Dark and Bright Sides of the iCloud (In)Security

Andrey Belenko
Sr. Security Engineer
viaForensics

Dmitry Sklyarov
Lead Analyst
Positive Technologies

Disclaimer

This work has been done in early 2012 while we have been still working for (and receiving paychecks from) Elcomsoft.

Permission has been obtained prior to preparing this presentation and we are thankful to Elcomsoft for providing such permission.

Elcomsoft also offers a commercial tool (based on this research) that can download iCloud backups.

Motivation

Mobile device forensics

Motivation

Devices are becoming increasingly interconnected but forensic tools are not ready for this (yet)

Motivation

iCloud is the natural target as it is quite big and popular

Motivation

So it is about iOS forensics in the first place

Forensics 101

Acquisition ➜ Analysis ➜ Reporting

GOALS:

1. Assuming physical access to the device extract as much information as practical

2. Leave as little traces/artifacts as practical

iOS: Why Even Bother?

- More than 5 years on the market
- 360+ million iOS devices sold worldwide
- 7 iPhones, 5 iPods, 5 iPads
- "Smart devices" – they do carry a lot of sensitive data
- Corporate deployments are increasing

There was, is, and will be a real need for iOS forensics

iOS Forensics 101

- Passcode
  - Prevents unauthorized access to the device
  - Bypassing passcode is usually enough
- Keychain
  - System-wide storage for sensitive data
  - Encrypted
- Disk encryption

iOS Forensics 101

- Logical: iPhone Backup
  - "Ask" device to produce backup
  - Device must be unlocked (by passcode or iTunes)
  - Device may produce encrypted backup
  - Limited amount of information

iOS Forensics 101

- Physical: filesystem acquisition
  - Boot-time exploit to run unsigned code
  - Device lock state isn't relevant, can bruteforce passcode
  - Can get all information from the device

- Physical+: flash memory acquisition
  - Same requirements as for physical
  - Also allows recovery of deleted files!

iOS Data Protection

Every iOS device contains secure AES engine with two embedded keys:

- GID – shared by all devices of same "family"
- UID – unique per device
- Newer devices have additional UID+ key

There are no (publicly) known ways to extract GID or UID key from the device

iOS Data Protection

- Content grouped by accessibility requirements:
  - Available only when device is unlocked
  - Available after first device unlock (and until power off)
  - Always available
- Each protection class has a master key
- Master keys are protected by device key and passcode
- Protected master keys form system keybag
  - New keys created during device restore

