



Mocana vs. Open Source

Developing Security for Embedded Devices

Included in this White Paper:

- Executive Summary
- Porting Considerations
- Security Concerns
- Hidden Costs
- Support Issues
- Code Quality
- Certification and Legal Issues

Mocana Corporation

350 Sansome Street
Suite 1010
San Francisco, CA 94104

415-617-0055 Phone
866-213-1273 Toll Free

info@mocana.com
www.mocana.com

Copyright © 2008
Mocana Corp.

Executive Summary

When embedded systems development teams investigate which security tools to include in their devices and applications, open source libraries often seem attractive. There seems to be an open source solution for virtually any security protocol, such as OpenSSL, OpenSSH, and the various flavors of “Swan” IPsec (FreeSWAN, Openswan, and strongSwan). Such projects are popular, offer loads of optional user-written add-on modules, and best of all, they’re free!

A closer observation, however, reveals that there is in fact ‘no such thing as a free lunch’, especially when it comes to implementing security in non-PC environments. Common downsides to using open source security code in production environments include:

- **Porting considerations**—Open source security products were designed for desktop systems. To adapt them to embedded devices requires costly development time for non-trivial platform ports, performance optimizations, and footprint reductions.
- **Security concerns**—Open source security code has a history of routine and significant security flaws, and of non-adherence to standards which causes interoperability issues.
- **Hidden costs**—Open source libraries *appear* to be free, but when the cost of extra development, maintenance, legal liabilities, and so on are included, the TCO (total cost of ownership) usually exceeds that of Mocana’s commercially sold and supported code. (Open source TCO can be easily calculated by using the free calculator at www.mocana.com/BuildvBuy.html.)
- **Support issues**—Lack of documentation, samples, support, and maintenance for open source means developers are on their own and must invest significant time to integrate security code, all the while raising the risk of introducing security holes into their applications.
- **Code quality**—The quality of open source code varies considerably from project to project, and even among modules in a given code base. Testers cannot take anything for granted, and will spend considerable effort on platform testing and integration efforts.
- **Certification and legal issues**—Open source security code has a history of difficulty getting and keeping FIPS validations, as well as leaving manufacturers with considerable legal exposure due to unresolved or simply ambiguous issues of patent protection, IP indemnification, licensing, and (unknown) country of origin.

Criteria For Evaluating Open Source:

- Porting Considerations
- Security Concerns
- Hidden Costs
- Support Issues
- Code Quality
- Certification and Legal Issues

