A critical analysis of Dropbox software security

Nicolas RUFF

Florian LEDOUX

EADS Innovation Works

Agenda

- Dropbox overview
- Source quest
- Configuration database
- Network protocols
- LAN sync protocol
- Conclusion

Dropbox overview

- Dropbox: a leader in Cloud backup
  - Over 50 million users
  - Estimated company value: over $1 billion
  - (Year: 2011 / Source: Wikipedia)
- Client software available for
  - Windows, OS X, Linux, Android, iOS and web browser
- Lot of competitors
  - Google Drive, SkyDrive, iCloud, box.com ...

Dropbox overview

- 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
    - Software upgrade issue provided password-free access to all user accounts for one day
  - USENIX 2011
    - "Dark Clouds on the Horizon“
  - August 2012
    - Stolen password from Dropbox employee lead to massive spam

Dropbox overview

- 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

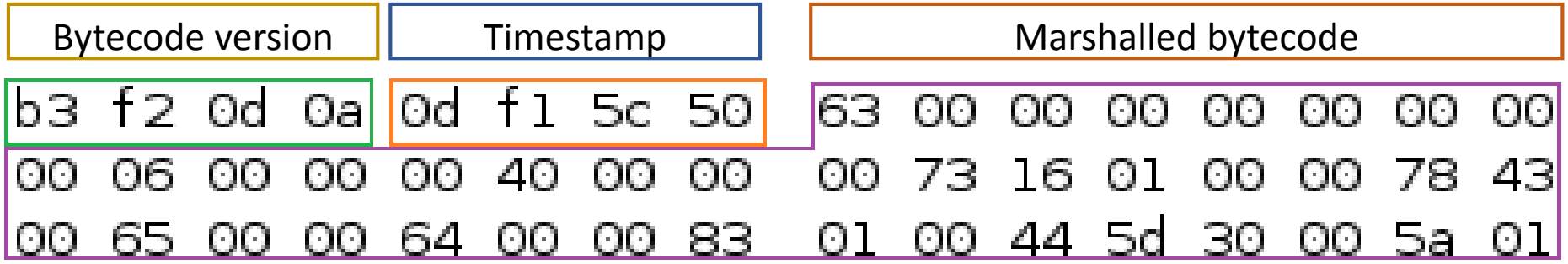
Dropbox overview

- 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
I
QUEST FOR THE CROWN

Source quest

- Standard PYC (redux)
  - PYC is Python bytecode
  - PYO is Python optimized bytecode



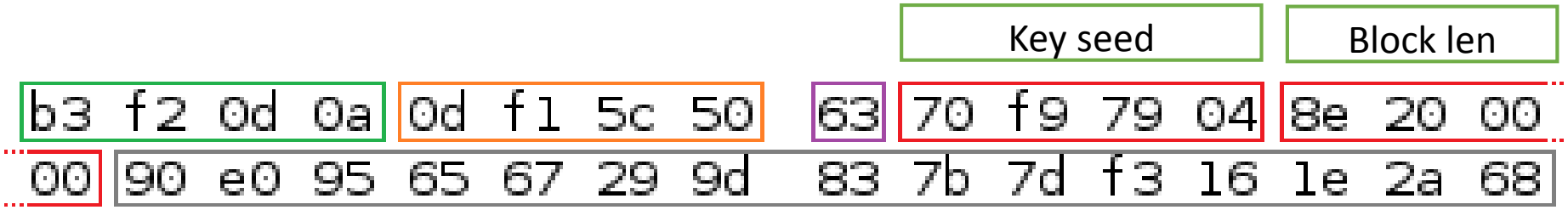
- Dropbox PYC

Source quest

- Diffing **PYTHON25.DLL** with original
  - 53 modified functions (out of ~4500)
  - Opcodes 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
    - Namely: **XXTEA**
  - **MT_getnext()** / **MT_decrypt()** functions are involved in decryption

Source quest

- XXTEA implementation
  - void btea(char *data, uint32 len, uint32 const key[4])



- 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

Source quest

- 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

Python statements injection

- PYTHON25.DLL is not easy to reach
  - Anonymously mapped in memory
    - WinDbg Synthetic Modules FTW ☺
  - 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

