程序代写代做 CS编程辅导

FORMAT STRING ATTACK

WeChat: cstutorcs

Assignment Project Exam Help

Martin Read

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Lecture aim

程序代写代做 CS编程辅导

Introduction to formatting attack

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Lecture Objectives

1. What happens when formatting of data allows attackers to control the data?
2. Example programmes ...

- Practical next week

Nothing is Secure

程序代写代做 CS编程辅导

- Finding the vulnerabili  ing it will increase the quality & efficiency of software
- Every programming language has its own pros & cons
- Some developers claim that there are some languages that are more or less secure than other languages

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Top 5 Vulnerable Programming Languages

程序代写代做 CS编程辅导

According to a recent report the most widely used & vulnerable programming languages

- C [47%]
- PHP [17%]
- Java [12%]
- JavaScript [11%]
- Python and C++ [6%]

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

WhiteSource, a security research company

https://www.whitesourcesoftware.com/most-secure-programming-languages/

https://medium.com/hackernoon/top-5-vulnerable-programming-languages-eab3144d6db7

https://developers.slashdot.org/story/19/03/25/0322202/which-programming-language-has-the-most-security-vulnerabilities

C programming vulnerabilities

程序代写代做 CS编程辅导

The most common vuln  es:

- Buffer Overflow Error: Most popular buffer overflows are:
  - Stack-based buffer overflow
  - Heap-based buffer overflow
- Format String Vulnerability
  - C/C++ languages mostly prone to format string attack
  - other modern languages, eg C#, Java, etc won't typically allow the execution of arbitrary code

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

程序代写代做 CS编程辅导

- C/C++'s internal design makes it harder to detect format string problems - including s  pecially dangerous commands that do not exist in some o nguages' format string languages
- A successful attack ca lead to the execution of arbitrary code, & to information disclosure

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

- Unfortunately, many programmers inadvertently committed a simple mistake while coding, leading to coding loopholes
  - Typically, allowing data from untrusted source
  - format string intruders then write unsolicited format strings to cause serious harm

Strings Functions

程序代写代做 CS编程辅导

- Assembly only provides basic functionality
- C provides functions to ease interaction with humans
  - Moving strings
  - Storing and loading strings
  - Comparing strings
  - Scanning
  - Finding string length
- Character is 1 byte long in C
  - What more do we need to represent strings?

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Working with strings

程序代写代做 CS编程辅导

$ ./format_error "Hello World"

Hello World

$ ./format_error "Go Navy"

Go Navy

WeChat: cstutorcs

What happens when you give a format character?

Assignment Project Exam Help

$ ./format_error "%x"

Email: tutorcs@163.com

b7fff000

QQ: 749389476

This is interpreted & the output is an address on the stack

https://tutorcs.com

Working with strings

程序代写代做 CS编程辅导

- What if you were to give it something longer?
- What if you were to give it something that would cause a memory address to be dereferenced, like a '%s' :

```
$ ./format_error "%s.%s.%s.%s.%s.%s.%s"
        4.??u?.UW1?VS???????(null).(null).?$?U?
$ ./format_error "%s.%s.%s.%s.%s.%s.%s.%s.%s.%s"
        Segmentation fault (core dumped)
```

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

- Can actually get the program to crash
  - getting the program to crash usually first step towards exploiting the program…

Format String Attacks

程序代写代做 CS编程辅导

- An alternate form of exploiting programming that doesn't necessarily require smashing the stack
  - leverages format characters in a format string to generate excessive data, read from arbitrary memory or write to arbitrary memory

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

- printf() & scanf() family of functions have formatting to define output/input
  - fprint, printf, sprintf, snprintf, vfprintf, vprintf
  - Many programs allow attackers to control the data in the function

Format Parameters

程序代写代做 CS编程辅导

```
printf ("The magic number is: %d\n", 1911);
```

- Behaviour of the formatting function controlled by the format string
  - retrieves parameters requested by format string from stack

```
printf ("a has value %d, b has value %d, c is at address: %08x\n",
a, b, &c);
```

