Security and Cryptography – What's up C12N ?

Sébastien Pouliot
sebastien@ximian.com

October 25, 2006

Novell®

Quick Overview

Secure Execution Environment

- What ? Runtime, Code Access Security, Class libraries
- When ? Mono 1.2, Mono 2.0 ...
- Who ?

Crimson Project

- Why and How ?

Tools

- Gendarme
- Monoxide

Secure Execution Environment

Runtime, Code Access Security & Class libraries

Mono 1.2

Goal

- Implement a usable subset of Code Access Security (CAS)
  - Runtime, JIT and class libraries

Results

- SecurityManager and most related managed classes
  - permissions, attributes, policies, membership conditions...
- Support for declarative security (i.e. attributes)
  - Demand, Assert, Deny, PermitOnly, LinkDemand, InheritanceDemand ...
- Stack Walks
  - with one or two appdomains (sandbox)

What's missing

- Imperative security, most permissions in BCL, **audit** ...

Mono 1.2 / Demo

NRobot

Author: Stuart Ballard

Home page: http://home.gna.org/nrobot/

License: GPL

Best open source

.NET sandbox

Mono 1.2 / Findings

Lack of applications requiring CAS

- almost every of them are from Microsoft (IE, SQL Server 2005)
- egg/chicken problem – at least for Mono

Very few applications/libraries are CAS aware

- not really an egg/chicken problem – considering MS implementation has been available for years

Lack of interest to contribute

- maybe it's not sexy enough ;-)

just remember

It's there but it's not ready for production use!

Mono 2.0

New security features (Fx 2.0)

- non CAS related
  - SslStream, System.Security.Cryptography.Pkcs, ...
- CAS
  - New features
  - Much wider API to cover (permission, audit)

Complete missing features

- from Fx 1.0/1.1

Help!

- Want it ? Want more ? Yes you can help!
- The “good thing” when there is so much to do is that there are tasks for everyone ;-) No security knowledge is required!

Crimson Project

Crimson / Goals

Provide a new playground to:

- Extend,
- Simplify, and
- Optimize

cryptography in the .NET framework.

Why ?

- it's not part of .NET so it doesn't have to be inside Mono
  - and it can be useful to everyone, everywhere (runtime agnostic)
- will be easier to change Crimson than Mono
  - e.g. stability requirements, export restrictions paperwork...
- different licenses may be required (but MIT.X11 is preferred)

Crimson / Extend

by providing ...

- alternative implementations of existing cryptographic algorithms
  - Existing native librairies, e.g. libmhash support is already in SVN
  - Optimized (e.g. unrolled) managed implementations
- new cryptographic algorithms unavailable in .NET
  - e.g. new hash algorithms, like TIGER (available in libmhash)
- new tools
  - e.g. benchmarking