Debug mode(s)

- 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(s)

- Dropbox ≥ 1.2

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

Debug mode(s)

- **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:
AUCwQ6iYIfVxGs1f6HjkWZgqcbmWZiTCs6HU8HRykzU
```

Debug mode(s)

- … 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
memegenerator.net

Configuration database

- SQLite 3 database: **config.dbx**
  - Dropbox < 1.2: easy to dump
  - Dropbox ≥ 1.2: "encrypted" SQLite

- Encryption
  - Not:
    - http://sqlcipher.net/
  - But rather:
    - http://www.hwaci.com/sw/sqlite/see.html
  - Activation password == license key == default value ☺
    - Namely: **7bb07b8d471d642e**

Configuration database

- 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!

Configuration database

- Effective encryption key is `PBKDF2(seed)`
- Please use this information for forensics purpose only ☺

- https://github.com/newsoft

```
USER_HMAC_KEY = '\xd1\x14\xa5R\x12e_t\xbdw.7\xe6J\xee\x9b'
APP_KEY = '\rc\x8c\t.\x8b\x82\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
```

Autoupdate

- Signed updates

```
MAGIC = 14401537
VERSION = 3
DIGESTTYPE = 'SHA256'

PUBLICKEY = '\n-----BEGIN RSA PUBLIC KEY-----
\nMIIBCAKCAQEAs24msupO4460ViJDTX4qbdqcosjkDKyjW8ZseZ8fm54hXUPwZz7V\nLinFS3M6mjjKnAH81dN
b3u3KnKadQ/8eHQXIjvmVPGSGHhCc7PRon30wQZYH/azQ\na+ld27xKdzxiB1zK9f2/uzV5sgs7QUhJdcqIpMXM
WAyH7MbsU8g+YEXu/Mz0yZv6\nrAHkupNWoddd7+AjEAeKvlKjOM805+pwedjN3FKnAWSWIIzJJZk76loXoboub
/RB\nPmN83HNJdmFmDda0AY8qWtgS+DX/xEaipbCvda33ZHt/pIhfwl0Wq8RPN7cdS6DE\nW4qbB0qxBdOF/Wt5
JJmGEIXiKHH/udTuIwIBBQ==\n-----END RSA PUBLIC KEY-----\n'
```

URL generation

- Public link
  - Generated from "uid"
    - Ex. https://dl.dropbox.com/u/12345678/toto.pdf

