CRYPTOGRAPHY INVENTORY

BUILDING, MAINTAINING AND EXPLOITING A USEFUL INVENTORY

Whitepaper v1.0

February 2020

cryptosense

Contents

Cryptosense SA, 231 Rue Saint-Honoré, 75001 Paris France

cryptosense.com

1. What is a Cryptography Inventory

A cryptographic inventory is a strategic cybersecurity asset. It enables an organisation to:

» enforce a secure cryptographic policy across IT infrastructure;

» react quickly to security issues;

» efficiently carry out strategic transformations, such as migrating cryptography services to the cloud, or deploying post-quantum cryptography.

While some more sensitive organisations have maintained cryptography inventories for years, the concept has recently become more mainstream thanks to recommendations from standards bodies [1] and industry analysts [2], and also due to high profile incidents where more careful cryptography management would have prevented or minimized losses [3]. At the same time, organisations with existing inventories are looking to improve their coverage and automation to meet new business goals.

At its most basic, a cryptography inventory can be thought of as a map of all the cryptography deployed in an organisation. The inventory will describe what cryptography is used by which applications for what purpose, as well as mapping cryptography use in infrastructure. It can include details of algorithms, usage modes, keys and key storage, certificates, protocol versions, library versions, as well as non-technical details such as data classification, business purpose, and the name of the person responsible.

At Cryptosense, we have followed several of our customers as they create or upgrade their cryptography inventory. We have seen that in practice, every organization's cryptography inventory will be a little different, depending on how and where cryptography is used relative to the business-critical functions of the organization. Indeed, successful cryptography inventory projects typically apply a different methodology to different parts of the application estate or infrastructure, in order to prioritize resources to the most important areas. In this whitepaper, we will discuss the lessons learned from these projects and the best practice approaches that will enable you to guide your own cryptography inventory project.

2. Why Maintain a Cryptography Inventory

There are multiple business reasons to create and maintain a cryptographic inventory.

2.1. Security and Resilience

At a high level, cryptography is a risk control like any other. "Crypto risk" is the possibility of an organisation suffering harm when cryptography doesn't work as it should. There have been several high profile examples of cryptography failures in the last few years, such as the $150M+ Capital One breach where the use of cryptography failed to protect personal data of millions of users due to a key management issue, or the $200M+ Marriott Hotel breach where partial use of encryption lead to millions of passport numbers being leaked in the clear (details of these failures and more are available in a recent Cryptosense Webinar [3]). Failures in cryptography have also caused availability outages, such as when Erikson failed to update a certificate in a mobile phone network management product, causing hours of downtime for millions of users [6].

A comprehensive and up-to-date cryptography inventory can help to reduce cryptography risk by raising the visibility of cryptography use and allowing policy control. In order to do so, it will have to include sufficient detail to allow issues to be detected and diagnosed. Attacks on cryptography almost never exploit a weak choice of algorithm. Instead, they "go around" the cryptography by exploiting key-management mistakes, padding oracles, malleable encryption, repeated nonce values etc. For a cryptography inventory to be effective in preventing these attacks, it needs to be sufficiently detailed to control the cryptography policies that prevent these issues: tracking key usage, use of authentication on encryption, etc.. We will discuss this further below.

2.2. Compliance

Partially in response to the incidents described above, auditors are getting more and more particular about controlling that cryptography is being used correctly, whereas in the past they were often satisfied just to know that cryptography was in use. Auditors under various regimes now expect to see a cryptography policy and evidence that the policy is controlled.

For example, the PCI-DSS standard gives the requirement that cardholder data must be protected by:

> *"Cryptography based on industry-tested and accepted algorithms, along with strong key lengths (minimum 112-bits of effective key strength) and proper key-management practices",*

referring to NIST publication SP 800-57 for more detail, while the Gramm-Leach-Bliley Act (GLBA) for US financial services customer data protection requires:

> *"The encryption of electronic customer information, including while in transit or in storage on networks or systems, in case unauthorized individuals are able to gain access. The selection of data to encrypt and the encryption technique and level should be supported by the risk assessment."*

Increasingly, auditors are asking for evidence that these requirements are being controlled across applications and infrastructure.