Open Source Porting Considerations

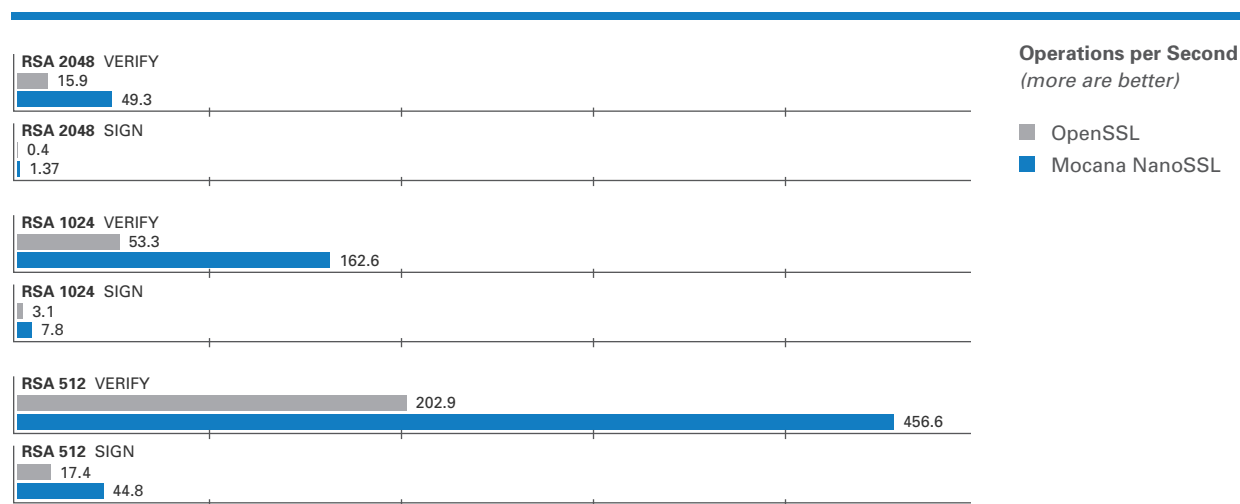
Platforms

OpenSSL, OpenSSH, and “Swan” IPsec were designed to be used on desktop systems, not embedded devices. Porting open source code to new device platforms often takes days, and engineers unfamiliar with the intricacies of encryption and authentication algorithms can unwittingly introduce defects and vulnerabilities into the security protocol during the port.

In contrast, Mocana® NanoSSL™, NanoSSH™, and NanoSec™ packages provide abstraction layers for more than 20 operating systems and 50 silicon CPU platforms, resulting in typical port times of under two hours. Additionally, Mocana documentation describes a simple 8-step process for porting its code to any operating system. Mocana’s Nano- products can even run *without* an operating system.

Performance

The performance of a product’s security components is one of the biggest factors affecting overall product behavior. (For example, on desktop systems it’s usually very obvious when a scheduled antivirus scan begins.) A sluggish security implementation-especially in comparatively low horsepower devices-can become a communications bottleneck. Unfortunately, OpenSSL and other open source security implementations are notoriously slow and computer-intensive, largely because they cannot simultaneously be optimized for specific platforms *and* retain their platform independence.



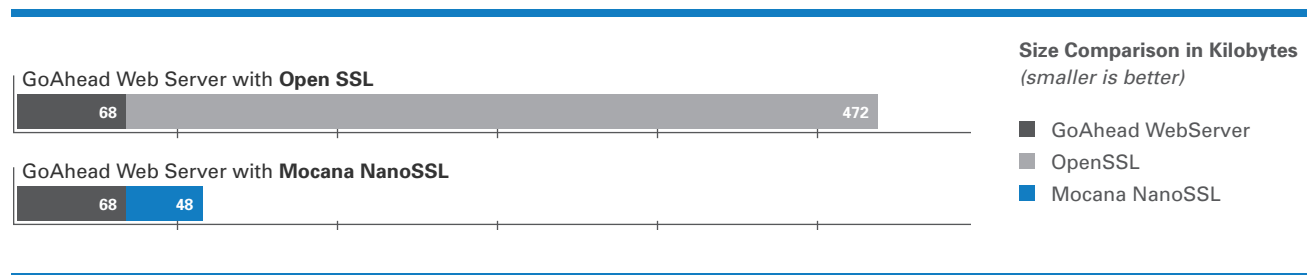
Assembly language optimization is a key contributor to Mocana performance. For example, on a 266 MHz ARM Intel XScale IXP420 processor, NanoSSL provides a tremendous performance boost over OpenSSL.

Size

Freeware security products were not designed for embedded devices, and their subsequent ports exhibit rather large memory footprints, often entirely too large to fit into the target device at all. Additionally, open source solutions are developed independently of each other, resulting in redundant code for common utility functions. Commercial products that integrate multiple freeware code bases become bloated because of such redundant code.

Mocana solutions are specifically designed for embedded devices. NanoSSL and NanoSSH memory footprints are just 25% and 50% of OpenSSL and OpenSSH, respectively, enabling applications to fit where there might not otherwise be space, and saving memory for applications instead of their security functions. And Mocana solutions leverage a single cryptographic code base, eliminating function redundancy. The more security protocols implemented, the greater the footprint savings!