```
def generate_public_link(self, path):
        path = unicode(path)
        public_root = '%s:/public/' % (self.root_ns,)
        if path.lower().startswith(public_root):
            return self.construct_full_url('/u/%s/%s' % (self.uid,
urllib.quote(path[len(public_root):].encode('utf-8'))),
host_prefix=self._public_host_prefix(), with_locale=False)
        else:
            raise Exception('no public link')
```

URL generation

- Gallery link
  - Hash generated from user_key + path + uid + depth
    - Ex. https://www.dropbox.com/gallery/12345678/1/toto?h=123456

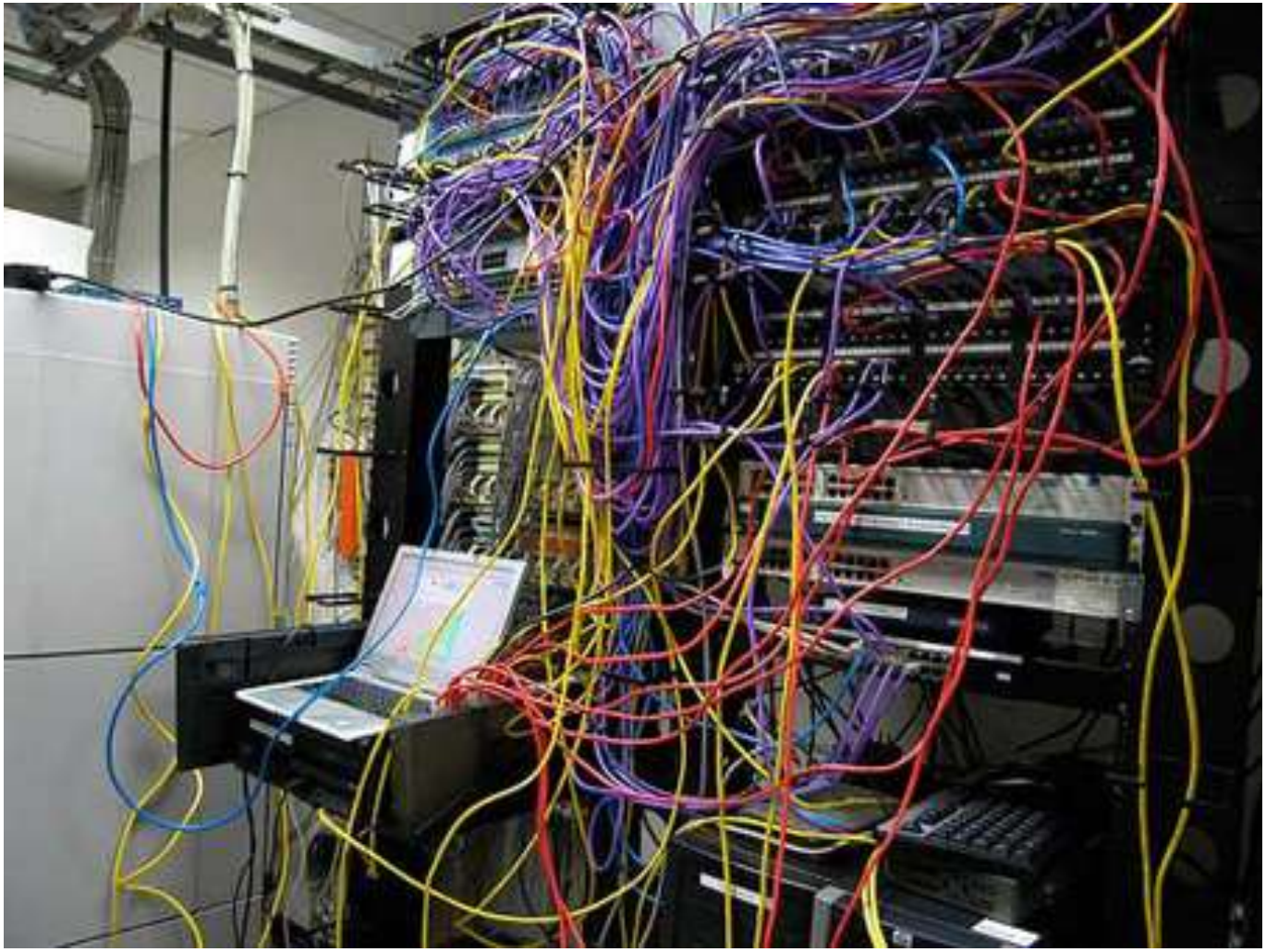
```
def generate_gallery_link(self, path, user_key):
  photos_root = '%s:/photos/' % (self.root_ns,)
  if path.lower().startswith(photos_root):
      ns_id, rel_path = path.split(':/', 1)
      tk_path_components = rel_path.split('/')[1:]
      depth = len(tk_path_components)
      hash = md5.new(('%s%s%s%s' % (user_key,
       '/'.join(tk_path_components),
       self.uid,
       depth)).encode('utf-8')).hexdigest()[3:9]
      urlpath = '/gallery/%s/%s/%s' % (self.uid, depth, urllib.quote(path[len(photos_root):].encode('utf-8')))
      return self.construct_full_url(urlpath, ['h=%s' % (hash,)], with_locale=False)
  else:
      raise Exception("Can't find photos root; (make sure it's converted to our root namespace first)")
```

URL generation

- Web link
  - Generated from host_id + hardcoded secret + timestamp