An up-to-date cryptographic inventory can greatly reduce the compliance burden around these kinds of requirements, although it must be up-to-date and sufficiently detailed to show not just what cryptography is in use, but also what data it is protecting and how the keys are managed.

2.3. Crypto Agility

The concept of crypto agility has risen up the agenda recently, this is primarily due to two considerations:

» firstly, the painful experience that many organisations went through to eliminate the use of the broken SHA-1 hash function in the late 2010s,

» second, the progress towards practical quantum computers that will be able to break the asymmetric (public key) cryptography widely used across all of our systems.

Wikipedia defines crypto agility as allowing:

> *"an information security system to switch to alternative cryptographic primitives and algorithms without making significant changes to the system's infrastructure."*

The recommended first step towards achieving crypto agility is knowing where your cryptography is, i.e. having a cryptography inventory [2,4].

As well as setting out a map for migrating algorithms when required, a good cryptography inventory identifies areas for work to remove obstacles to agility, for example hard-coded keys, key-lengths, even hardcoded keys.

2.4. Cloud Migration

Migrating applications that handle sensitive data to the public cloud requires careful consideration of cryptography. First, the existing cryptography used in the application likely either uses keys stored in a physical hardware device like a Hardware Security Module (HSM) that will not be present in the cloud data centre, or stored on the application filesystem in a way that will not be considered adequately secure for a cloud deployment. These keys will need to be migrated to cloud cryptography services (Key Management Services or Cloud HSMs). Infrastructure keys present on the application's filesystem, such as SSH keys or TLS certificates, will need migrating or replacing. Finally, extra encryption may need to be added to data that was passed or stored unencrypted in the on-prem data centre.

A cryptography inventory can help with this process by allowing the identification of cryptographic keys that need to be migrated and highlighting what data is currently protected by encryption. To be useful, it needs to include information about how keys are actually used, to allow the selection of the right cloud cryptography service.

3. What should be in a Cryptography Inventory

Often, the business reason for conducting cryptography inventory will be some combination of the reasons above. A cryptography inventory must cover all relevant artifacts to achieve its purpose. Typically this will include an inventory of algorithms and keys as they are used in applications and infrastructure. The exact details will depend on precise inventory goals. For example, if the goal is to prepare for crypto agility, the inventory should distinguish between algorithms used for cryptographic and non-cryptographic purposes, so as to allow for planning and decision making during a migration [2]. If the goal is to control a cryptographic policy, then the inventory must be as detailed as the policy. For example, if the policy allows use of an algorithm only in the context of a specific protocol, then the inventory must include protocols. If the policy specifies that only specific cryptography libraries be used, then the inventory must include libraries and library versions.

3.1. Application Cryptography

In a typical enterprise, applications use cryptography to:

» encrypt sensitive data at rest or in transit,

» sign or verify documents and files,

» protect passwords and credentials and access sensitive resources,

» participate in other protocols such as single sign-on, TLS, Kerberos, etc.,

» and often, to perform business-specific cryptography such as authorising payments.

All of this cryptography will involve particular algorithms and keys, modes of operation, protocol versions, and cryptographic libraries. Keys and certificates will be accessed by the application from a variety of sources such as filesystem keystores and HSMs. All of this information must be captured in an effective cryptographic inventory.

3.2. Infrastructure Cryptography

Enterprise infrastructure typically makes extensive use of network cryptography protocols such as TLS, IPSec and SSH. Each of these requires an infrastructure of keys and certificates that must be included in a cryptographic inventory to meet the goals discussed above.

Items of hardware including Hardware Security Modules, VPN appliances, firewalls, and intrusion detection systems will all contain cryptographic artifacts and capabilities to use specific algorithms. Often these systems control the most sensitive cryptography in the organisation, i.e. the keys used to protect the most business critical data. As such, their inclusion in a cryptographic inventory is an important consideration.

4. How to make a Cryptography Inventory

To create a cryptography inventory, multiple challenges must be addressed:

Scale - organisations may need to inventory thousands of applications and tens or hundreds of thousands of endpoints. This requires automation using tools that scale, and procedures that can be distributed across the organisation. It also has a bearing on the data model to be used for the results. For larger organisations, a hierarchical model that can be queried is more appropriate than a flat spreadsheet that will quickly become unusably complex.