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

- Format string parameters are used to determine the data type of an input
- **Parameter**
  - %d - Value decimal (int)
  - %u - Value unsigned decimal (int)
  - %x - Value hexadecimal (int)
  - %s - Pointer string
  - %n - Pointer number of bytes written so far

Format String Attacks

程序代写代做 CS编程辅导

- printf() – output formatter
  - Attacker can make excessive output – leading to buffer overflow
  - Attacker can expose stack data
  - %n lets attacker overwrite arbitrary memory

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

- scanf() – input formatter
  - Attacker can accept too much data – leading to buffer overflow
  - Attacker can determine what data enters system

- Related to Uncontrolled Format String (qv)

Format String Vulnerability

程序代写代做 CS编程辅导

- Format string exploits can be used to gain control of a program

  printf("A is %d and is at address %08x. B is %x.\|n", A, &A, B);

- What if you provided the wrong number of parameters?

  printf("A is %d and is at %08x. B is %x.\|n", A, &A);

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Can this program pass the compiler?

- Sometimes, the format string is not a constant string - generated during execution
- Therefore - no way for the compiler to find the mis-match, in this case

Format String Vulnerability

程序代写代做 CS编程辅导

Can *printf()* detect the mis-match?

- printf() fetches the arguments from the stack
  - If format string needs 3 arguments - will fetch 3 data items from stack
  - Unless stack marked with a boundary, *printf()* does not know that it runs out of the arguments provided
- *printf()* will continue fetching data from the stack
  - In a mis-match case, it will fetch data that does not belong to this function call

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Format String Vulnerability

程序代写代做 CS编程辅导

- So, incorrect formatting could cause format string vulnerabilities
  E.g. *printf(string)*, ra[illegible]an *printf("%s", string)*

  - print function will still display the string, but the format function is passed the address of the string, not address of a format string
  - Could cause stack pointer to reference a piece of memory in a preceding stack frame

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Reading from Arbitrary Addresses

程序代写代做 CS编程辅导

- %s format could be used to read from arbitrary memory addresses
  - Part of the original string can be used to supply an address to the %s format parameter

    $ ./fmt_vuln AAAA%08x.%08x.%08x.%08x

  - AAAA indicates that the fourth format parameter is reading from the beginning of the format string

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

- What if the fourth format parameter is %s instead of %x?
  - It will attempt to print the string located at 0x41414141

Writing to Arbitrary Memory Addresses

程序代写代做 CS编程辅导

- %s format could be used to read from arbitrary memory addresses
- Can write to an arbitrary address with the %n parameter:

```
$ ./fmt_vuln $(printf "\x94\x97\x04\x08")%x%x%400x%n
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

- Resulting value depends on number of bytes written before %n
- However, starting with Visual Studio 2005, the capability of using %n is **off** by default
  - To perform this attack, would have to explicitly allow this specifier

Direct Parameter Access

程序代写代做 CS编程辅导

- Previous examples required sequential attempts to pass format parameter arguments
- To simplify format strings and edits, we can use direct parameter access
  - Allows parameters to be accessed directly using the dollar sign qualifier

    e.g. %n$d will access the nth parameter and display it as a decimal number

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

printf("7th: %7$d, 4th: %4$05d\n", 10, 20, 30, 40, 50, 60, 70, 80);

will print:

7th: 70, 4th: 00040

printf("%s%s%s%s%s%s%s%s%s%s%s%s");

程序代写代做 CS编程辅导

For each %s, printf() w a number from the stack, treat it as an address & print o memory contents pointed by this address as a string

- until a NULL character (i.e. number 0, not character 0) found
- Number fetched by printf() might not be an address
  - memory pointed by this number might not exist (i.e. no physical memory has been assigned to such an address)
    & program will crash
- Also possible that the number is a good address, but address space is protected (e.g. reserved for kernel memory)
  - so program will also crash!

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Viewing the stack

程序代写代做 CS编程辅导

printf("%08x %08x %08x %08x %08x\n");

- Instructs function to r[illegible]5 parameters from stack & display them as 8-digit padded hexadecimal numbers

WeChat: cstutorcs

- A possible output may look like:

Assignment Project Exam Help

40012980 080628c4 bffff7a4 00000005 08059c04

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Viewing memory at any location

程序代写代做 CS编程辅导

- We have to supply an address to the memory. However, we cannot change the co[illegible] only supply the format string
- If we use printf(%s) w[illegible] specifying a memory address, the target address will be obtained from the stack anyway by printf()
  - Function maintains an initial stack pointer, so it knows location of parameters in stack
- Observation: format string is usually located on the stack
  - If we can encode the target address in the format string, the target address will be in the stack
  - In the following example, the format string is stored in a buffer, which is located on the stack

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

程序代写代做 CS编程辅导