The following footprint comparisons, obtained by testing with the GoAhead Web server running on Microsoft® Windows®, show the typical advantage of Mocana NanoSSL over OpenSSL:



And when optimized for speed instead of size, the difference between NanoSSL and OpenSSL is even greater: using the same GoAhead Web server running on Microsoft® Windows®, the NanoSSL footprint is only 9.8% of the OpenSSL footprint.

Open Source Security Concerns

Adding value by adding security is the primary reason to add SSL, SSH, or IPsec to a product. But ironically, using open source security libraries can actually introduce new security problems (exemplified by the recent randomness issues with Ubuntu and Debian distributions of Linux) and cause interoperability issues. In juxtaposition, Mocana NanoSSL, NanoSSH, and NanoSec are specifically designed to ensure that devices, applications, and operating systems using Mocana code are secure and interoperable.

Vulnerabilities

OpenSSL has a history of releases later found to contain multiple vulnerabilities, detailed on the OpenSSL website: www.openssl.org/news/vulnerabilities.html. And although open source projects do respond to community pressure to fix major security bugs, it's up to user software engineers to keep track of it all, apply the relevant patches in the correct order, and retest the affected applications and systems whenever a new flaw is uncovered.

By comparison, when Mocana NanoSSL, NanoSSH, or NanoSec is used, Mocana takes care of all issues around security vulnerabilities. Mocana products are continuously and exhaustively tested, and Mocana engineers monitor an array of security resources, such as:

- US-CERT Vulnerability Database (www.kb.cert.org/vuls/)
- Dark Reading (www.darkreading.com)
- Slashdot (www.slashdot.org)
- Secunia (www.secunia.com)
- Offensive Computing (www.offensivecomputing.net)

If an issue surfaces that affects a Mocana security product, a patch is immediately released and affected customers are personally notified. If required, Mocana will even send an entirely new library, eliminating the patching process altogether.

Interoperability

Open source security code is not always compliant with industry standards. For example, The current OpenSSL DTLS implementation (OpenSSL 0.9.8g, 19 October 2007) deviates in several aspects from RFC 4347, *Datagram Transport Layer Security*, making it non-interoperable with many compliant products.

Mocana is a full member of the VPNC (Virtual Private Network Consortium, www.vpnc.org). Mocana guarantees interoperability, and will fix any interoperability problems faced by our customers quickly and for free.

Open Source Hidden Costs

As the preceding topics have explained, the only thing free about open source libraries is the acquisition price. As soon as the code is downloaded, the TCO (total cost of ownership) begins to climb. Engineers and architects must invest considerable time to adapt open source code to embedded device environments, integrate and optimize code with minimal assistance, and test and refactor code that often is not held to the same high standards as commercial-grade products. Add in the likelihood of security and interoperability issues, and loss of revenue due to recalled product and poor company reputation becomes a growing concern.

Mocana has developed a free, easy to use online calculator for determining hidden costs associated with using open source. Visit www.mocana.com/BuildvBuy.html.

In addition to engineering's investment, there are real and potentially very large legal risks and financial obligations resulting from IP, licensing, and export issues for open source products.

Open Source Support Issues

It's after hours, an anxious customer is waiting, and things just aren't working. Who can the engineering team call for help? With open source projects, documentation and support options are limited to say the least. It's a different story with Mocana NanoSSL, NanoSSH, and NanoSec, which ship with *all* the information needed for successful product integrations. Additionally, Mocana tech support provides reliable, fast, personal guidance and real-time assistance.

Documentation

Open source projects seldom attract the attention and participation of professional technical writers, resulting in incomplete, inconsistent, confusing, and hard to use documentation. Often, little is provided beyond release notes. The consequence is that it takes much more time than it otherwise would

for engineers to integrate freeware, test it, and be sure the code is operating correctly and efficiently.

In addition to the expected *Release Notes*, Mocana's Nano- product line ships with a full documentation suite: *Installation Guide*, porting instructions, sample code, product guides that provide background and detailed best practices, integration instructions, and an explicit HTML-viewable *API Reference*.

