On the effectiveness of NX, SSP, RenewSSP and ASLR against stack buffer overflows

Hector Marco-Gisbert, Ismael Ripoll
Universitàt Politècnica de València (Spain)

2014 IEEE 13th International Symposium on Network Computing and Applications

August 21-23, 2014

Table of contents

Motivation

- Buffer overflows are still a major software threat. [Top 25]
- The NX, SSP, RenewSSP and ASLR protection techniques:
  - Try to defeat/mitigate stack buffer overflows.
  - Used on modern operating systems like Windows, Linux, Android etc,.
- **New attack vectors**, not considered when these techniques were developed, makes necessary to reassess their effectiveness to avoid a false sense of security.
- We reassess the NX, SSP, RenewSSP and ASLR exploiting a stack buffer overflow on: Single process, Inted and Forking servers.

Stack buffer overflow vulnerabilities

- The study has been focused on the stack buffer overflow vulnerabilities, considering multiple attack vectors.

```
void func1(char *src, int lsrc)
{
   char buff[48];
   int i = 0;
   ...
   memcpy(buff, src, lsrc);
   ...
}
```

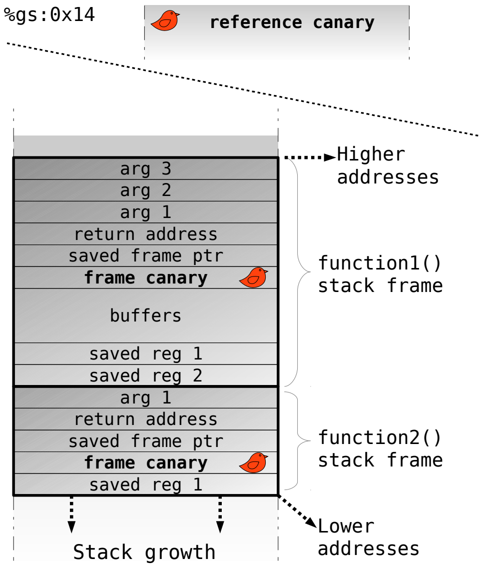
Listing 1: memcpy example.

```
void func2(char *str, int lstr){
   char buff[48];
   int i = 0;
   ...
   for (i = 0; i < lstr; i++) {
     if (str[i] != '\n')
       buff[lbuff++] = str[i];
   ...
}
```

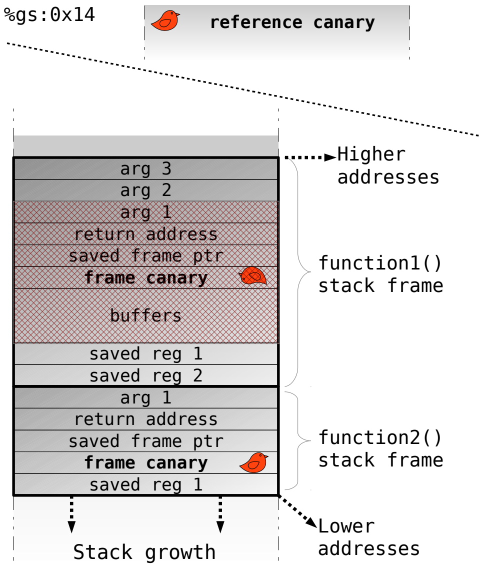
Listing 2: loop example.

- Exploit successfully these vulnerabilities depends on the kind of server.
- It is more reliable to exploit these vulnerabilities on forking servers.