```
int main(int argc, char *
{
        char user_input[100];
        ... ...  /* other variable definitions and statements */
        scanf("%s", user_input);        /* getting a string from user */
        printf(user_input);             /* Vulnerable place */
        return 0;
}
```

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

The Stack and Format Strings

程序代写代做 CS编程辅导

- If we can force printf to obtain the address from the format string (also on the stack), we can control the address

  printf ("\x10\x01\x48\x08 %x %x %x %x %s");

  \x10\x01\x48\x08 are the four bytes of the target address

- In C: \x10 in a string tells compiler to put a hexadecimal value 0x10 in the current position. Value will take up just one byte
- Without using \x, if we directly put "10" in a string, the ASCII values of the characters '1' & '0' will be stored – (49 & 48)
- %x causes stack pointer to move towards the format string
- %s passed to printf(), causing it to print out the contents in the memory address 0x10014808
- printf() will treat the contents as a string & print out the string until reaching the end of the string (i.e. 0)

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Writing an integer to nearly any location in process memory

程序代写代做 CS编程辅导

- %n: The number of characters written so far is stored into the integer indicated by the corresponding argument

```
printf ("12345%n", &i);
```

- Output = 12345, but causes printf() to write 5 into variable i
- Using the same approach as that for viewing memory at any location, we can cause printf() to write an integer into any location
- Contents at the address 0x19814808 will be overwritten

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

程序代写代做 CS编程辅导

Using this attack, attacker can do the following:

- Overwrite important program flags that control access privileges
- Overwrite return addresses on the stack, function pointers, etc.

- However, the value written is determined by the number of characters printed before %n reached. Is it really possible to write arbitrary integer values?
  - Use dummy output characters. To write a value of 1000, a simple padding of 1000 dummy characters would do
  - To avoid long format strings, we can use a width specification of the format indicators

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

程序代写代做 CS编程辅导

- So, even such a simple construct as *printf(name)* can be dangerous!

- What you think is safe code might contain a vulnerability
  - If you don't see a catch in your code, it doesn't mean there isn't any
- Follow all the compiler's recommendations on using updated versions of string functions
- Even better if you do not use low-level string handling
  - functions are a heritage of the C language
- Now there is *std::string* & safe methods of string formatting, such as *boost::format* or *std::stringstream*

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Countermeasures

程序代写代做 CS编程辅导

- Address randomization:
  - just like the counte[illegible]res used to protect against buffer-overflow attacks, a[illegible] randomization makes it difficult for attackers to find out what address they want to read/write

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

You should give consideration to methods of getting private data.

When developing software containing variable-argument functions, think about if there are cases when they may be the source of data leak

It could be a log-file, a batch passed on the network & the like

Be careful when external data is input into your program - manage what & where it is written into memory

Countermeasures

程序代写代做 CS编程辅导

The important message is this: if programs are not correctly coded to protect their data structures, attacks on them are possible…

- Whilst the defences can block many such attacks, some - like corrupting an adjacent variable value in a manner that alters the behaviour of the attacked program - cannot be blocked, except by coding to prevent them occurring in the first place

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Summary

程序代写代做 CS编程辅导

Introduction to formatting attack

1. What happens when formatting of data allows attackers to control the data?
2. Example programmes ...

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

FURTHER READING

程序代写代做 CS编程辅导

- Hacking: The art of exploitation, section 0x350, pg 167-193

If you fancy taking this further, take a look at these webpages from the US Naval Academy

https://www.usna.edu/Users/cs/aviv/classes/si485h/s17/units/06/unit.html

https://www.usna.edu/Users/cs/aviv/classes/si485h/s17/units/05/unit.html

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com