Security Reloaded

VisualWorks Security New Generation

Martin Kobetic
Cincom Smalltalk Engineering
STIC 2012

What is Security?

Where?
* good security is invisible.
* presents itself when something is wrong.
* quickly spreading everywhere

How?
* Usually lurks at the bottom of technology stacks
  - WWW, e-mail, file transfer, remote shell, ...
  - HTTPS, SMTPS, POP3S, IMAPS,...
  - SSL/TLS, SSH, X.509, PKCSx,....
  - AES, RC4, SHA, MD5, ...

What's new?

cryptographic primitives (Xtreams-Crypto)

* open to external cryptographic toolkits
  - MS Windows - CNG/BCrypt (Vista and later)
  - Others - libcrypto (OpenSSL)
  - extendable to other toolkits (e.g. certified, hardware backed, ...)
* new streaming hash and cipher APIs
  - better performance and scalability
* new public key APIs
  - focused around key types rather than algorithms
* new algorithms - library dependent
  - e.g. SHA512

What's new?

PKCS5 - password based encryption/signing

no API changes

PKCS8 - secure private key storage/transport

minor changes dues to new public key APIs

X509 - public key certificates

new signing API, SHA-2 support

SSL/TLS - secure sockets

* complete overhaul and update
* version support SSL3.0, TLS 1.0 - 1.2
* new algorithms AES, SHA-2
* flexible session management
* pluggable certificate/key management

Hashes - Old

```
hello := 'Hello' asByteArrayEncoding: #ascii.
(SHA hash: hello) asHexString.

buffer := ByteArray new: 16384.
hash := MD5 new.
file := (ObjectMemory imageFilename
                withEncoding: #binary) readStream.
[      [      [ file atEnd ] whileFalse: [  | read |
                read :=
                        file nextAvailable: buffer size
                                into: buffer
                                startingAt: 1.
                hash updateWith: buffer from: 1 to: read ].
        ] ensure: [ file close ].
] timeToRun.
hash digest asHexString.
```

Hashes - New

```
file := ObjectMemory imageFilename reading.
sha := file hashing: 'SHA512'.
[ nil writing write: sha ] timeToRun.
sha close.
sha digest asHexString.

(nil writing hashing: 'MD5')
        write: hello;
        close;
        digest
```

Ciphers - Old

```
message := 'Hello World!' asByteArrayEncoding: #ascii.
key := 'Open Sesame!!!!!' asByteArrayEncoding: #ascii.

((ARC4 key: key) encrypt: message) asHexString.

cipher := AES key: key.

cipher := CipherBlockChaining on: cipher.
iv := ByteArray new: 16 withAll: 1.
cipher setIV: iv.

cipher := BlockPadding on: cipher.
(cipher encrypt: message) asHexString
```

Ciphers - New

```
cipher := 'vault' asFilename writing
                    encrypting: 'AES'
                    mode: 'CBC'
                    key: key
                    iv: iv.
padded := cipher closing: [ | pad |
                    pad := iv size - (padded position \\ iv size).
                    padded write: pad from: pad repeating.
                    cipher close ].
hash := padded compressing hashing: 'SHA1'.
file := ObjectMemory imageFilename reading.
[ hash write: file; close ] ensure: [ file close ].
hash digest asHexString
```

Public Key - Old

```
message := 'Hello World!' asByteArrayEncoding: #ascii.

keys := RSAKeyGenerator keySize: 1024.
keys publicKey.

rsa := RSA new privateKey: keys privateKey.
rsa useMD5.
sig := rsa sign: message.

rsa publicKey: keys publicKey.
rsa verify: sig of: message
```

Public Key - New

```
digest := (message reading hashing: 'SHA1')
                    -= 0; close; digest.

pri := PrivateKey RSALength: 2048.
pub := pri asPublicKey.
[      sig := pri sign: digest hash: 'SHA1' padding: 'PKCS1'
] ensure: [ pri release ].

[      pub verify: sig of: digest hash: 'SHA1' padding: 'PKCS1'
] ensure: [ pub release ]
```

PKCS8 - Private Key Storage

```
bytes := ByteArray new readWriteStream.
password := 'Open Sesame' asByteArrayEncoding: #ascii.
pri := PrivateKey RSALength: 2048.
[      pri asPKCS8Key writeOn: bytes password: password
] ensure: [ pri release ].
bytes := bytes contents.

pri := PKCS8 readKeyFrom: bytes readStream password: password.
pri := pri getKey.
pri release.
```