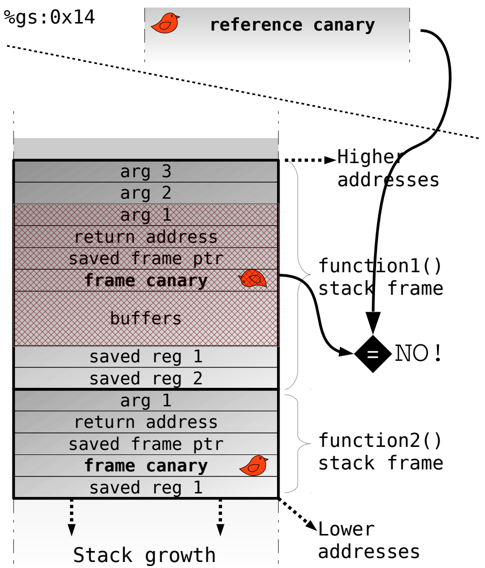
Example 1/3


%gs:0x14
reference canary
Higher addresses
arg 3
arg 2
arg 1
return address
saved frame ptr
frame canary
buffers
saved reg 1
saved reg 2
function1() stack frame
arg 1
return address
saved frame ptr
frame canary
saved reg 1
function2() stack frame
Lower addresses
Stack growth

Example 2/3


%gs:0x14
reference canary
Higher addresses
arg 3
arg 2
arg 1
return address
saved frame ptr
frame canary
buffers
saved reg 1
saved reg 2
function1() stack frame
arg 1
return address
saved frame ptr
frame canary
saved reg 1
function2() stack frame
Lower addresses
Stack growth

Example 3/3


%gs:0x14
reference canary
arg 3
arg 2
arg 1
return address
saved frame ptr
frame canary
buffers
saved reg 1
saved reg 2
arg 1
return address
saved frame ptr
frame canary
saved reg 1
Higher addresses
function1() stack frame
=
NO!
function2() stack frame
Lower addresses
Stack growth

Type of servers

Single server:

- An incorrect attempt attack $\rightarrow$ crash $\rightarrow$ service stopped.
- Little chances to break into the server but easy to do a DoS.
- No real servers use this model.

Type of servers

Single server:

- An incorrect attempt attack → crash → service stopped.
- Little chances to break into the server but easy to do a DoS.
- No real servers use this model.

Inted server:

- An incorrect attempt attack → crash → relaunch the service.
- Every attempt → renew all secrets. (`fork()+exec()`→`attend()`)
- Paranoid servers (SSH suit) or services through the Inted (ftpd).

Type of servers

Single server:

- An incorrect attempt attack → crash → service stopped.
- Little chances to break into the server but easy to do a DoS.
- No real servers use this model.

Inted server:

- An incorrect attempt attack → crash → relaunch the service.
- Every attempt → renew all secrets. (`fork()+exec()→attend()`)
- Paranoid servers (SSH suit) or services through the Inted (ftpd).

Forking server:

- An incorrect attempt attack → crash → use a new child.
- Every attempt → **not** renew all secrets. (`fork() → attend()`).
- Most servers use it. Examples: Apache, lighttpd, etc.

Protection techniques

NX or DEP:

- Executable pages are not writable.
- Prevent the execution of the injected code.

Protection techniques

NX or DEP:

- Executable pages are not writable.
- Prevent the execution of the injected code.

SSP:

- Random value placed on the stack initially to protect the return address.
- Detects stack buffer overflows and aborts the execution.

Protection techniques

NX or DEP:

- Executable pages are not writable.
- Prevent the execution of the injected code.

SSP:

- Random value placed on the stack initially to protect the return address.
- Detects stack buffer overflows and aborts the execution.

ASLR:

- New process are loaded randomly in the main memory.
- Prevents attacks relying on the knowing absolute addresses.

Protection techniques

NX or DEP:

- Executable pages are not writable.
- Prevent the execution of the injected code.

SSP:

- Random value placed on the stack initially to protect the return address.
- Detects stack buffer overflows and aborts the execution.

ASLR:

- New process are loaded randomly in the main memory.
- Prevents attacks relying on the knowing absolute addresses.

RenewSSP:

- A recent modification of the SSP.
- Prevents SSP brute force attacks on forking servers.

Bypassing NX, SSP, RenewSSP and ASLR 1/3

NX/DEP:

- Using attacks that do not require to execute the injected code.
- Modern attacks do not inject code but use ROP, JOP etc.

