Owning Stuff With Format Strings

deanx

Intercon II

13th June 2008

1 yin

- what are they
- syntax

2 yang

- what are they
- syntax

what?

Format strings are a way of secifying the format when data is outputted

what?

Format strings are a way of secifying the format when data is outputted

- dave is 20

what?

Format strings are a way of secifying the format when data is outputted

- dave is 20
- %s is %d

what?

Format strings are a way of secifying the format when data is outputted

- dave is 20
- %s is %d
- printf(" %s is %d", name, age)

special characters

These are used to format the output of variables

special characters

These are used to format the output of variables

- %d - int

special characters

These are used to format the output of variables

- %d - int
- %f - float

special characters

These are used to format the output of variables

- %d - int
- %f - float
- %u - unsigned int

special characters

These are used to format the output of variables

- %d - int
- %f - float
- %u - unsigned int
- %x - hex representation of an int

special characters

These are used to format the output of variables

- %d - int
- %f - float
- %u - unsigned int
- %x - hex representation of an int
- %s - string (prints till \x00)

modifyers

Used to more specifically control how variables are represented

modifyers

Used to more specifically control how variables are represented

- %.5f - 3.14159

modifyers

Used to more specifically control how variables are represented

- %.5f - 3.14159
- %#x - 0xdeadbeaf

modifyers

Used to more specifically control how variables are represented

- %.5f - 3.14159
- %#x - 0xdeadbeaf
- %4u - xxxx4

modifyers

Used to more specifically control how variables are represented

- %.5f - 3.14159
- %#x - 0xdeadbeaf
- %4u - xxxx4
- %hx - beaf

modifyers

Used to more specifically control how variables are represented

- %.5f - 3.14159
- %#x - 0xdeadbeaf
- %4u - xxxx4
- %hx - beaf
- %hhx - af

modifyers 2

- dave is 20, dave is male

modifyers 2

- dave is 20, dave is male
- %s is %d, %s is %s

modifyers 2

- dave is 20, dave is male
- %s is %d, %s is %s
- printf(“ %s is %d, %s is %s”,name, age, name, sex)

modifyers 2

- dave is 20, dave is male
- %s is %d, %s is %s
- printf(“ %s is %d, %s is %s”,name, age, name, sex)
- %1$s

modifyers 2

- dave is 20, dave is male
- %s is %d, %s is %s
- printf(“ %s is %d, %s is %s”,name, age, name, sex)
- %1$s
- printf(“ %s is %d, %1$s is %s”,name, age, sex)

what

They can be misused if programmers forget about them

what

They can be misused if programmers forget about them

- dave

what

They can be misused if programmers forget about them

- dave
- printf(name)

what

They can be misused if programmers forget about them

- dave
- printf(name)
- %s

what

They can be misused if programmers forget about them

- dave
- printf(name)
- %s
- demo

reading memory

reading memory

- %x

reading memory

- %x
- %x%x%x%x%x%x%x%x%x%x%%x%x%x%x%x%x%x%x%x

reading memory

- %x
- %x%x%x%x%x%x%x%x%x%%x%x%x%x%x%x%x%x%x
  - aloadofcrap

reading memory

- %x
- %x%x%x%x%x%x%x%x%x%%x%x%x%x%x%x%x%x%x
  - aloadofcrap
- aaaa-%x%x%x%x%x%x%x%x%x%x

reading memory

- %x
- %x%x%x%x%x%x%x%x%x%%x%x%x%x%x%x%x%x%x
  - aloadofcrap
- aaaa-%x%x%x%x%x%x%x%x%x%x
  - aaaa-aloadofcrap61616161morecrap

reading memory

- %x
- %x%x%x%x%x%x%x%x%x%%x%x%x%x%x%x%x%x%x
  - aloadofcrap
- aaaa-%x%x%x%x%x%x%x%x%x%x
  - aaaa-aloadofcrap61616161morecrap
- aaaa-%8$#x

reading memory

- %x
- %x%x%x%x%x%x%x%x%x%%x%x%x%x%x%x%x%x%x
  - aloadofcrap
- aaaa-%x%x%x%x%x%x%x%x%x%x
  - aaaa-aloadofcrap61616161morecrap
- aaaa-%8$#x
  - aaaa-0x61616161

reading memory

- %x
- %x%x%x%x%x%x%x%x%x%%x%x%x%x%x%x%x%x%x
  - aloadofcrap
- aaaa-%x%x%x%x%x%x%x%x%x%x
  - aaaa-aloadofcrap61616161morecrap
- aaaa-%8$#x
  - aaaa-0x61616161
- demo

writing memory

writing memory

- Introducing %n

writing memory

- Introducing %n
  - %n will write the number of bytes written to the pointer location specified

writing memory

- Introducing %n
  - %n will write the number of bytes written to the pointer location specified
- dave has 4 characters in his name

writing memory

- Introducing %n
  - %n will write the number of bytes written to the pointer location specified
- dave has 4 characters in his name
- printf("%s%n has %d characters in his name", name, &chars, chars)

writing memory

- Introducing %n
  - %n will write the number of bytes written to the pointer location specified
- dave has 4 characters in his name
- printf("%s%n has %d characters in his name", name, &chars, chars)
- printf("%s%x has %d characters in his name", name, &chars, chars)

writing memory

- Introducing %n
  - %n will write the number of bytes written to the pointer location specified
- dave has 4 characters in his name
- printf("%s%n has %d characters in his name", name, &chars, chars)
- printf("%s%x has %d characters in his name", name, &chars, chars)
- demo