Crimson / Simplify

The existing .NET design for cryptography is

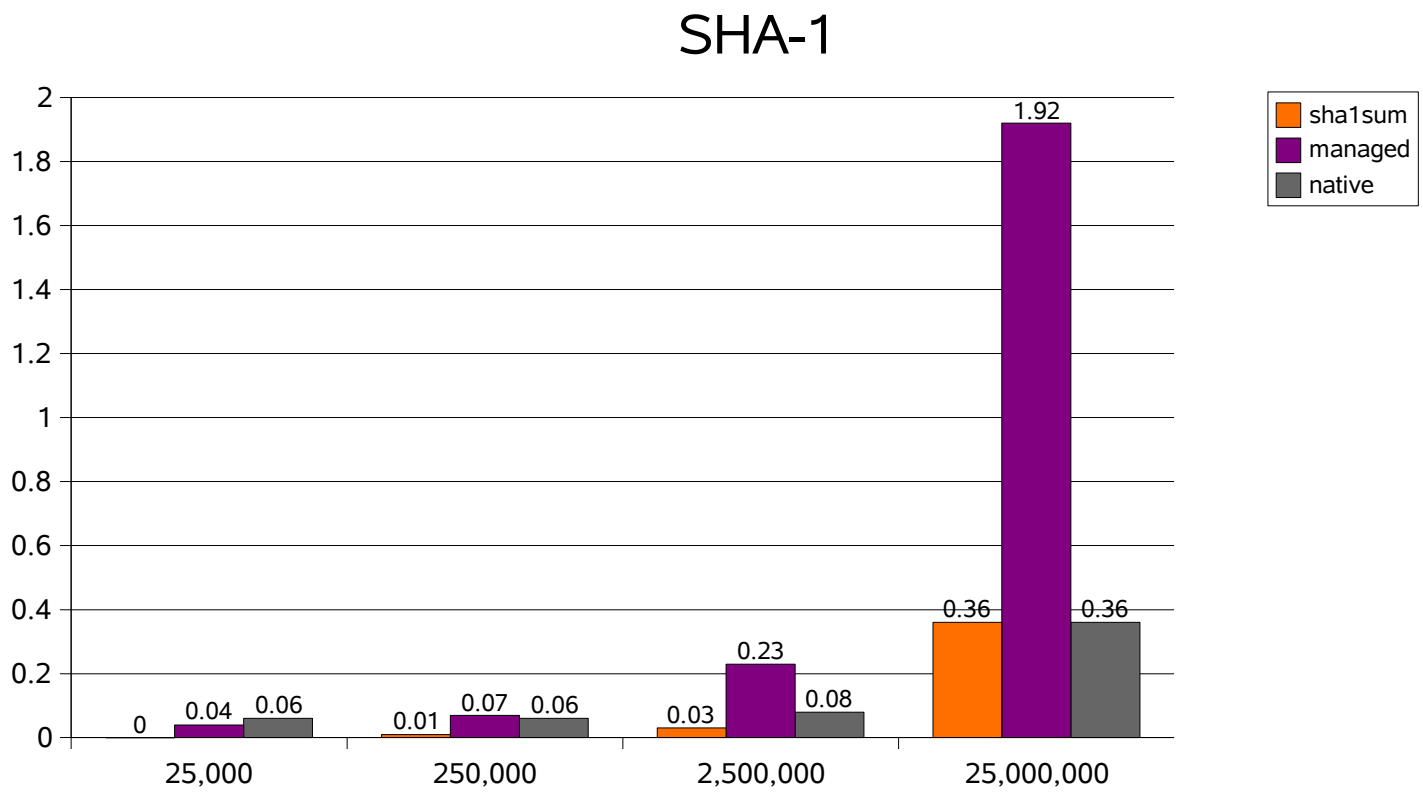
- 80% nice, for new stuff
- 20% bad, mostly because of CryptoAPI workarounds
- but you need to know what you're doing, so many people ends up reinventing the wheel.

Solution

- Provide easier API mapping to user, not cryptographic, tasks
  - e.g. 2.0 introduces ProtectedData and ProtectedMemory classes, but it doesn't reuse the design strengths

WARNING: don't let security people define the easy API ;-)

Crimson / Optimize



sha1sum *versus* SHA1Managed *versus* libmhash SHA1

[Recent] Tools

Tools

The partial work done on securing the class libraries has shown the need for tools to assist in auditing Mono code base.

This resulted in two different tools:

- Gendarme and
- Monoxide

Gendarme

Built on top of **Cecil**, Gendarme is split into 3 main components:

- **Runner(s)**
  - User interface to run rules on selected assemblies;
  - Currently only a console-based runner exists and output directly to the console, XML or HTML files;
- **Framework**
  - Shared logic to ease writing rules (helpers) and writing runners (avoid code duplication);
- **Rules**
  - Anything you want / and that you don't want to audit again;
  - For me it mostly means, find potential security issues and promote Gendarme rule development (e.g. UseStringEmptyRule).

Gendarme / Demo

btw Gendarme could use a logo...

Gendarme / Future

- **More** rules!
- More polish
  - add inclusion/exclusion of rules
  - add ignore (*notabug*) lists
  - I18N
  - more documentation ...
- GUI runners like
  - a standalone GTK# runner
  - a standalone SWF runner (for Windows users); and
  - a MonoDevelop add-in (e.g. similar to the NUnit add-in)

Monoxide

Same goal as Gendarme - find defects in compiled assembly, but with a different strategy!

Extensible assembly viewer with plug-ins for:

- Assembly dependencies graphs
- Callers analysis
- IL viewer : source and graphs
- Declarative security source

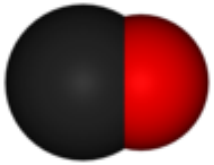
Yes, it's currently very security oriented.

Monoxide / Demo

Warning

Carefully review any license agreements before using Monoxide on assemblies that you have not written yourself.

In many cases using Monoxide could be interpreted as some kind of reverse engineering.



Monoxide / Future

User interface improvements

- add zoom +,-,fit... to graphical views
- better treeview (icons like MD)
- add colors (e.g. icall, unsafe)
- add printing support

More plug-ins

- detect unsafe code (e.g. ldflda, pointers)
- add monop-style support on class (C#)
- Inheritance graphs (dot);
- UML plug-in (non-dot);

Semi-automation (i.e. virtual tours)

Questions ?
or (better) Answers ?