```
def launch_dropbox_url(self, url):
  if self._host_id is not None and self.host_int is not None and self._server_time is not None:
      delta = (get_monotonic_time() - self._monotonic_time) / get_monotonic_frequency()
      t = self._server_time + int(delta)
      TRACE('Delta monotonic time is %r, computer server time is %r, %r', delta, t, time.ctime(t))
      query_pieces = ['i=%d' % (self.host_int,),
       't=%d' % (t,),
       'v=%s' % (hashlib.sha1('%ssKeevie4jeeVie9bEen5baRFin9%d' % (self._host_id, t)).hexdigest(),),
       'url=%s' % (urllib.quote(url, safe=''),)]
      full_url = self.construct_full_url('/tray_login', query_pieces)
  else:
      full_url = self.construct_full_url(path=urllib.quote(url, safe=''))
  self.launch_full_url(full_url)
```

Network protocols



Source: http://www.techeblog.com/index.php/tech-gadget/spaghetti-cabling-cable-management-gone-wrong

Network protocols

- Network traffic
  - Fully transported over HTTPS
  - OpenSSL + nCrypt wrapper
  - Proper certificate checking
    - Hardcoded CA list

```
root_certs = '#         Subject: C=ZA, ST=Western Cape, L=Cape
Town, O=Thawte Consulting cc, (…)
-----BEGIN CERTIFICATE-----\n
MIIDEzCCAnygAwIBAgIBATA
(…)
L7tdEy8W9ViH0Pd\n
-----END CERTIFICATE-----\n\n'
```

Network protocols

- 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

Network protocols

- 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

Network protocols

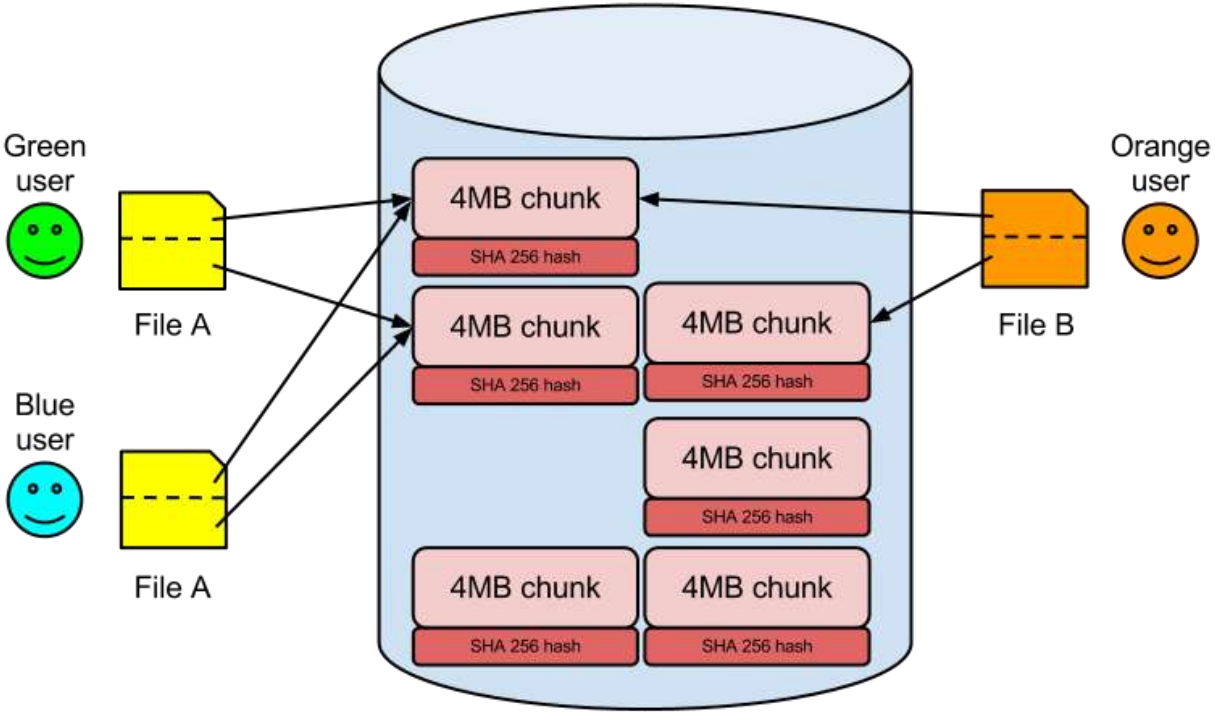
- Servers of interest
  - **Blockserver**: manages 4MB blocks
  - **Authserver**: user authentication, software setup
  - **Metaserver**: handles information requests about files and directories
  - **Metaexcserver** / **blockexcserver**: handle exceptions
  - **Statserver** / **notifyserver**: statistics (HTTP)