Access - whether for the deployment of cryptography inventory tools, or manual methods such as code review, inventory construction requires cooperation from multiple stakeholders in the organisation. Access may be needed to code, running applications in test environments, filesystems, containers in the deployment pipeline, Cloud VMs, network endpoints, etc.. A good inventory plan will consider not just the “what” but the “who” and aim to use time efficiently, for example by testing out the inventory procedure for an application on a small cohort before pushing it out to the whole estate.

Precision - false positives and false negatives drastically reduce the value of the inventory, no matter what the inventory goals. They can lead to wasted time on unnecessary remediation work, undetected crypto risk or non-compliance, and misplaced prioritisation for migration or agility projects. Therefore any decisions to save on inventory resources, such as asking a third-party vendor to self-report their product’s cryptography inventory, or using scanning tools rather than runtime tooling to test applications, must be weighed up against the loss of precision in the inventory and its effect on the business goals.

Prioritisation - not everything can be done at once, so an appropriate roadmap, with short time-to-value to ensure buy-in, and realistic milestones towards completion must be identified. Due to resource constraints, typically cryptography inventories are not uniform in their coverage of IT assets. For example, external-facing applications treating customer data will typically require full, precise coverage, whereas internal applications treating non-sensitive data may only require a more basic scan, where some risk of imprecision can be accepted. This should be reflected in the inventory project roadmap.

In the rest of this section, we examine these challenges as applied to approaches for taking cryptography inventory in applications and infrastructure.

4.1. Application Cryptography

There are a number of possible approaches to discovering cryptography in applications.

Manual - cryptography can be discovered manually either by reviewing code or by interviewing developers or suppliers. While this might look like the most lightweight approach, in practice this is often very time consuming and can quickly become adversarial, as the inventory-taker tries to obtain precision, while the application owner or vendor pushes back in order to return to other priorities. One Cryptosense customer described the process of taking application cryptography inventory by hand as “hand-to-hand combat”. Typically, results are disappointing in terms of precision. Another Cryptosense customer found that some vendors, when asked to report on algorithms used inside applications, in fact reported on the TLS ciphersuites that were available on the network layer wrapping the application. Without a secondary control, it is hard to see where these imprecisions are.

Static scanning - static scan tools take either source code, bytecode or binary as input. They are typically used to check for bugs, vulnerabilities, or non-compliant coding, but can also be used to check for cryptographic vulnerabilities. However, they tend to be much less effective at detecting cryptographic errors than other kinds of vulnerabilities, such as buffer overflows.

To understand why, we need to understand the difference between these vulnerability types. Buffer overflows are typically local issues: data is read, and then written to an area of memory, and either there is some nearby range check on the size of the data, or there is not. Since the whole pattern can be checked locally, precision is not affected by the logical over-approximation of the control flow graph (i.e., the best guess at what will happen in the code at run-time that the static tool must make, since it will not actually execute the code).

Cryptographic vulnerabilities often depend on the very precise choice of parameters that are supplied in calls to cryptographic libraries, which may depend on non-local inputs such as configuration files. Moreover, whether a call is secure or not may depend on other parts of the code which are in completely different parts of the application. For example, whether this key has already been used for a different operation, or whether this nonce value has already been used with the same key. These issues are extremely hard, if not impossible to detect with static tools. Hence if reduction of crypto risk is a main goal of the inventory, these tools are not suitable.

Some static tools also attempt to give details of all the cryptographic algorithms that are detected in an application. However, they suffer from the same limitations since the execution path will govern whether certain cryptography is actually used or not. For example, many applications include a cryptography library that includes insecure algorithms such as MD5 or DES for legacy reasons, but the application may never call them. Middleware components or web frameworks used by the application may include the capability to call these algorithms, but only do so if certain configuration options are set. The application may use a function like MD5 for a non-cryptographic purpose such as creating an identifier on data controlled by the application, for which attacks on MD5 are not relevant. Finally, if cryptographic libraries are loaded dynamically and algorithm identifiers are created by reflection from strings in a run-time configuration file, the static scanner may have no visibility on what is used. Hence, while knowing what algorithms are inside an application can be useful as a very first step, it tends to produce inaccurate inventories that risk not meeting business objectives.