X509 Certificates

```
pri := PrivateKey RSALength: 2048.
pub := pri asPublicKey.
name := Name new CN: 'STIC 2012'; yourself.
certificate := Certificate new
        serialNumber: 1000;
        issuer: name;
        subject: name;
        notBefore: Date today;
        notAfter: (Date today + 100 days);
        publicKey: pub asX509Key;
        forKeyExchange;
        yourself.
certificate signUsing: pri hash: 'SHA1' padding: 'PKCS1'.
pub release.
certificate publicKey getKey
```

TLS Client

```
LoggingTool open.
'https://www.google.com' asURI get.

socket := ('www.google.com' asIPv4: 443) connect.
context := TLSContext newClientWithDefaults.
connection := context newConnection: socket.
connection when: TLSAnnouncement do: [ :m |
	Transcript cr; print: m; flush ].
connection connect: [ :cert | cert inspect. true ].
stream := connection readAppendStream.
(HttpRequest get: 'https://www.google.com') writeOn: stream.
stream flush.
HttpResponse readFrom: stream.
stream close. connection close. socket close.
context release
```

TLS Server

```
certificates := TLSCertificateStore newWithDefaults.
certificates certificate: (Array with: certificate) key: pri.
context := TLSContext newServerWithDefaults.
context certificates: certificates.
server := (WebAdaptorConfiguration new
	addExecutor: (WebFileServer prefix: #('files') directory: '.' asFilename);
	addExecutor: WebEcho new;
	transport: (HTTPSTransportConfiguration new
			serverContext: context;
			marshaler: WebMarshalerConfiguration new )
) newAtPort: 4433.
server when: #addingServerConnection:in: do: [ :c :a |
	Transcript cr; print: c ].
server start.
server stop.
context release. certificates release.
```

TLS Context

certificates - TLSCertificateStore
* certificate management and processing
* public/private key management and operations

sessions - TLSSessionCache
* TLS session management

suites - TLSCipherSuite
* bulk encryption/hashing setup
* key exchange setup

compressions - TLSCompression
* compression setup

SSH Client

```
home := '$(HOME)' asLogicalFileSpecification asFilename.
user := home tail.
keys := SSH2Keys fromUser: home.
configuration := SSH2Configuration new keys: keys.
socket := ('localhost' asIPv4: 22) connect.
client := configuration newClientConnectionOn: socket.
client when: SSH2Announcement do: [ :m | Transcript cr; print: m ].
client connect: user.
session := client session.
session exec: 'ls -l'.
session put: 'ssh' to: '/dev/shm/'.
session close.
client close.
configuration release
```

SSH Client

```
home := '$(HOME)' asLogicalFileSpecification asFilename.
user := home tail.
keys := SSH2Keys fromUser: home.
configuration := SSH2Configuration new keys: keys.
socket := ('localhost' asIPv4: 22) connect.
client := configuration newClientConnectionOn: socket.
client when: SSH2Announcement do: [ :m | Transcript cr; print: m ].
client connect: user.
tunnel := client tunnelTo: 'localhost' port: 5555.
tunnel writing write: 10000 from: 42 repeating; close.
tunnel close.
client close.
configuration release
```

SSH Server

```
keys := SSH2KeysTest sampleKeys.
configuration := SSH2Configuration new keys: keys.
listener := ('localhost' asIPv4: 2222) listen.
[ socket := listener accept ] ensure: [ listener close ].
server := configuration newServerConnectionOn: socket.
server when: SSH2Announcement do: [ :m | Transcript cr; print: m ].
server accept.
server waitForDisconnect.
queue := server openQueueAt: 'test:6666'.
worker := [ tunnel := queue next.
        [ nil writing write: tunnel reading ] ensure: [ tunnel close ] ] fork.
server closeQueue: queue.
server close.
configuration release
```

To Do

crypto fallback

BCrypt: Vista and later only, poor support for DSA/DH
libcrypto: < 1.0.0 no public key crypto (OSX, older Linux releases)

TLS

renegotiation_info
smarter client session management
client authentication
DHE_RSA key exchange
elliptic crypto (ECDH/ECDSA)

Summary

* more implementation options
* more algorithms
* highly extensible
* significantly faster
* up to date SSL/TLS support
* capable of handling server side load

Contact Info

Star Team (Smalltalk Strategic Resources)

sfortman@cincom.com	Smalltalk Director
athomas@cincom.com	Smalltalk Product Manager
jjordan@cincom.com	Smalltalk Marketing Manager

http://www.cincomsmalltalk.com