MANAGING & ACCELERATING XML WEB SERVICES SECURITY

THE NEED FOR HARDWARE-BASED XML SECURITY

SUMMARY

XML Security functions such as XML Signatures, Verification, Encryption and Decryption require dedicated hardware for three significant reasons:

- *Hardened security* — Security functions require keying information that needs to be secured. A private key is considered to be the 'crown jewel' of the company and protecting it should be the company's foremost priority, i.e., all vendors should certify that they are using hardware-based RSA key acceleration.

- *Manageability* — Centralized XML Security management provides significant reduction in policy management and serves as a good security policy by separating application development from security enforcement.

- *Price/Performance* — 2048-bit key RSA operations for XML Security running on general-purpose processors are significantly slower than dedicated cryptographic chips. Even when response times may initially be acceptable with software-based solutions, with increasing transaction throughput (TPS) requirements, the price/performance advantages of crypto chips over general-purpose processors becomes evident.

HARDENED SECURITY

This section provides a brief description of the issues that should be considered by IT specialists when evaluating the use of security products to protect their XML applications. In addition to analyzing the features of a security product that protects the respective XML application, an IT specialist should verify the security of those features before selecting a security product. If the implementation of the XML security product is weak, then the entire application is compromised, irrespective of the additional benefits offered with the product. This is because the XML application protected by a weak security mechanism can give a false sense of security, which is riskier than being aware of the deployment of an insecure XML application.

Key-Finding Threat

Key-finding describes a threat by which an unauthorized user can find the private key used in a cryptographic security scheme. Once the key is found, the unauthorized user can impersonate the legitimate user in electronic transactions.

The key-finding threat occurs in the following way: Typically, in a commercial Web server, the key is encrypted and stored within the server where it must be decrypted before it can be used. If a web server is compromised as a result of an attack, then an attacker can read the memory and can retrieve the private key.[1]

Security managers face internal attackers, such as disgruntled employees, and outsiders, such as hackers. The internal attacker is the more common and more difficult to defend against since an internal attacker can gain legitimate access to a web server and extract private keys. This internal attacker, perhaps a disgruntled employee, would then use the private keys to wreak havoc on the company. Some reports suggest that up to 80% of security breaches are carried out by employees. As a result, the key-finding threat will most likely come from an internal source.

Both internal and external key-finding attacks can be prevented by using hardware-based key storage where the keys are stored separately from the application address space. This prevents an attacker from ever accessing the key in its unencrypted form. In the case of a disgruntled employee, s/he will be prevented from exporting the private keys to her/his computer if proper security practices are adopted by the company when keys are exported. The company should enforce key splitting when private keys are exported so two or more users share the key where no one user has access to the full key. This security practice distributes the trust exposure and can be achieved by hardware-based key storage that supports key splitting technologies.

[1] nCipher: Protecting Commercial Secure Web Servers From Key-Finding Threats

Buffer Overflow Exploits

The majority of security alerts are attributed to buffer overflow attacks against networking servers.[2] This is one of the most devastating exploits. Even the widely used OpenSSL security library suffered a buffer overflow exploit.[3]

> *"It is an embarrassment for the IT industry that we need a section with this title. Buffer overflow problems have been known for 40 years. Perfectly good solutions to avoid them have been available for the same amount of time. Some of the earliest higher-level programming languages, such as Algol 60, completely solved the problem by introducing mandatory array bounds checking. Even so, buffer overflows cause about half of the security problems of the Internet. And still people refuse to banish them by using better tools. We consider this criminal negligence. It is comparable to a car manufacturer making a gas tank out of waxed paper..."* -- Neils Ferguson and Bruce Schneier

Hardware-based XML security devices are not immune to buffer overflow attacks. Hardware XML device vendors should be thoroughly investigated on their implementation tools to insure that they do not suffer similar exploits. All the hardware-based XML solutions have an http stack and cryptographic software, which could be exposed to the same problems. A company evaluating hardware-based XML security device vendors should ensure that by introducing a device in their network, they are not introducing more security risks. Unfortunately, this topic is swept under the rug by most vendors. A company that is evaluating a security solution for XML must insist on learning from vendors how their implementation is protected from buffer overflow exploits.