Run-time tracing - run time tracing or IAST (Instrumented Application Security Testing) tools work with the application at run-time, typically in a test environment but sometimes in production, by "hooking" certain parts of the application to observe behaviour. The advantage of this approach is that these tools have visibility on all cryptography that is used, including calls from third-party libraries and framework components, without false positives, since only cryptographic calls that actually happen will be traced. The only disadvantage is access requirements: the tools only work on running applications, and those applications must be exercised to use all their cryptography, either by unit or integration tests, interactively, or by tracing a production server.

At Cryptosense, we combine both IAST and static approaches. Cryptosense Analyzer is the state-of-the-art IAST tool for cryptography, with coverage for Java JCE, .NET cryptography, PKCS#11 and OpenSSL, giving not just inventory of algorithms, keys, key-lengths, libraries, modes of operations, passwords, and parameters, but also full vulnerability analysis and customizable cryptographic policy enforcement. Cryptosense Analyzer provides full usage information for all the cryptographic keys the application employs, including details of key storage, operations, code locations and data protected. Cryptosense Analyzer integrates in CI pipelines thanks to its REST and GraphQL APIs and plugins for common build and CI tools such as Maven, Gradle and Jenkins, making sure cryptography inventory is always up-to-date, and allowing the inventory update workflow to become an automated, background task.

Cryptosense also offers a Static Crypto Scanner that detects cryptographic operations and algorithms in bytecode. This can be used in combination with the IAST tool, in which case results are processed together to give coverage information and detection of hardcoded keys.

For applications where, for whatever reason (non-sensitive data, imminent replacement, difficulty in running tests, internal use only) a IAST cryptography inventory is considered inappropriate, the static cryptography scanner can be used as a standalone tool for basic cryptography detection.

4.2. Infrastructure Cryptography

Modern IT infrastructure uses cryptography extensively, principally to encrypt data in transit (protocols like TLS, IPSec) and data at rest (encrypted storage), and to enforce access control (SSH keys, LDAP/PKI etc.). Endpoints in the network, whether they are devices, employee terminals, application servers or specific appliances, will include some cryptographic capability in order to interface with the rest of the infrastructure. The security of this cryptography is just as important as the security of application cryptography, and so needs to be included in the cryptography inventory.

4.2.1. Network Cryptography

Most network cryptography uses a small number of widely-used protocols.

TLS

TLS is a client-server protocol originally developed by Netscape in the mid 1990s and now an IETF standard. It was originally designed to secure electronic commerce between web browsers and servers on the Internet, but is widely used as a general purpose network encryption protocol. It relies on a public key infrastructure (PKI) for authentication, which can be mutual when client certificates are used.

There have been several versions of TLS, some of which are now considered insecure. Older versions of TLS tended to have many configuration options that set the handshake and channel encryption ciphersuite as well as a range of additional options, some combinations of which are also considered insecure. Managing secure TLS configuration at scale is a challenge, and there are several tools available to help with this, such as proprietary dynamic application scanning tools (DAST) like Qualys and Tenable, which can give details of TLS cryptography as one of their outputs, and open-source tools such as `testssl.sh`, which can be scripted at the command-line. However, many organisations use these tools on a limited subset of their network, such as external-facing TLS servers only, to save time and resources. For a full cryptography inventory, internal TLS must also be scanned, which may require different tooling.

Cryptosense's offering in this space is our Network Crypto Analyzer which can be tested online at https://discovery.cryptosense.com. It includes a number of unique features which help cryptography inventory to be more thorough and efficient: first, it interfaces with the certificate transparency log to discover relevant registered domains that may have escaped attention or been forgotten. Second, it suggests remediations for configuration issues for common webservers. Third, at the users option, it can scan not just the given set of servers but also (when they serve http webpages) their dependencies, i.e. the servers that provide the critical javascript and other components that modern websites rely on. Additionally, as well as providing compliance information with respect to a user's policy, our Network Crypto Analyzer provides attack trees that explain how combinations of vulnerabilities could be exploited to compromise encrypted data, allowing appropriate prioritisation of fixes. Automated scans can be programmed, and results exported and cross-referenced to applications in the Cryptosense Analyzer Platform.

