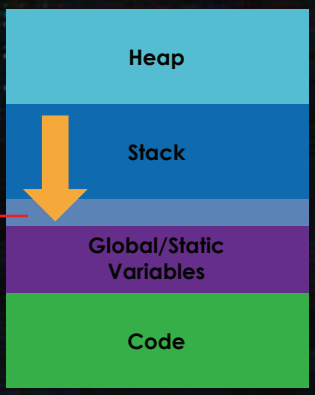
Another common approach is for the input to come in the form of a protocol message in a network communication stream. In many cases, such protocols encode data as <length><message>, sending the length to expect first, followed by that number of bytes of the variable-length message. What if an attacker hand-crafts protocol messages, with malicious lengths designed to overwrite memory arbitrarily?

Of course, program developers should thoroughly validate such cases. Looking at the history of such vulnerabilities, it is evident that some (perhaps most) do not.

What are the security implications?

Let's look at a typical computer memory architecture in use today. Code and data reside in the same memory. Code is at the bottom, global variable memory sits above that, and the stack is at the top. There may also be an area called the 'heap' for dynamic memory (that can be extended upwards, if necessary). The key here is that code and the global variables are fixed in size, but the stack grows downwards. It looks like this:

Typical computer memory architecture

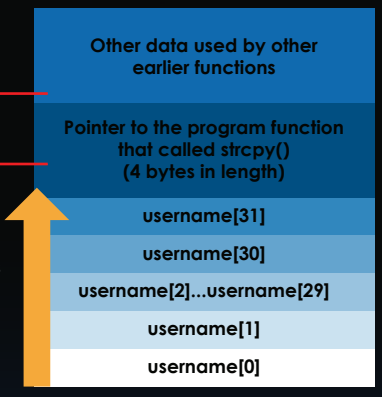


The stack grows downwards as more functions are added to the memory

The stack is an interesting piece of technology. Let's look at our strcpy() example above. What happens is that when the program calls the strcpy() function, the current address of the program is 'pushed' onto the stack, and then control flows to the strcpy() function itself. When that function has done its work, it calls the 'return' function, which 'pops' the return address off the stack and resumes control flow at that point. This scheme allows for functions to call other functions in a nested fashion only limited by the stack's size.

It is also common for local variables, such as the username, to be stored on the stack. Let's look at the memory arrangement for our original strcpy() program while running the strcpy() function:

The Stack



Payload Target

Payload Pointer

The *username* data goes up the stack to the Payload Target area

Let's say the attacker delivers a 'payload' as input:

- 32 bytes of username (no zero characters)
- A 4 byte pointer to the payload target (just above)
- The rest of the payload containing code to be executed
- Zero termination to end the strcpy()

Now, the strcpy() function dutifully copies all that into the username, overwrites the return pointer, and then overwrites the upper stack with the payload provided. The strcpy() function returns, and the CPU dutifully pops the return address (now pointing to the code in the payload provided by the attacker) and transfers control to the attacker. Game over, and the attacker wins.

The good news

The above sounds bad. But the good news is that nowadays, it is not so simple. There is an ongoing cat-and-mouse game between defenders and attackers, with the defenders adding more and more complications and controls to make things harder for the attackers. Technologies such as:

- Tagging different memory regions as not executable, such as the stack
- Address randomization, so the memory layout is not predictable
- Stack canaries, unique values on the stack that if modified signal stack corruption

These all make it a lot harder for an attacker to succeed with such attacks.

Industry partnerships (such as the Microsoft Active Protection Program - MAPP - which Network Box participants in) also exist to share detailed vulnerability information. Security providers like Network Box can release protections to give users time to update their applications.

Nowadays, most popular operating systems allow you to log in and use the system as a low privilege user (only escalating to administrative privileges when required). Logging in like this initially limits the impact of any exploit to the user's access rights, again gaining time.

That said, new vulnerabilities are announced every day (the most popular tracking service currently has more than 140,000 in its database), and while not all are exploitable, some are. The rewards for a successful zero-day exploit (one that nobody else knows about) can be in the hundreds of thousands of dollar range.

So, the next time a program crashes on you, perhaps it would be a good time to 'check for updates' and see if a fix has been released. You can also opt-in to share crash data with the application developers so that they can be notified of such problems and make iterative improvements.

Network Box HIGHLIGHTS

Network Box Germany
F-Secure and Computer Insider GmbH Partnerships

Network Box Germany has partnered with Finnish Anti-Virus specialist, **F-Secure**, further expanding security offerings with Rapid Detection & Response, Microsoft Office 365 Cloud Protection, and Premium Endpoint-Protection.

Furthermore, Network Box Germany has partnered with **Computer Insider GmbH**, specialists in IT in medical practices, law firms, and home offices.

Newsletter Staff	Subscription
Mark Webb-Johnson Editor **Michael Gazeley** **Kevin Hla** Production Support **Network Box HQ** **Network Box USA** Contributors	Network Box Corporation nbhq@network-box.com or via mail at: **Network Box Corporation** 16th Floor, Metro Loft, 38 Kwai Hei Street, Kwai Chung, Hong Kong Tel: +852 2736-2083 Fax: +852 2736-2778 www.network-box.com

Network Box Media Coverage and Security Headlines

SCMP
Nearly 235 million social media profiles from Instagram, TikTok and YouTube exposed in data leak
LINK: https://bit.ly/3lyoyiF

Asia Times
The problem of Internet domain abuse
LINK: https://bit.ly/3bdaeY6

Funkschau
Network Box nominated in the funkschau reader choice for the IT product of the year 2020
LINK: https://bit.ly/3lxCdqb

BBC
Tea at the Ritz soured by credit card scammers
LINK: https://bbc.in/32HOnEe

ZDNet
Patch now: Cisco warns of nasty bug in its data center software
LINK: https://zd.net/32GLKm1