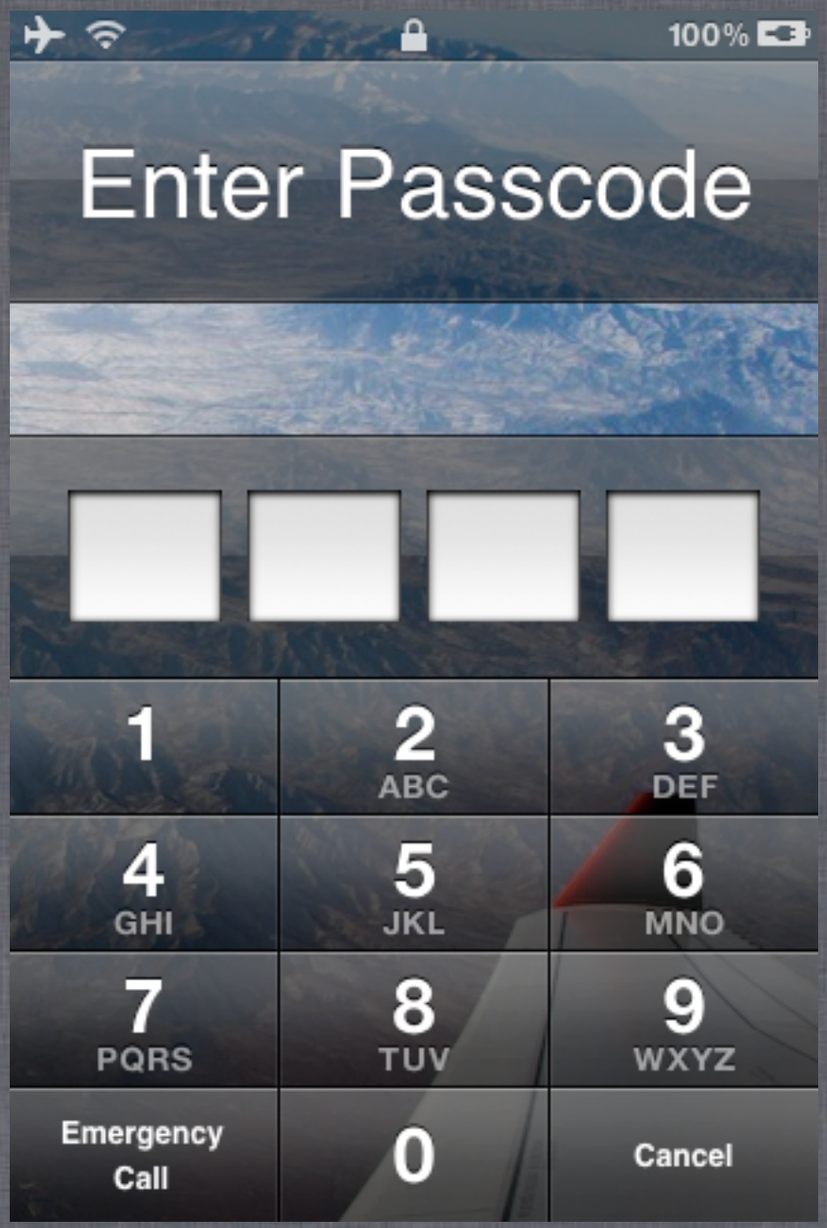
iOS 4+ Passcode

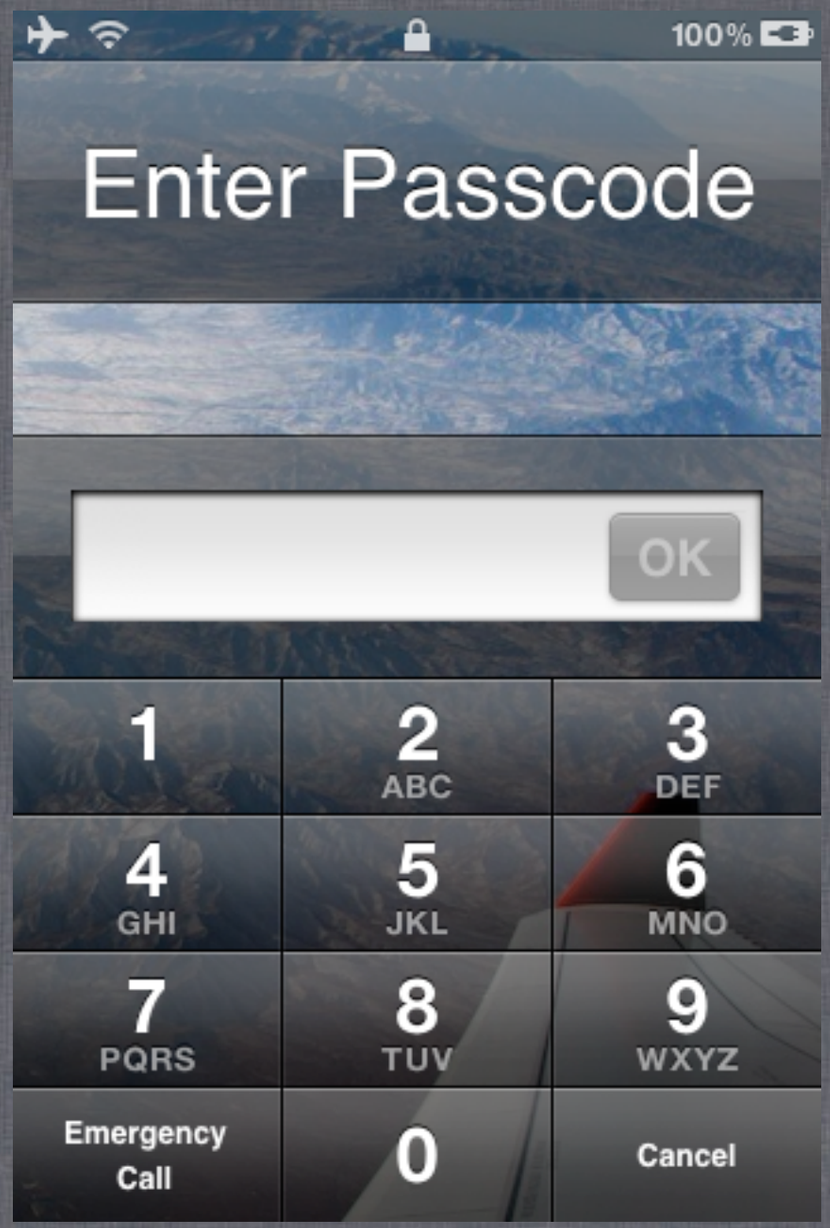
- Passcode is used to compute passcode key
  - Computation is tied to hardware key (UID/UID+)
  - Same passcode will yield different passcode keys on different devices!
- Passcode key is required to unlock most keys from the system keybag
  - Most files are protected with NSProtectionNone and don't require a passcode
  - Most keychain items are protected with ...WhenUnlocked or ...AfterFirstUnlock and require a passcode

