A CRITICAL ANALYSIS OF DROPBOX SOFTWARE SECURITY

Florian LEDOUX
florian.ledoux(at)eads.net
@Myst3rie

Nicolas RUFF
nicolas.ruff(at)eads.net
@newsoft

EADS Innovation Works
SE/IT department
Suresnes, FRANCE
Dropbox: a leader in Cloud backup

- Over 50 million users
- Estimated company value: over $1 billion

Client software available for

- Windows, OS X, Linux, Android, iOS and web browser

Lot of competitors

- Google Drive, SkyDrive, iCloud, box.com ...
Dropbox security record (partial)

- March 2011: Dropbox client for Smartphones do not make use of SSL/TLS encryption
- April 2011: Derek Newton realized that login/password is useless (if you happen to know host_id secret)
- June 2011: a software upgrade issue provided password-free access to all user accounts for one day
- USENIX 2011: "Dark Clouds on the Horizon"
- August 2012: a stolen password from Dropbox employee lead to massive spam
Why studying Dropbox?

- Dropbox is a leader
- No previous work on the effective implementation
- "LAN Sync" protocol routinely observed during penetration testing assignments
- We are happy Dropbox users too 😊
Further analysis holds true for client versions 1.1.x to 1.5.x

Windows, Linux and OS X clients are mostly written in Python
  - "How Dropbox Did It and How Python Helped" (PyCon 2011)

Windows client
  - Generated using PY2EXE
  - A ZIP with all PYC files to be found within PE resources
  - Python 2.5 interpreter has been slightly customized
Source quest
### Standard PYC (redux)

- PYC is Python bytecode
- PYO is Python optimized bytecode

<table>
<thead>
<tr>
<th>Bytecode version</th>
<th>Timestamp</th>
<th>Marshalled bytecode</th>
</tr>
</thead>
<tbody>
<tr>
<td>b3 f2 0d 0a</td>
<td>0d f1 5c 50</td>
<td>63 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00 06 00 00</td>
<td>00 40 00 00</td>
<td>00 73 16 01 00 00 78 43</td>
</tr>
<tr>
<td>00 65 00 00</td>
<td>64 00 00 83</td>
<td>01 00 44 5d 30 00 5a 01</td>
</tr>
</tbody>
</table>

### Dropbox PYC

| b3 f2 0d 0a      | 0d f1 5c 50 | 63 70 f9 79 04 8e 20 00 |
| 00 90 e0 95 65 67 29 9d | 83 7b 7d f3 16 1e 2a 68 |
Diffing **PYTHON25.DLL** with original

- 53 modified functions (out of ~4500)
- Opcode have been swapped in `PyEval_EvalFrame()`
- Decryption function added in `ReadObjectFromString()`

Which encryption algorithm is used?

- **0x9e3779b9** constant is linked to TEA symmetric encryption family
  
  Here: **XXTEA**

- `MT_getnext()` / `MT_decrypt()` functions are involved
XXTEA implementation

```c
void btea(char *data, uint32 len, uint32 const key[4])
```

- **Key seed**
- **Block len**

---

**ReadObjectFromString()**

- Read 1st byte (e.g. **0x63** = code)
- 1st DWORD (e.g. **0x0479F970**) used for key generation
- 2nd DWORD (e.g. **0x208e**) gives block size

---

**Not as easy as it may sounds**

- Spurious NULL bytes all over the place
Bytecode decompilation
- Pyretic / unpyc
  - Targets **Python 2.5** (Fails in real life)
- Uncompyle2
  - Targets **Python 2.7** only (Works in real life)

Our solution
- Uncompyle2 fork
- Bytecode translator 2.5 & 2.6 ➔ 2.7
- Single decompilation engine
- Kudos to Eloi Vanderbeken

https://github.com/Mysterie/uncompyle2
CODE INJECTION (BONUS)

PYTHON25.DLL is not easy to reach
- Anonymously mapped in memory
- Not easy to locate import / export tables
- Some functions like `PyRun_File()` are nop'ed

Yet ...
- `PyRunString()` is not patched
- Arbitrary Python statements can be run in Dropbox context 😊
Debugging is hard

DBDEV environment variable to the rescue

Dropbox <= 1.1

def is_valid_time_limited_cookie(cookie):
    t_when = int(cookie[:8], 16) ^ 1686035233
    if abs(time.time() - t_when) < 172800:
        if md5.new(cookie[:8] + 'traceme').hexdigest()[:6] == cookie[8:]:
            return True
DEBUG MODE

Dropbox ≥ 1.2

```
IS_DEV_MAGIC = DBDEV and
hashlib.md5(DBDEV).hexdigest().startswith('c3da6009e4')
```
DEBUG MODE

DBTRACE can help, too

10.224 | MainThread: Dropbox-win-1.1.45 (2796) starting

10.865 | MainThread: u'host_id' = u'ab75c...