Tech support

With open source products, support typically is limited to the project's website FAQs (which often must be manually searched) and posting to user forums or generic developer email aliases... and hoping for the best.

Mocana understands deadlines, and Mocana Tech Support is available 24/7/365 to provide dependable, personal, hands-on guidance and implementation assistance with quick turnaround time.

Maintenance

Due to their very nature, open source projects don't provide proactive maintenance (patches, periodic updates, software upgrades, and emerging standards support). Engineers using open source code must be sure to frequent such projects' websites to keep informed of new releases, and then invest additional time to download the correct patches, apply them in the correct order, re-port, retest, and reoptimize. Additionally, because open source is not designed for embedded systems, new releases typically exhibit issues with respect to backwards compatibility, and typically *all work performed to port open source code to embedded systems must be repeated*.

Mocana automatically notifies customers of patches and new releases, and provides unique logon instructions for software downloads. And Mocana code is guaranteed backwards compatible, with conversion functions to make it easy to use new functionality (for example, new key blob formats) without complex application code refactoring.

Open Source Code Quality

At the heart of any security solution is the code itself. Freeware does not provide the guaranteed robust, well-architected, high quality solution that Mocana NanoSSL, NanoSSH, and NanoSec deliver.

The Mocana Nano- product line uses a common cryptographic code base, is API-based, and adheres to Mocana's strict *Coding Guidelines and Style* guide that covers file organization, naming conventions, declarations and types, error handling, memory management and more, ensuring that the code is of the highest possible quality.

Architecture

OpenSSL, OpenSSH, and "Swan" IPsec are standalone, socket/stream interfaces, not API-based solutions. To use these products in embedded devices, applications must hook directly into the open source function calls. When the freeware functions change, so too must the application code.

Mocana Nano- products are designed for embedded systems: they are *ROM-able* (code can run in ROM, not just RAM), reentrant, and use an asynchronous event driven architecture. The interface is via clear, documented, easy to use ANSI C API functions that do not change over time, eliminating any need to refactor and retest applications when upgrading to a new release.

Integration

Both OpenSSL and OpenSSH often require lengthy operating platform ports, testing, and even function rewrites in assembly for performance optimization. And whenever a new release comes along, *the lengthy process must be repeated*.

Mocana NanoSSL, NanoSSH, and NanoSec provide abstraction layers for more than 20 operating systems and 50 silicon CPU platforms, as well as easy to use and fully documented API functions, resulting in typical ports and integrations in less than two hours.

Features

Open source projects often do not fully implement a protocol's specifications. For example, FreeS/WAN does not support aggressive mode.

Mocana code isn't open source, but source code is provided to customers. And Mocana certainly supports open standards, meaning the fullest, most

unambiguous support for IETF standards as formalized in the RFCs. (And yes, that means that Mocana NanoSec includes aggressive mode support.)

Status codes

Open source projects take “agile” development methods to the extreme, and therefore often fail to design in items such as detailed error codes. Most open source projects use a single error code, -1, for every error!

Mocana software includes over 900 unique status codes with macro defines for easy use, such as `ERR_SSH_TRANSPORT_BAD_MAC` and `ERR_SSH_SEND_ACK_FAIL`. Developers and applications don’t need to guess why a call might have failed, and instead can implement appropriate logic (which reduces tech support expenses) or inform users as necessary (resulting in a better user experience).

Memory leak detection

Open source tools are typically focused on functional essentials, and frequently expect the systems into which they’re integrated to include virtual memory managers. When the inevitable developer issues surface, such as tracking down memory leaks, considerable debugging time can be expended, and additional development tools, such as code analyzers, may be needed.

Mocana NanoSSL, NanoSSH, and NanoSec include an easy to use memory leak debugger, enabled by defining a single build flag:

`__ENABLE_MOCANA_DEBUG_MEMORY__`. The debugger detects memory leaks, double frees, and other memory issues. The file and line of *every* call to `malloc` and `free` are recorded. Then a single function call at the end of an application’s cleanup sequence dumps the information to a standard console or telnet port, depending on the code configuration.