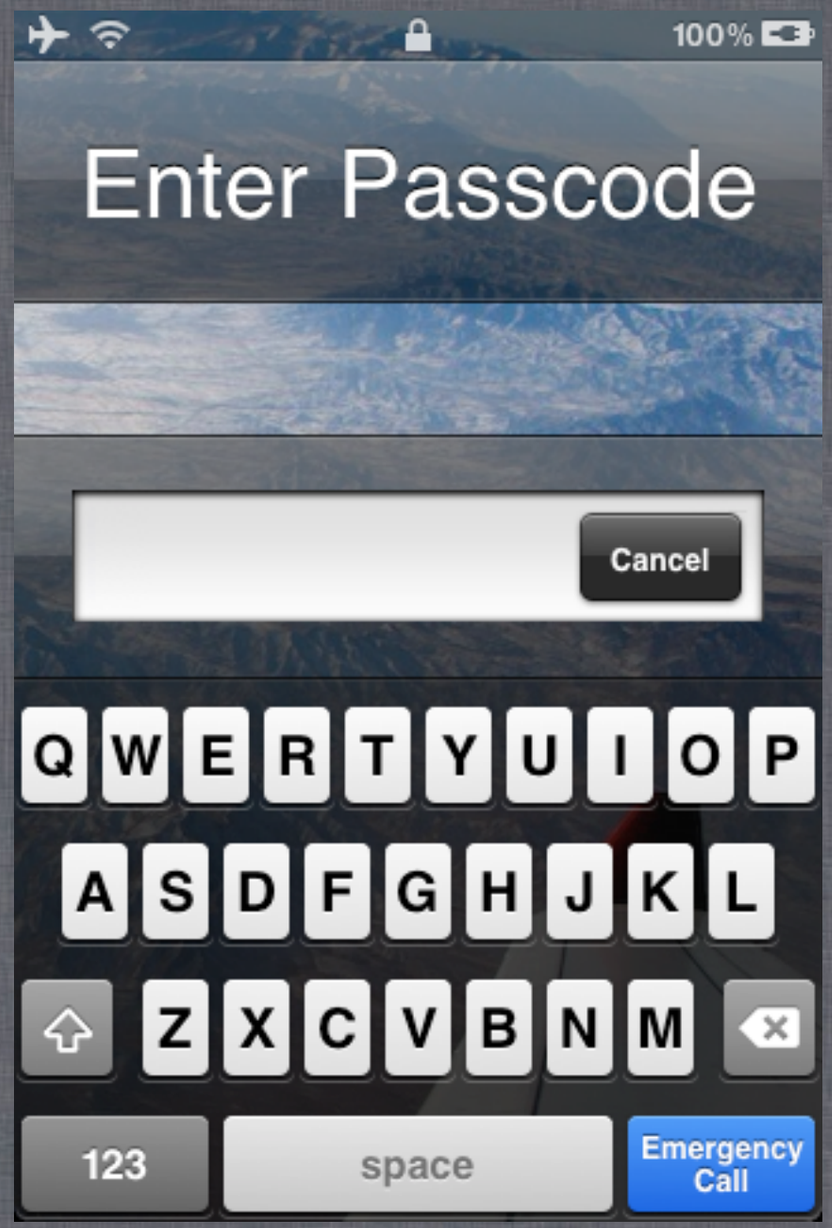
iOS 4+ Passcode

- Passcode-to-Key transformation is slow
- Offline bruteforce currently is not possible
  - Requires extracting UID/UID+ key
- On-device bruteforce is slow
  - 2 p/s on iPhone 3G, 7 p/s on iPad
- System keybag contains hint on password complexity