13.509 | MainThread: Opened Dropbox key

32.356 | RTRACE: Sending trace 1327936014 (C:\...\Dropbox\l\4f26b5fc)

33.058 | STATUS: Creating named pipe

59.318 | UPLOAD_HASH: Next needed hash: AUCwQ6iYIfVxGs1f6HjkWZgqc_bmWZiTc6HU8HRykJzU
... and many others

- `DBMEMPROF, DBCPUPROFILE, DBPROFILE`
- `FAKE_BLOCK`
- `DROPBOX_HOST`

Who's in charge here?

- `host = 'tarak.corp.dropbox.com'`
- Not exposed on the Internet 😊
GIMME RESULTS …

… not excuses!
CONFIGURATION DATABASE

SQLite 3 database: config.dbx

- Dropbox < 1.2: easy to dump
- Dropbox ≥ 1.2: "encrypted" SQLite

Encryption

Not: http://sqlcipher.net/
But: http://www.hwaci.com/sw/sqlite/see.html

Activation password == license key == default value 😊

Namely: 7bb07b8d471d642e
Encryption key is machine-protected

Windows
- Seed stored in HKCU\Software\Dropbox\ks\Client
- DPAPI encryption

Linux
- Seed stored in ~/.dropbox/hostkeys
- Custom "obfuscator" (reversible encryption)

Mac OS X
- Seed stored in ~/.dropbox/hostkeys
- Custom "obfuscator" based on IOPlatformSerialNumber, DAVolumeUUID and more
- Kudos to the Mac OS X developer for full API re-implementation!
Effective encryption key is PBKDF2 (seed)

Please use this information for forensics purpose only 😊

USER_HMAC_KEY = 'xd1\x14\xa5\xe_t\xbdw.7\xe6J\xee\x9b'
APP_KEY = 'rc\x8\t.\xb\xfcE(\x83\xf9_5[\x8e'
APP_IV = 'xd8\x9bC\x1f\xb6\x1d\xde\x1a\xfd\xa4\xb7\xf9\xf4\xb8\r\x05'
APP_ITER = 1066
USER_KEYLEN = 16
DB_KEYLEN = 16
Network protocols
Network traffic

- fully transported over HTTPS
- OpenSSL + nCrypt wrapper
- Proper certificate checking
  - Hardcoded CA list

```python
root_certs = '# Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc, ...

-----BEGIN CERTIFICATE-----
MIIDEzCCAnygAwIBAgIAMBIGAwIAYDVR0jBGYqEwU şekEk
(...)
L7tdEy8W9ViH0Pd

-----END CERTIFICATE-----
```

Issues

OpenSSL … 0.9.8e?
- as of DropBox 1.4.17
- Hello CVE-2011-4109, CVE-2012-2110, and others

nCrypt … completely buggy and unsupported software?

http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=614051

No patch since 2007
File synchronisation: RSYNC protocol

File storage: Amazon Cloud S3

Implementation details

- Blocks of 4 MB in size
- SHA-256 of each block
- Encryption is provided by SSL/TLS only
DROPBOX PROTOCOL

Servers of interest

**Blockserver**: manages 4MB blocks

**Authserver**: user authentication, software setup

**Metaserver**: handles information requests about files and directories

**Metaexcserver / blockexcserver**: handle exceptions

**Statserver / notifyserver**: statistics

```python
set_server(ret, 'blockserver', secure=True, timeout=60, **non_exc_kwargs)
set_server(ret, 'metaserver', secure=True, timeout=90, **non_exc_kwargs)
set_server(ret, 'metaexcserver', secure=True, timeout=90, **exc_kwargs)
set_server(ret, 'blockexcserver', secure=True, timeout=90, **exc_kwargs)
set_server(ret, 'statserver', secure=True, timeout=90, **exc_kwargs)
set_server(ret, 'notifyserver', secure=False, timeout=90, **non_exc_kwargs)
```
DROPBOX PROTOCOL

HOST_ID
- Unique and forever user identifier
- 128-bit length
- Server-side generated on 1\textsuperscript{st} installation
- Not affected by password change
- Stored in local configuration database

HOST_INT
- Unique identifier per device

NS_MAP
- User namespace identifier
- Killed "dropship" hack

Before: \texttt{get\_block( hash\_for\_block )}
After: \texttt{get\_block( hash\_for\_block ; ns\_map + host\_id)
LAN sync protocol
LAN SYNC PROTOCOL

Local sync between two Dropbox clients
- Discovery: UDP/17500 broadcasts
- Data exchange: TCP/17500

Data exchange protocol
- Each Dropbox instance can act as a Client or a Server
- Client SSL/TLS authentication
  - Key pair in configuration database
Attacking a client in server mode

Requires a server-known key pair 😞
LAN SYNC PROTOCOL

Attacking the client mode
  - Server certificate is not checked

LAN Sync protocol (redux)
  - HELLO / HOWDY
  - PING / PONG
  - HAS / HASREPLY / HASFAIL (+ hash)
  - GET / GETREPLY / GETFAIL (+ hash & file content)
Demo !
QUESTIONS