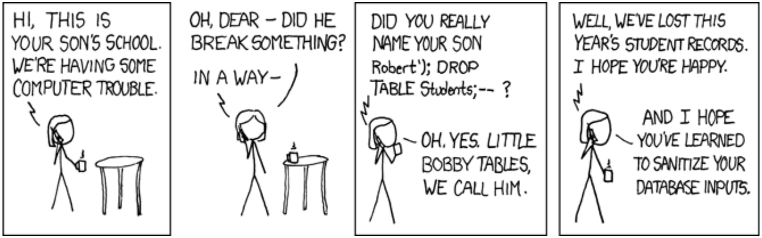
Hardened Stateless Session Cookies


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

`http://xkcd.com/327/`

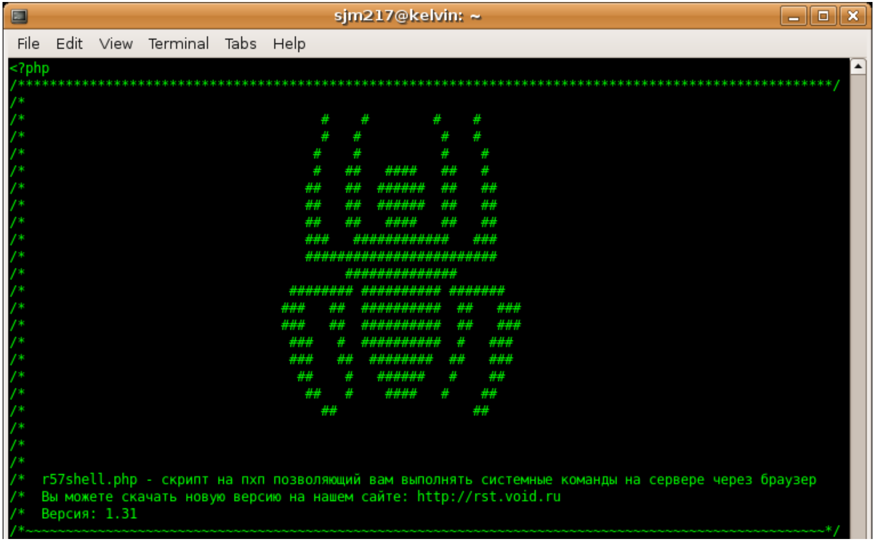
Steven J. Murdoch

`www.cl.cam.ac.uk/users/sjm217`

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

www.torproject.org

Why I am interested in web security



```
<?php
/**********************************************************************************************************************/
/*
/*                                          #    #        #    #
/*                                          #   #          #   #
/*                                         #    #          #    #
/*                                         #   ##   ####   ##   #
/*                                        ##   ##  ######  ##   ##
/*                                        ##   ##  ######  ##   ##
/*                                        ##   ##   ####   ##   ##
/*                                        ###   ############   ###
/*                                        ########################
/*                                             ##############
/*                                      ######## ########## #######
/*                                     ###   ##  ##########  ##   ###
/*                                     ###   ##  ##########  ##   ###
/*                                      ###   #  ##########  #   ###
/*                                      ###   ##  ########  ##   ###
/*                                       ##    #   ######   #    ##
/*                                        ##    #    ####   #    ##
/*                                          ##                 ##
/*
/*
/*
/*   r57shell.php - скрипт на пхп позволяющий вам выполнять системные команды на сервере через браузер
/*   Вы можете скачать новую версию на нашем сайте: http://rst.void.ru
/*   Версия: 1.31
/*--------------------------------------------------------------------------------------------------------------------*/
```

Found on `www.lightbluetouchpaper.org`

How users authenticate to a website

Login:

Client $\rightarrow$ Server	$<$username, password$>$
Server	Checks username and password against database
Server $\rightarrow$ Client	cookie

Page request:

Client $\rightarrow$ Server	$<$request, cookie$>$
Server	Checks cookie

Cookie allows a site to know which user makes a page request. The site can then give access to restricted information, track who makes changes etc. . .

What goes into the cookie

		Database state	
	Cookie	Per-user	Per-session
PHP	Session ID	$f(pw)$	Session ID $\rightarrow$ User ID
Wordpress	H(H(pw))	H(pw)	—
Fu et al.	User ID, MAC	$f(pw)$	—

Per-session state should be avoided as it increases storage requirements, load balancing complication and DoS vulnerability

Problem:
With **read-access** to the database, an attacker can generate a fake cookie

Solution:
Store something in the cookie which can be verified but not spoofed by the server

SQL injection

This SQL statement (based on an example in Wordpress) is vulnerable to SQL injection because `$user_login` is not properly sanitized:

```
SELECT * FROM wp_users WHERE user_login =
'$user_login'
```

An attacker can exploit this vulnerability by setting `$user_login` to be:

```
' UNION ALL SELECT 1,2,user_pass,4,5,6,7,8,9,10
FROM wp_users WHERE ID=1/*
```

An attacker can inject arbitrary SQL after the vulnerable variable, and can terminate the statement. However, the PHP MySQL API only allows **one statement per request**

What to do if your website is vulnerable to SQL injection

Give up?

Once you've got read-only access to a database, how much more vulnerable do you want? — "Computer Guru"

As has already been noted, if an attacker already has read access to your database, then you've probably lost the battle, regardless of anything else. — "dougal"

What to do if your website is vulnerable to SQL injection

Give up?

Once you've got read-only access to a database, how much more vulnerable do you want? — "Computer Guru"

As has already been noted, if an attacker already has read access to your database, then you've probably lost the battle, regardless of anything else. — "dougal"

Defence in depth?

We've lost the battle, but can we win the war? The attacker can't change the statement type so if the exploitable query is `SELECT`, the **attacker can read, but not write**

This model of attacker can still break all the previous authentication schemes, so can we do better?

A new cookie proposal (simplified)

Server stores: $<$User ID, s =salt, v =H(A(s, password))$>$.
Where: H() is a hash function; A() is password salting function

Login:

Client $\rightarrow$ Server	$<$username, password$>$
Server	Checks if H(A(s, password))$=$ v
Server $\rightarrow$ Client	Cookie c =A(s, password)

Page request:

Client $\rightarrow$ Server	$<$request, cookie $c$$>$
Server	Checks if H(c)$=$ v

Potential attacks:
Read cookie: can't go from c on client to password (without s)
Read database: can't go from $< v, s >$, to c (unless password is weak)

Can we do any better? Are there any other attacks?