iOS 4+ Passcode







iOS 5 Keychain

- SQLite3 DB, all columns are encrypted
- Available protection classes:
  - kSecAttrAccessibleWhenUnlocked (+ ...ThisDeviceOnly)
  - kSecAttrAccessibleAfterFirstUnlock (+ ...ThisDeviceOnly)
  - kSecAttrAccessibleAlways (+ ...ThisDeviceOnly)
- Random key for each item, AES-GCM
- Item key is protected with corresponding protection class master key

2	Class	Wrapped Key Length	Wrapped Key	Encrypted Data (+Integrity Tag)
0	4	8	12	

iOS 5 Storage

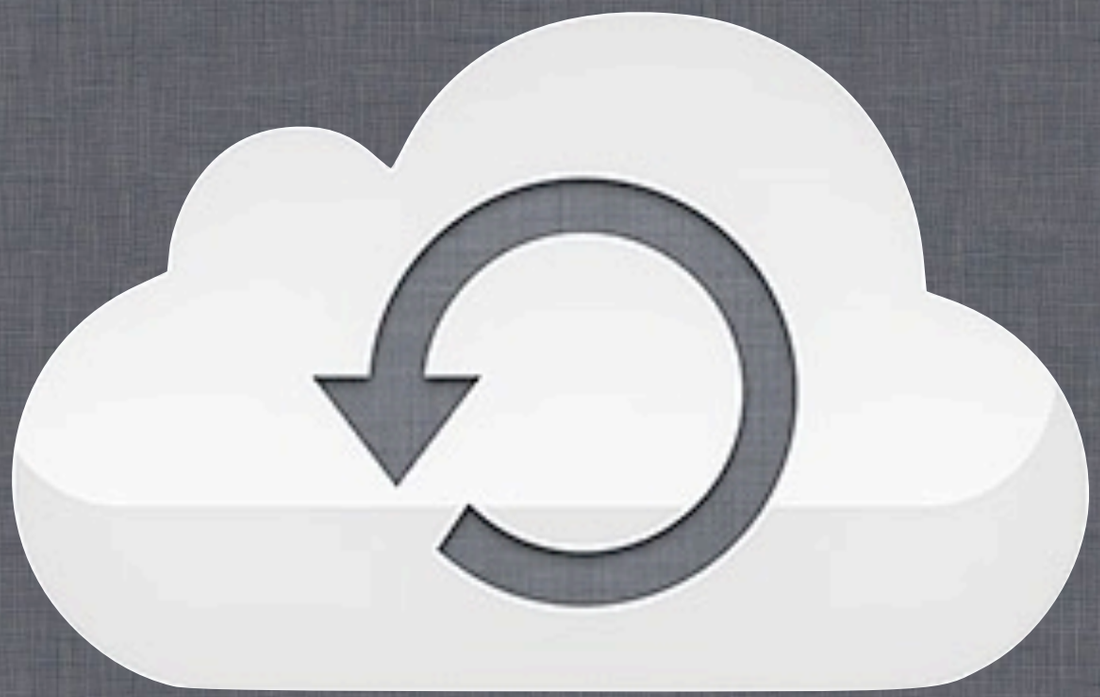
- Only User partition is encrypted
- Available protection classes:
  - NSProtectionNone
  - NSProtectionComplete
  - NSFileProtectionCompleteUntilFirstUserAuthentication
  - NSFileProtectionCompleteUnlessOpen
- Per-file random encryption key
  - File key protected with master key is stored in extended attribute com.apple.system.cprotect
- No protection class – partition key is used
  - Filesystem metadata and unprotected files
  - Transparent encryption and decryption (same as pre-iOS 4)

iCloud

- Introduced in Oct 2011
- Introduced with iOS 5
- Successor to MobileMe, .Mac, iTools
- 5 GB free storage
- Up to 50 GB paid storage

iCloud Services

iCloud Backup



iCloud Backup: What ?

- Messages (including iMessages)
- Application data
- Device settings
- Camera roll (photos and videos)
- Visual voicemails
- Purchases (music, movies, TV, apps, books)
- Home screen arrangement
- Ringtones

iCloud Backup : When?