Bypassing NX, SSP, RenewSSP and ASLR 1/3

NX/DEP:

- Using attacks that do not require to execute the injected code.
- Modern attacks do not inject code but use ROP, JOP etc.

SSP-tat (SSP trial-and-test):

- The canary value is replaced after each trial. (sampling **with** replacement)
- The attacker can try at will but can not discard already tested values.

Bypassing NX, SSP, RenewSSP and ASLR 1/3

NX/DEP:

- Using attacks that do not require to execute the injected code.
- Modern attacks do not inject code but use ROP, JOP etc.

SSP-tat (SSP trial-and-test):

- The canary value is replaced after each trial. (sampling **with** replacement)
- The attacker can try at will but can not discard already tested values.

SSP-bff (SSP brute-force-full):

- The canary value is the same in every trial. (sampling **without** replacement)
- The attacker can build a brute force attack to obtain the canary.

Bypassing NX, SSP, RenewSSP and ASLR 2/3

SSP-bfb (SSP byte-for-byte):

- The canary value is the same in every trial. (sampling **without** replacement)
- The attacker can build a brute force attack but trying all possible values of each byte sequentially.

Bypassing NX, SSP, RenewSSP and ASLR 2/3

SSP-bfb (SSP byte-for-byte):

- The canary value is the same in every trial. (sampling **without** replacement)
- The attacker can build a brute force attack but trying all possible values of each byte sequentially.

RenewSSP-tat (RenewSSP trial-and-test):

- The canary value is replaced after each trial. (sampling **with** replacement)
- Only trial-and-test is possible independently of type of server (single, inted or forking)

Bypassing NX, SSP, RenewSSP and ASLR 2/3

SSP-bfb (SSP byte-for-byte):

- The canary value is the same in every trial. (sampling **without** replacement)
- The attacker can build a brute force attack but trying all possible values of each byte sequentially.

RenewSSP-tat (RenewSSP trial-and-test):

- The canary value is replaced after each trial. (sampling **with** replacement)
- Only trial-and-test is possible independently of type of server (single, inted or forking)

ASLR-bff (ASLR brute force full):

- The memory map is the same in all trials. (sampling **without** replacement)
- The attacker can build a brute force attack trying all possible addresses.

Bypassing NX, SSP, RenewSSP and ASLR 3/3

ASLR-tat (ASLR trial-and-test):

- The memory map is the same in all trials. (sampling **with** replacement)
- The attacker can **not** build a brute force attack trying all possible addresses.

Bypassing NX, SSP, RenewSSP and ASLR 3/3

ASLR-tat (ASLR trial-and-test):

- The memory map is the same in all trials. (sampling **with** replacement)
- The attacker can **not** build a brute force attack trying all possible addresses.

ASLR-one (ASLR one shot):

- Applications under certain circumstances the ASLR can be bypassed using a single attempt.
- For example building a ROP sequence from non-randomised applications (Not PIE compiled)

Summary of symbols

Symbol	Description
C	entropy bits of the canary.
n	number of entropy bytes of the canary ($n = C/8$).
c	number of values that can take the canary ($c = 2^C$).
R	entropy bits of the ASLR for libraries.
r	number of places where the library can be located ($r = 2^R$).
k	number of trials (attempts) done by a attacker to a service.

Table : Summary of symbols.

Example on some 32 bit architectures:

- $n = 3$ canary bytes (one byte is zeroed)
- $C = 24 \rightarrow c = 2^{24} = 16777216$ possible canary values.
- $R = 8 \rightarrow r = 2^8 = 256$ places to load the library.

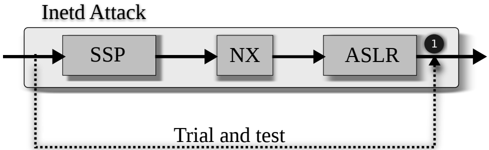
Single process

- The attacker only has a single trial to bypass both the SSP and the ASLR.