Testing

By definition, open source projects are created by committee, and the primary focus is function. Testing is typically done in an ad hoc manner, with “customers” continuously providing beta testing.

This is in sharp contrast to the rigorous, continuous, integrated testing to which the Mocana Nano- product line is subjected:

- **Test monkeys**—Before a Mocana engineer even checks in new code to the source control repository, the code is exercised by a test monkey on the local development machine, ensuring that the code basically functions as expected.

Following code check-in, test monkeys running on more than two dozen OS-platform combinations automatically build the entire Mocana software suite, run more than 650 functional tests, and automatically notify Mocana developers and test engineers of any build or test errors.

- **Standalone test tools**—In addition to in-house module tests, commercial standalone test tools, such as ANVL from Ixia and Insight from Klocwork, are used to ensure that the Mocana Nano- product line is bug-free.
- **Fuzz testing**—Providing fuzz-random data-to the inputs of a program augments test monkey suites by finding bugs that occur in unusual situations outside the normal program flow (and testing). Mocana uses commercial tools such as Codenomicon, Nessus, and SSHredder to ensure that Mocana Nano-products are free from malware vulnerabilities, network vulnerabilities, and SSH protocol vulnerabilities.
- **Third parties**—Mocana contracts with third parties, such as VPNC and OffensiveComputing, who provide independent testing and verification for protocol conformance, interoperability, performance metrics, and more.

Open Source Certification and Legal Issues

Freeware supplies code that can be integrated into products, but successful products require much more than code snippets. Customers demand solutions that meet federal certification requirements and that present no legal obstacles. Mocana Nano- products leverage FIPS 140-2 certified cryptographic algorithms, and are clear with respect to IP, licensing, and export status.

FIPS certification

FIPS 140-2 (*Federal Security Requirements for Cryptographic Modules*) certification is required for all products used by the US and Canadian governments, and is increasingly required by commercial organizations as well. Historically, open source projects have had a difficult time gaining and maintaining their FIPS 140-2 certification credentials for a variety of reasons: only binary code can be FIPS 140-2 certified, but open source is delivered as source code; FIPS certification requires several months of consistent effort, which is difficult to maintain with a volunteer team; and finally, although a binary can be created for a particular combination of OS, CPU, compiler/linker version, and defined set of build flags, any combination chosen cannot meet

the requirements of a diverse open source user community. These factors contributed to the suspension of OpenSSL's FIPS certification within six months of its initial validation. Whenever an open source security product loses its validation, so too do commercial products that include the open source code, which can place big government or banking contracts in jeopardy.

Mocana invests substantial engineering effort and infrastructure, and works with expert consultants in the FIPS certification field, to ensure that its FIPS 140-2 validated cryptographic algorithms will always maintain their certification, and by extension so will products containing Mocana code.

Legal Complications

Although engineering-related issues often top the list of security concerns, there are significant legal issues that must be considered if a product is to be successful in the market place:

- **Patent protection and IP indemnification**—The status of intellectual property issues for open source code is frequently murky, making it impossible to provide full IP indemnification for products that incorporate such code.
- **Licensing issues**—Open source code is typically covered by the GNU GPL (General Public License), which carries many conditions as to how it can be modified, included in software to be licensed, reporting and contribution requirements when the code is changed, and so on. Indeed, some interpretations of the GPL hold that any software derived from a GPL product also be released under GPL—something of a “deal breaker” for any company looking to sell (or indeed maintain unchallenged ownership of) their own code.
- **Known country of origin**—The country of origin usually cannot be determined for open source products, thereby restricting the export of devices containing open source security code.

With Mocana, IP ownership is straightforward, licensing is simple, and NanoSSL, NanoSSH, and NanoSec have CCATS numbers and are cleared for export. Furthermore, Mocana fully indemnifies its customers.