Backup runs daily when device is:

- Connected to the Internet over Wi-Fi
- Connected to a power source
- Locked

Can force backup

- Settings - iCloud - Storage & Backup - Back Up Now

iCloud Backup : How?

- Can be enabled/disabled per device, at any time
- iCloud Control Panel shows list of backed up devices and space/quota used (no access to data though)

iCloud Backup : How?

- Restore can only be done on clean device (i.e. new one or after a firmware restore)
- No other ways to access/ browse/download backups in iCloud
- Challenge accepted! :)

Master Plan

- Capture traffic while device restores from the iCloud
- Deduce protocol from traffic dump(s) and some RE, if needed
- Create a tool that will speak iCloud backup/restore protocol and pull data from the cloud

Capturing Traffic

- iCloud backup/restore happens over SSL
- Need to plant CA certificate to do MITM
- Trivial for backup (just install “profile”)
- Not-so-trivial for restore: clean device, no usual apps (Safari, Settings) and limited UI
- Yet, there are ways to do this
  - Tethered jailbreak and add certificate to TrustStore.sqlite3
  - iPhone Configuration Utility may also help
  - Kiosk-mode hack/bypass on iOS anyone? :)

Traffic Flow

- Authenticate: send Apple ID and password, receive authentication token
- Get list of devices/backups
- Get OTA backup keybag
- Build list of files to download
  - iCloud backups are incremental and 3 most recent snapshots are maintained
- Download file data (encrypted and chunked, from Amazon/Microsoft clouds!)

Protocol

- Dynamic: endpoints depend on Apple ID
- Built on Google Protocol Buffers (mostly)
- Files are split into chunks
- Apple provides file-to-chunks mapping, chunk encryption keys, and full request info to 3rd-party storage provider (Amazon/Microsoft)
- Encryption key depends on chunk data (deduplication?)

Protocol Flow

/mbs/<personId>

List of backups

/mbs/<personId>/<backupUDID>/getKeys

OTA backup keybag

/mbs/<personId>/<backupUDID>/<snapshotId>/listFiles

File manifest

/mbs/<personId>/<backupUDID>/<snapshotId>/getFiles

File auth tokens

/mbs/<personId>/authorizeGet

Info about containers of chunks for files (FileGroups)

Request containers of chunks (FileGroups)

Containers of chunks

iCloud

amazon web services™

Message Definitions

```
message FileGroups {
  repeated FileChecksumStorageHostChunkLists file_groups      = 1;
  repeated FileError                         file_error       = 2;
  repeated FileChunkError                    file_chunk_error = 3;
  optional uint32                            verbosity_level  = 4;
}

message FileChecksumStorageHostChunkLists {
  repeated StorageHostChunkList       storage_host_chunk_list      = 1;
  repeated FileChecksumChunkReferences file_checksum_chunk_ref_list = 2;
}

message FileChecksumChunkReferences {
  required bytes          file_checksum    = 1;
  repeated ChunkReference chunk_references = 2;
}

message ChunkReference {
  required uint64 container_index = 1;
  required uint64 chunk_index     = 2;
}
```

Message Definitions

```
message FileChecksumStorageHostChunkLists {
  repeated StorageHostChunkList       storage_host_chunk_list      = 1;
  repeated FileChecksumChunkReferences file_checksum_chunk_ref_list = 2;
}

message StorageHostChunkList {
  required HostInfo  host_info                                    = 1;
  repeated ChunkInfo chunk_info                                   = 2;
  required string    storage_container_key                        = 3;
  required string    storage_container_authorization_token = 4;
}

message HostInfo {
  required string         hostname                        = 1;
  required uint32         port                            = 2;
  required string         method                          = 3;
  required string         uri                             = 4;
  required string         transport_protocol              = 5;
  required string         transport_protocol_version = 6;
  required string         scheme                          = 7;
  repeated NameValuePair headers                          = 8;
}
```

Message Definitions

```
message FileChecksumStorageHostChunkLists {
  repeated StorageHostChunkList       storage_host_chunk_list      = 1;
  repeated FileChecksumChunkReferences file_checksum_chunk_ref_list = 2;
}

message StorageHostChunkList {
  required HostInfo  host_info                                    = 1;
  repeated ChunkInfo chunk_info                                   = 2;
  required string    storage_container_key                        = 3;
  required string    storage_container_authorization_token = 4;
}

message ChunkInfo {
  required bytes  chunk_checksum        = 1;
  optional bytes  chunk_encryption_key = 2;
  required uint32 chunk_length          = 3;
}
```