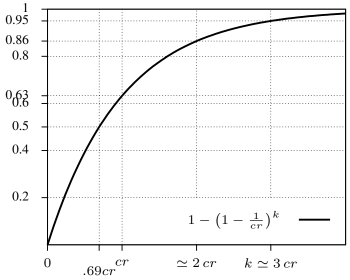
$$Pr(\mathcal{X} = n) = \begin{cases} 1 - \frac{1}{cr} & \text{if } n = 0, \text{"failure"} \\ \frac{1}{cr} & \text{if } n = 1, \text{"success"} \end{cases} \quad (1)$$

- A crash $\rightarrow$ service stopped. (the service is not restarted)
- This type of server has been introduced for completeness.

Inetd server



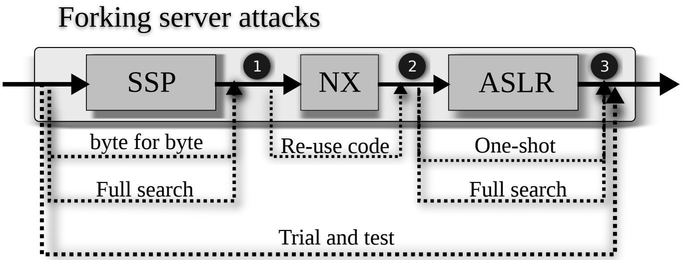
- The attacker can do as many trials as needed but the success is **not** guaranteed.
- Each trial has a probability of success of $\frac{1}{cr}$.
- Approx. 3 times more effort than in forking servers. (95% of success in 3 cr trials).



Geometric

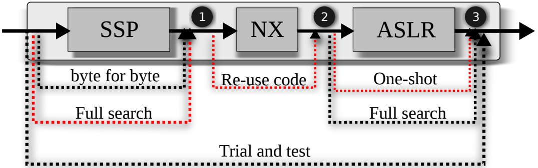
PMF	$\frac{1}{cr}\left(1-\frac{1}{cr}\right)^{k-1}$
CDF	$1-\left(1-\frac{1}{cr}\right)^{k}$
Mean	$\mu = cr$
Variance	$\sigma^2 = \frac{1-cr}{cr}$
Trials for 100%	$= \infty$
95%	$\simeq 3\,cr$
50%	$\simeq 0.693\,cr$

Forking server

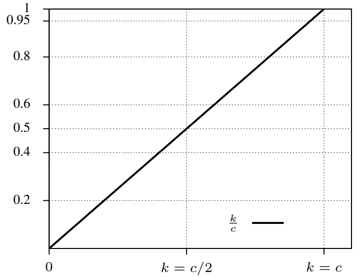


- The attacker can do as many trials as needed:
  - Success **is** guaranteed.
  - Some times is not practical.
- Different attack strategies are possible.
- Realistic attacks bypasses the three protection mechanisms.
- The attacker can attack first the SSP and later the ASLR.

Forking server: SSP-bff + ASLR-one

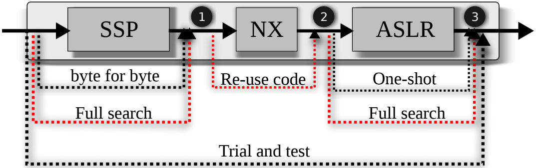


- Full search SSP $\rightarrow$ Uniform distribution.
- One shot ASLR attack $\rightarrow$ zero cost.
- Full search SSP + One shot ASLR = Full search SSP.

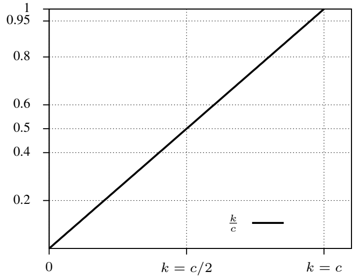


Uniform	
Mean	$\mu = c/2$
Variance	$\sigma^2 = (c-1)/12$
PMF	$1/c$
CDF	k/c
Trials for 100%	$= c$
95%	$= 0.95c$
50%	$= c/2$