Weakness of Software Cryptography

Software-Based PRNG Risk:

XML applications performing security functions in software, such as cryptographic operations, are vulnerable because key generation draws on a software-based PRNG (Pseudo Random Number Generator); XML encryption is especially susceptible to weak key generation. Irrespective of the strength of the cryptographic algorithm or the key size, a weak PRNG can compromise the security of the application.[4] In 1995, a practical research attack by graduate students from the University of California, Berkeley demonstrated how easily Netscape's Browser security could be compromised, which was due to the poor seeding of the PRNG.[5]

XML security keys, which are generated on dedicated hardware, provide stronger security because of the superior entropy-gathering mechanism for seeding the Random Generator that the keys are based on.[6] This supplies the optimum key material for strong encryption and signatures.

Another approach for guarding against a weak PRNG implementation in an XML security product is to apply the FIPS (Federal Information Processing Standard) process. This process will require Random Number Generator self-tests to assess the strength of the PRNG.[7]

---

[2] http://securityresponse.symantec.com/avcenter/security/Advisories.html

[3] http://www.counterpane.com/alert-v20020731001.html

4 http://www.counterpane.com/real-world-security-ft.txt

5 http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html

6 http://www.intel.com/design/security/rng/rngfaq.htm#link_2

7 http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

Extracting Private Keys Through Timing Attacks

Recent research has demonstrated that performing RSA private key operations in software exposes the secret of the private key on a web server. In a 2003 article, David Brumley and Dan Boneh of Stanford University established the practicality of extracting private keys from web servers through timing attacks.[8] Many cryptographic software libraries do not defend against timing attacks and are, therefore, highly at risk. In addition, Brumley and Boneh further stated that hardware-based accelerators do not experience this attack and are, therefore, immune from this type of attack. With regards to this vulnerability, a strict policy to enforce hardware-based acceleration should be implemented.

A private key is considered to be the 'crown jewel' of the company and protecting it should be the company's foremost priority, i.e. all vendors should certify that they are using hardware-based RSA key acceleration.

Introducing Security Holes

Running software applications linked with security software on the same internal host server can introduce several challenges:

- By running the security software on the same host as the software application, one exposes the security software to hacks because the host might not be locked down where all TCP/UDP ports are open.
- File sharing on a typical host might be enabled, which would allow the attacker to access the machine.

Separating the security software from the application software allows the application developer to focus on improving the quality of the application rather then seek short cuts in linking in with the security software on the same host. By separating the two functions, one can easily narrow down the problem area.

MANAGEABILITY

Separation of XML Security functions from general application development ensures stricter security. Another significant advantage of this separation is reducing the cost and complexity of managing XML security policies. There are two significant impacts of dedicated XML Security Hardware:

- Rapid, non-programmatic addition of XML security policies. The number of policies increases with trading partners and types of XML messages exchanged. With toolkit-based XML solutions, adding such policies would require significant effort over dedicated hardware-based solutions.
- Centralized XML security policy management, separated from application severs enables policy updates to be performed at a central location instead of on individual application servers. The task of updating such security policies can be daunting in data centers with large application server farms. Considerable savings in centralized certificates on XML Security hardware, instead of certificates across multiple application servers, offers further incentive for a centralized approach.

Binding security policies to applications can inadvertently open security holes. Security concerns should be kept separate from application concerns.

PERFORMANCE

XML Security functionality is available in toolkits and software that can run on servers with general-purpose processors. For small key sizes (512-bit), general-purpose keys can yield acceptable price/performance compared to crypto chips. However, with increasing key-sizes, computational requirements of private key operations on general-purpose processors increase exponentially. For 2048-bit keys typically used for XML security operations, general-purpose processors price/performance is significantly higher than crypto chips, especially for 2048 bit private key operations. 2048-bit RSA private key operations take 30 milliseconds on a 2.8GHz CPU compared to 4 milliseconds on a specialized crypto-chip.

With the increasing number of XML Security transactions (throughput), maintaining acceptable response times requires additional computational power. Adding general-purpose processors to maintain response time thresholds with increasing TPS requirements is not scalable and is exponentially cost prohibitive, especially for 2048-bit private key RSA operations. Dedicated XML Security Hardware provides the price/performance advantage over powerful multi-processor servers for such security operations.

[8] Boneh, Dan and David Brumley. Remote timing Attacks are Practical. (2003)