```
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)
```

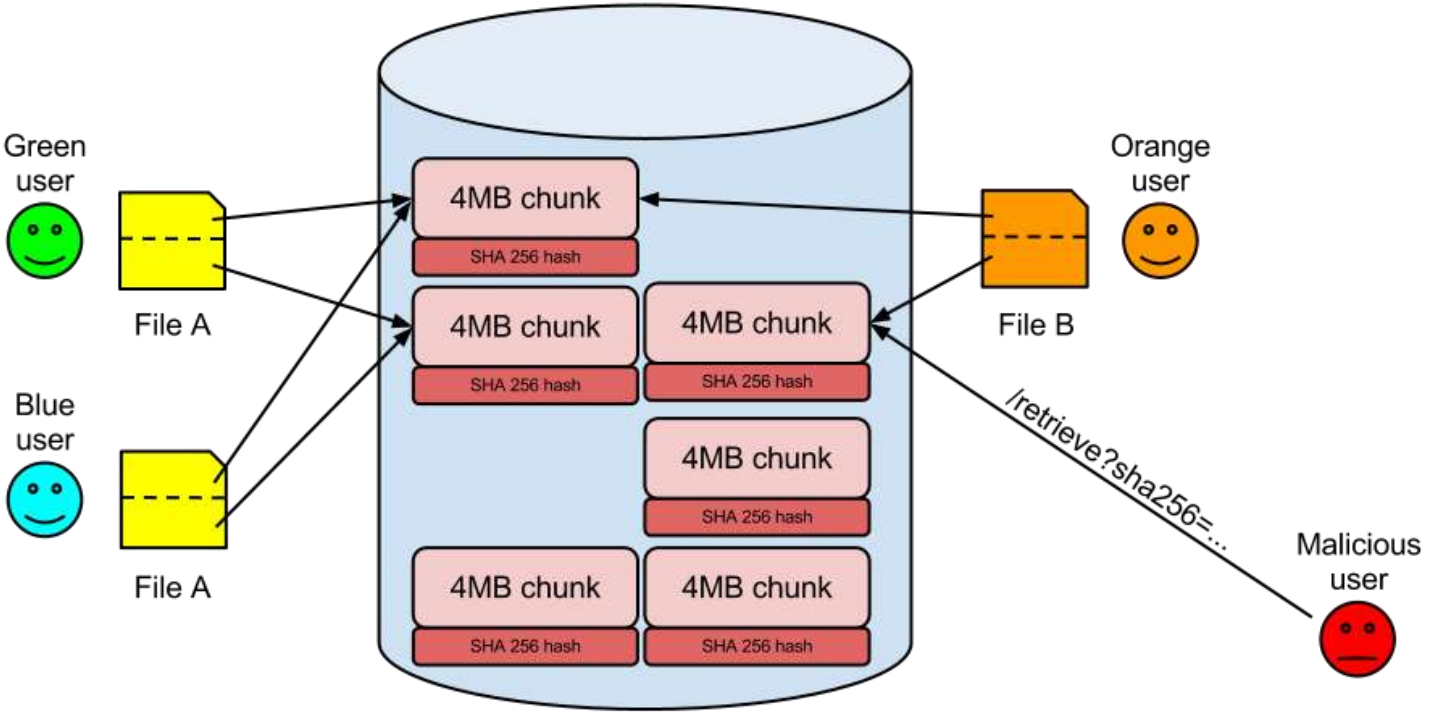
Network protocols

- HOST_ID
  - Unique and forever user identifier
  - 128-bit length
  - Server-side generated on 1st installation
  - Not affected by password change
  - Stored in local configuration database
- HOST_INT
  - Unique identifier per device
- NS_MAP
  - User namespace identifier

Network protocols

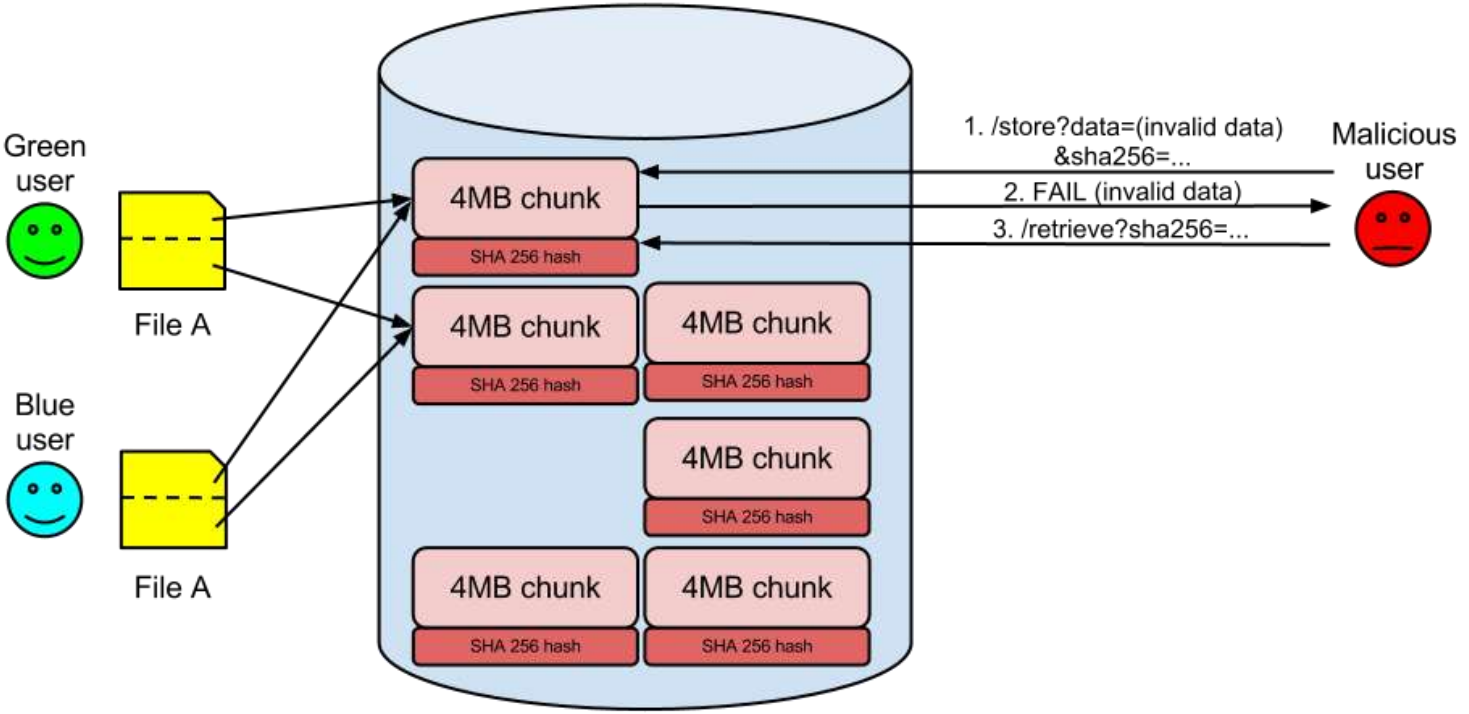


Network protocols (Dropship)



https://github.com/driverdan/dropship

Network protocols (Usenix 2011)



http://www.sba-research.org/2011/06/22/usenix-security-11-dark-clouds-on-the-horizon/

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 stored in configuration database

LAN sync protocol

- 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)

Conclusion

- There used to be critical issues in Dropbox design
  - Most of them are now fixed
  - Software is even pretty robust

- Our contribution to the "state of the art"
  - Full Python 2.5/2.6/2.7 decompilation
    - 96% of Python standard library decompiled in original form
  - DBX decryption
    - Useful for forensics
  - LAN sync attacks
    - DoS against a remote client
    - Remote monitoring of SHA-256 hashes

Questions?