Forking server: SSP-bff + ASLR-bff

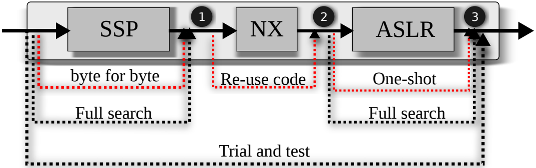


- Full search SSP $\rightarrow$ Uniform distribution.
- Full search ASLR $\rightarrow$ Uniform distribution.
- Since $c/r > 256$ then:
  SSP-full + ASLR-full $\approx$ Uniform. $(k = c + r)$

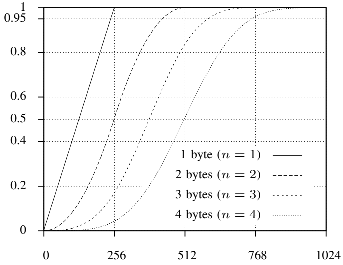


Uniform	
Mean	$\mu = c/2$
Variance	$\sigma^2 = (c-1)/12$
PMF	$1/c$
CDF	k/c
Trials for 100%	$= c$
95%	$= 0.95c$
50%	$= c/2$

Forking server: SSP-bfb + ASLR-one



- Each SSP brute-forced byte → Uniform distribution.
- One shot ASLR attack → zero cost.
- The sum of distributions > 3 can be approx. to a Normal distribution.

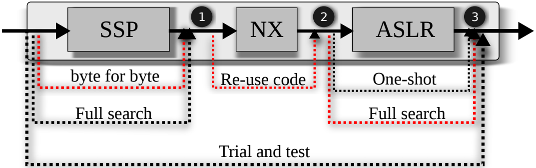


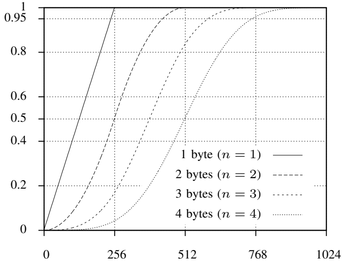
Sum of n uniforms
$\simeq$ Normal when $n > 3$

Mean	$\mu = \frac{256n}{2} = \frac{256\log_2(c)}{2}$
Variance	$\sigma^2 = \frac{(256-1)n}{12}$
PMF	$\simeq \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-(x-\mu)^2/2\sigma^2\right)}$
CDF	$\simeq \frac{1}{2}\left(1 - erf\left(\frac{k-\mu}{\sqrt{2\sigma^2}}\right)\right)$

Trials for 100%	$= 2\mu$
95%	$= \mu + 1.645\sigma^2$
50%	$= \mu$

Forking server: SSP-bfb + ASLR-bff





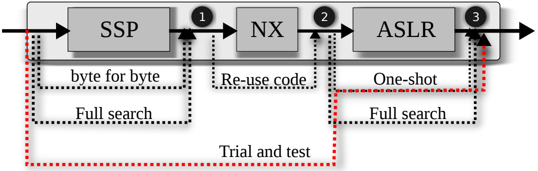
Sum of n uniforms
$\simeq$ Normal when $n > 3$

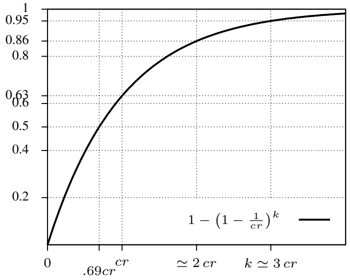
Mean	$\mu = \frac{256n}{2} = \frac{256\log_2(c)}{2}$
Variance	$\sigma^2 = \frac{(256-1)n}{12}$
PMF	$\simeq \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-(x-\mu)^2/2\sigma^2\right)}$
CDF	$\simeq \frac{1}{2}\left(1 - erf\left(\frac{k-\mu}{\sqrt{2\sigma^2}}\right)\right)$

Trials for 100%	$= 2\mu$
95%	$= \mu + 1.645\sigma^2$
50%	$= \mu$

- Each SSP brute-forced byte $\rightarrow$ Uniform distribution.
- Full search ASLR $\rightarrow$ Uniform distribution.
- The sum of distributions > 3 can be approx. to a Normal distribution.
- Example, in Ubuntu 13.10 (x86): The canary has 3 bytes (2^{3x8}), and the ASLR 2^8 which can be seen as a canary value of 4 bytes $\approx$ Normal distribution.