Conclusion

As shown in the table on the following page, the choice is clear. Using Mocana NanoSSL, NanoSSH, and NanoSec is the smart way to lower the TCO and provide the best security solution.

Issue	Open source	Mocana NanoSSL, NanoSSH, NanoSec
Organization	Volunteer network.	Mocana is the winner of the Red Herring 100 Top Tech Startups in North America.
Platforms	Developed for desktop environments; platform support not guaranteed.	Abstraction layers for more than 20 OSES and 50 silicon CPU platforms.
Performance	Little or no data available for performance on embedded systems.	In head-to-head tests against open source implementation, the Mocana Nano- product line typically delivers 2x to 3x the number of operations per second.
Size	OpenSSL: ~470 KB OpenSSH: ~270 KB	NanoSSL: 50 KB NanoSSH: 70 KB
FIPS 140-2	OpenSSL: Presently certified (has twice lost its certification). OpenSSH: No. Free S/MAN IPsec: No.	Common set of FIPS 140-2 certified algorithms is used by all products in the Mocana Nano-product line.
Vulnerability monitoring	No. Users must keep up-to-date on vulnerability reports, apply patches, re-port, and retest.	Yes. Mocana monitors many security resources. If vulnerabilities are found, Mocana releases patches and automatically notifies customers.
Interoperability	Known issues. No guarantees.	Guaranteed. Backed up by VPNC testing.
Documentation	Ad-hoc, incomplete, and inconsistent.	Full suite of professional technical documentation.
Tech Support	Project website FAQs, user forums, generic developer email aliases.	24/7/365. Dependable, personal, hands-on.
Maintenance	Users must monitor project websites, download and apply patches, re-port, retest, and reoptimize. Not backwards compatible for embedded systems.	Automatic notification of patches and new releases. Backwards compatible, with conversion functions for convenience.
Architecture	Designed for desktop systems. Standalone, socket/stream interfaces.	Designed for embedded systems. ROM-able, reentrant, asynchronous event driven. Fully documented ANSI C API.
Integration	Lengthy process for OS ports, testing, and even rewriting functions in assembly for performance optimization. Process must be repeated for every new release.	Easy, and typically in less than two hours. Backwards compatibility means no repeated integration is required for new releases.
Features	Often fail to fully implement a protocol's specifications.	Fullest, most unambiguous support for IETF standards as formalized in the RFCs.
Status codes	Most often a single error code, -1, for every error.	Over 900 unique status codes, with macro defines for easy use.
Memory leak detection	None.	Easy to use memory leak debugger included in all Mocana Nano- products.
Testing	Ad-hoc, volunteer, with users providing de facto beta testing.	Rigorous, continuous, integrated testing, employing test monkeys, commercial standalone test tools, fuzz testing, and third party testing and verification.
Patent protection and IP indemnification	Users assume all risk.	Ownership is straightforward, and Mocana fully indemnifies its customers.
Licensing	Covered by the GNU GPL (General Public License), which carries many conditions and is open to interpretation.	Customers granted unconditional license.
Known country of origin	Cannot be determined, which can restrict export to some markets.	Yes. US.

About Mocana

Mocana securely enables Internet-scale applications and services for connected devices. Mocana's industry-leading infrastructure software solutions ensure that wired and wireless devices, networks, and services perform and scale with the utmost security—a necessary foundation for a networked society. Customers include Philips, Dell, Cisco, Nortel Networks, Harris, Honeywell, Symbol, Net.com, and Radvision, among others.

Winner of the 2008 Frost & Sullivan Technology Innovation of the Year and 2008 Red Herring 100 Top Tech Startups in North America awards, Mocana was founded in 2002, is privately-held, and is headquartered in San Francisco, California. For more information, visit www.mocana.com.



Free Source Code Evaluation

To request a free full source code and documentation evaluation of Mocana's security solutions, visit www.mocana.com/evaluate.html.