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS
  - \xef\xbe\xad\xde

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS
  - \xef\xbe\xad\xde
- We have now written 4 chars to the screen

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS
  - \xef\xbe\xad\xde
- We have now written 4 chars to the screen
- So we know need to write another (90909090h-4) or

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS
  - \xef\xbe\xad\xde
- We have now written 4 chars to the screen
- So we know need to write another (90909090h-4) or
- we use %hn or %hhn so we can write it in 2 or 1 byte sections

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS
  - \xef\xbe\xad\xde
- We have now written 4 chars to the screen
- So we know need to write another (90909090h-4) or
- we use %hn or %hhn so we can write it in 2 or 1 byte sections
- Lets choose %hn, so 2 byte chunks.

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS
  - \xef\xbe\xad\xde
- We have now written 4 chars to the screen
- So we know need to write another (90909090h-4) or
- we use %hn or %hhn so we can write it in 2 or 1 byte sections
- Lets choose %hn, so 2 byte chunks.
- So now we write

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS
  - \xef\xbe\xad\xde
- We have now written 4 chars to the screen
- So we know need to write another (90909090h-4) or
- we use %hn or %hhn so we can write it in 2 or 1 byte sections
- Lets choose %hn, so 2 byte chunks.
- So now we write
  - 9090h (370008) to 0xdeadbeef

controlling memory writes

We want to overwrite 0xdeadbeef with 0x90909090

- We push 0xdeadbeef onto the stack as the first part of our FS
  - \xef\xbe\xad\xde
- We have now written 4 chars to the screen
- So we know need to write another (90909090h-4) or
- we use %hn or %hhn so we can write it in 2 or 1 byte sections
- Lets choose %hn, so 2 byte chunks.
- So now we write
  - 9090h (370008) to 0xdeadbeef
  - 9090h (370008) to 0xdeadbef1

generating the FS

We have a stack-pop length of 8 using 2x2 byte writes

- Push the addresses to the stack

generating the FS

We have a stack-pop length of 8 using 2x2 byte writes

- Push the addresses to the stack
  - \xef\xbe\xad\xde\xf1\xbe\xad\xde

generating the FS

We have a stack-pop length of 8 using 2x2 byte writes

- Push the addresses to the stack
  - \xef\xbe\xad\xde\xf1\xbe\xad\xde
- So now we have written 8 bytes so only another 37000 to go

generating the FS

We have a stack-pop length of 8 using 2x2 byte writes

- Push the addresses to the stack
  - \xef\xbe\xad\xde\xf1\xbe\xad\xde
- So now we have written 8 bytes so only another 37000 to go
  - %37000u - simple

generating the FS

We have a stack-pop length of 8 using 2x2 byte writes

- Push the addresses to the stack
  - \xef\xbe\xad\xde\xf1\xbe\xad\xde
- So now we have written 8 bytes so only another 37000 to go
  - %37000u - simple
- \xef\xbe\xad\xde\xf1\xbe\xad\xde%37000u

generating the FS

We have a stack-pop length of 8 using 2x2 byte writes

- Push the addresses to the stack
  - \xef\xbe\xad\xde\xf1\xbe\xad\xde
- So now we have written 8 bytes so only another 37000 to go
  - %37000u - simple
- \xef\xbe\xad\xde\xf1\xbe\xad\xde%37000u
- now we use %8$hn and %9$hn to write the data

generating the FS

We have a stack-pop length of 8 using 2x2 byte writes

- Push the addresses to the stack
  - \xef\xbe\xad\xde\xf1\xbe\xad\xde
- So now we have written 8 bytes so only another 37000 to go
  - %37000u - simple
- \xef\xbe\xad\xde\xf1\xbe\xad\xde%37000u
- now we use %8$hn and %9$hn to write the data
- \xef\xbe\xad\xde\xf1\xbe\xad\xde%37000u%8$hn%9$hn

and finally

- very simple example

and finally

- very simple example
  - no nx
  - no ramdomised stack

and finally

- very simple example
  - no nx
  - no ramdomised stack
- execute shell code from our buffer

and finally

- very simple example
  - no nx
  - no ramdomised stack
- execute shell code from our buffer
- overwrite $eip with location of shell code

and finally

- very simple example
  - no nx
  - no ramdomised stack
- execute shell code from our buffer
- overwrite $eip with location of shell code
- demo