Forking server: RenewSSP-tat + ASLR-one





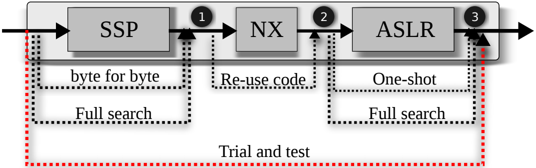
Geometric

PMF	$\frac{1}{cr}\left(1-\frac{1}{cr}\right)^{k-1}$
CDF	$1-\left(1-\frac{1}{cr}\right)^{k}$
Mean	$\mu = cr$
Variance	$\sigma^2 = \frac{1-cr}{cr}$

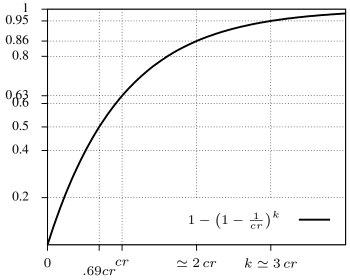
Trials for 100%	$= \infty$
95%	$\simeq 3\,cr$
50%	$\simeq 0.693\,cr$

- Each child has a different canary value $\rightarrow$ **prevents** brute force attacks.
- ASLR one shot $\rightarrow r = 1$
- Success **not guarantee**.
- Each trial has a probability of success of $\frac{1}{c}$.

Forking server: RenewSSP-tat + ASLR-tat



- Each child has a different canary value $\rightarrow$ **prevents** brute force attacks.
- Success **not guarantee**.
- Each trial has a probability of success of $\frac{1}{cr}$.
- Similar to Inted protection but on forking servers.



Geometric

PMF	$\frac{1}{cr}\left(1-\frac{1}{cr}\right)^{k-1}$
CDF	$1-\left(1-\frac{1}{cr}\right)^{k}$
Mean	$\mu = cr$
Variance	$\sigma^2 = \frac{1-cr}{cr}$

Trials for 100%	$=\infty$
95%	$\simeq 3\,cr$
50%	$\simeq 0.693\,cr$

Results

Putting all together ....

	Attack/Bypass	100%	Mean
32bits syst.	SSP-bff + ASLR-bff	4 Hours	2 Hours
	SSP-bff + ASLR-one	4 Hours	2 Hours
	SSP-bfb + ASLR-bff	1 sec	< 1 sec
	SSP-bfb + ASLR-one	< 1 sec	< 1 sec
	RenewSSP-tat + ASLR-one	∞	3 Hours
	RenewSSP-tat + ASLR-tat	∞	34 Days
64bits syst.	SSP-bff + ASLR-bff	2.32 Myr	1.16 Myr
	SSP-bff + ASLR-one	2.32 Myr	1.16 Myr
	SSP-bfb + ASLR-bff	74 Hours	37 Hours
	SSP-bfb + ASLR-one	1 sec	< 1 sec
	RenewSSP-tat + ASLR-one	∞	1605.79 Kyr
	RenewSSP-tat+ASLR-tat	∞	431.05 Tyr

Table : Time cost for attacks in forking servers at 1000 trials/sec.

Conclusions

- NX/DEP obsoleted by new attacks: ret*, ROP, JOP etc,.
- Forking servers reduce the effectiveness of the protection techniques:
  - Allow attack first the SSP and later the ASLR.
  - Allow build brute force attacks.
- SSP is reasonably effective, but fails on forking servers, specially against byte-for-byte attacks.
- The effectiveness of SSP is much better than that of the ASLR (but the ASLR covers more types of attacks).
- RenewSSP removes the dangerous byte-for-byte attack.
- SSP and ASLR are useless on Android.
- The ASLR in Windows is useless against local attacks.

Thank you for your attention !