SSH

The Secure Shell Protocol was initially designed to allow secure remote login access, but much like TLS, it has now become a general network security protocol. Much like TLS, SSH servers have a number of configuration options, some of which are now considered insecure, although configuration complexity is somewhat lower than for TLS, in particular because enterprises will typically control both the server and the client configuration, and not have to worry about connecting to customers with legacy browser versions etc.

Tools for testing SSH configurations include open source tools such as `SSHscan` (for obtaining a list of ciphersuites) and `ssh-keyscan` (for obtaining host keys). The security of SSH also depends on the authorized keys that can be used to access servers, but these cannot in practice be obtained by network scans. A filesystem scan is necessary (see below).

Cryptosense Network Cryptography Analyzer includes SSH ciphersuite and key coverage, and checks that ciphersuite configuration meets policy. Network scans can be cross-referenced with filesystem scans (see below).

IPsec

IPsec is frequently used to manage virtual private networks (VPNs). IPsec cipher configuration can be complicated (see for example the list of ciphersuite options for the StrongSwan software implementation of IPsec [5]). IPsec management often involves hardware appliances with proprietary configuration mechanisms, and to add to the difficulty, different manufacturers refer to the same ciphers by different names. There are open source tools for scanning IPsec over the network, for example `ike-scan`, but these require some manual configuration to enumerate acceptable ciphersuites. Additionally, VPN devices can be quite sensitive to failed connection attempts, which are necessary to determine acceptable configurations. Full testing usually requires examination of the config files.

Other Protocols

Some new protocols are starting to find their way into enterprises, such as the Wireguard VPN protocol that can replace IPsec and is embedded in modern Linux Kernels, and domain-specific network cryptography protocols for example for IoT. Where network scanning tools are not available, these will have to be inventoried manually from their configuration files.

4.2.2. Cryptography Artifacts on the Filesystem

As we saw above, some aspects of infrastructure cryptography can't be inventoried by network scans alone. To complete the inventory, we will have to scan filesystems and retrieve cryptographic artifacts.

Certificates

Public key certificates that manage PKI may be used for TLS or other application specific protocols. They are typically stored on filesystems in standardised formats, or in platform specific filestructures such as a Java keystore or the Windows certificate store. Each certificate contains a variety of information pertinent to a cryptography inventory, such as the public key and signature algorithm, the hash function used to produce a digest of the signature for verification in the PKI, the validity dates, common name etc. When a private key is present, it is usually encrypted by some password-based encryption method. The security of this encryption depends on the password, the password based key derivation function parameters, and the cipher used, all of which we may want to include in a complete cryptography inventory.

Certificates can be added to the inventory from filesystem scans, application scans, and network scans. Cryptosense Analyzer Platform includes a filesystem scanner for cryptographic artifacts including certificates, and allows these scans to be cross referenced to identify where the same certificate has been seen and used. Some enterprises may already be using in-house or third-party certificate management tools. Sometimes, these tools support export of their certificate list, allowing it to be added to the inventory.

SSH keys

Since SSH keys have their origins in a proprietary format, and don't form part of a PKI signature hierarchy, they are typically somewhat separate from other standardised certificate keys. However, an SSH infrastructure includes an implicit hierarchy, since an SSH server will typically include a list of authorized keys which that server will permit to be used for a login. Depending on inventory goals, this information might be pertinent to include in the inventory. It may already be managed elsewhere by an in-house or third-party tool. Other information that can be picked up from a filesystem scan includes the algorithm, keylength, and in the case of private keys, the password-based encryption mechanism used to protect the private key value. Cryptosense Analyzer Filesystem Scanner includes coverage for SSH keys.

4.2.3. HSM keys

Hardware Security Modules (HSMs) often contain an organisation's most important keys in terms of business value. These can include master keys like root signing keys for the PKI and domain-specific keys for payment processing and other critical functions. These keys, and the HSMs cryptographic capabilities, are therefore an important component of cryptographic inventory for all business objectives.