Encryption

- Data stored at 3rd-party storage providers is encrypted
- Apple has encryption keys to that data
- Few files are further encrypted using keys from OTA backup keybag
  - Probably files encrypted by Data Protection
- Keychain items are encrypted using keys from OTA backup keybag
- Need key 0x835 (securityd) to decrypt most keys from OTA backup keybag

Encryption

	Apple ID + Password or AuthToken	Key 0x835
Files	+	–
Files encrypted by Data Protection	+	+
Keychain Records	+	+

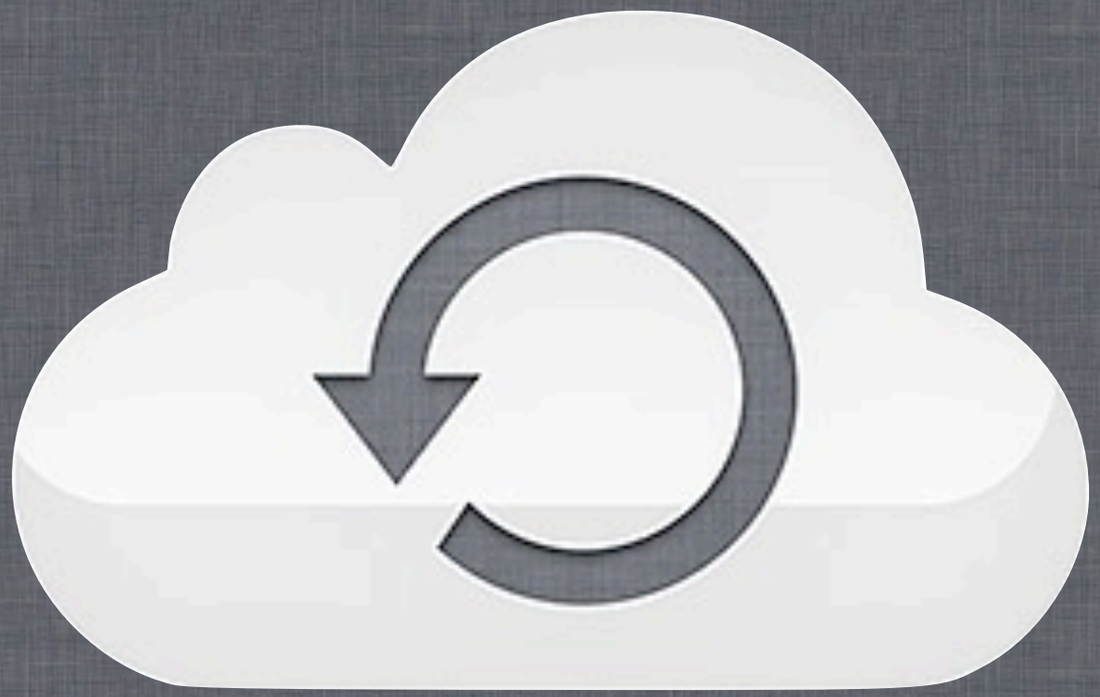
Key 0x835

- Device-specific key, computed from unique hardware-embedded key (UID) at startup
- Apple can have a list of those keys
- Apple claims not to store UID key but it's not clear whether they store keys derived from it
- Few things that are encrypted in iCloud backups are encrypted using key 0x835

Summary

- There is no user-configurable encryption for iCloud backups
- iCloud backups are stored in Microsoft and Amazon clouds in encrypted form
- Apple holds encryption keys and thus have access to data in iCloud backups
- If Apple stores 0x835 keys then it can also have access to Keychain data (i.e. passwords)
- Apple may have legal obligations to do this (LE, USG, etc)

Conclusions



There is no privacy or confidentiality

(not exactly news, go ask ex-CIA director if you don't believe us)

Thank You!

Questions?

Dark and Bright Sides of the iCloud (In)Security

Andrey Belenko
Sr. Security Engineer
viaForensics

Dmitry Sklyarov
Lead Analyst
Positive Technologies