For HSMs that support the PKCS#11 standard interface, Cryptosense Analyzer Platform has extensive tool support for evaluating HSM security, scanning key information, and tracing use of the HSM in order to identify which the algorithms and parameters used.

4.2.4. Other keys

There are typically other keys lurking on filesystems than these standardised ones. Some applications store keys in their own proprietary ways, often insecurely. The only way to discover these keys is to trace the applications and examine the key lifecycle information. In Cryptosense Analyzer Platform, you can see all keys used by an application and filter out those which have been loaded from standard keystores, leaving only those for which the storage requires further investigation.

5. Planning an Inventory Project

We have surveyed the kind of information that might be appropriate for inclusion in a cryptographic inventory, and the modalities for obtaining this information from applications, networks, filesystems and hardware devices. The first step for an inventory project is to decide what information is to be included to meet the business objectives for that particular organisation. A typical experience of Cryptosense customers is that the inventory reveals far more cryptography in use that was initially expected. Hence the the inventory project must be designed to handle large volumes of information and make is accessible in ways that meet the business objectives.

The inventory can then be broken down into packages of network, application, filesystem and hardware scans, and appropriate procedures and tools selected for each. At Cryptosense we have seen that inventory projects tend to succeed when they demonstrate value early to other stakeholders in the organisation. Hence, starting with the most critical applications in the organisation is frequently an effective strategy, before moving on to fill in the rest of the inventory on less sensitive parts of the estate.

5.1. Five Questions to ask of Cryptography Inventory Tools

Cryptography inventory tooling needs to meet certain criteria to give the project a strong chance of success.

1. Continuous visibility or one-time?

Good cryptography inventory tools will support automation not just for a one-time scan, but also for continuous updates, so that the inventory is always up-to-date, and changes and improvements can be tracked over time. For example, tools for application cryptography inventory should integrate into the CI toolchain and report in to the central inventory.

2. Does the precision meet the business goals of the inventory?

An inventory of cryptography is not an end in itself. It must be accurate and precise enough to meet the business goals of reduced risk, cryptography agility etc. Typically this requires the false positive and negative rate to be very low, and the detail to include not just algorithms but also details of modes of operation, padding, parameters etc., both in application and infrastructure cryptography.

3. Is there support for open input and output of data?

Many organisations will already have some partial support for cryptography inventory, for example from certificate managers, SSH managers, TLS scanners, endpoint management tools etc. A good inventory tool will support input of inventory information from other tools. Additionally, the organisation may already have a preferred way to consume this kind of data, for example in Splunk or a similar visualization tool, or in ServiceNow-type inventory management tools. A good inventory tool will export information in a manner suitable for these tools.

4. Is the final inventory queryable?

In order for the inventory to be useful, we will need to be able to query it to answer questions like “where are we using SHA-1 outside of a random number generation context?” or “where do we have certificates with a lifespan of more than 4 years?”. This requires a data model that supports querying, not just a flat spreadsheet.

5. Can applications and infrastructure be cross-referenced?

The inventory as a whole must be queryable, not just individual scans. For example, it should be possible to ask the tool to check whether a certificate found in a filesystem scan has also been seen in a network scan or in use by an application, and if so where.

Cryptosense Analyzer Platform has been explicitly designed to meet these goals. Cryptosense is pleased to advise and provide tools for all aspects of cryptography inventory projects. Contact us at inventory@cryptosense.com.

6. References

[1] NIST Special Publication SP 800-57 - Key management recommendations https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf

[2] Gartner, Better Safe Than Sorry: Preparing for Crypto-Agility
Published: 30 March 2017 https://www.gartner.com/en/documents/3645384

[3] Cryptosense Webinar, The Impact of Crypto Failures, https://cryptosense.com/webinars/watch-the-impact-of-cryptography-failures-webinar-part-1/

[4] Accenture, Cryptography in a Post-Quantum World https://www.accenture.com/_acnmedia/pdf-87/accenture-809668-quantum-cryptography-whitepaper-v05.pdf

[5] Strong Swan IPSec IKEv2 Cipher Suites Configuration https://wiki.strongswan.org/projects/strongswan/wiki/IKEv2CipherSuites

[6] "Why millions of Brits' mobile phones were knackered on Thursday: An expired Ericsson software certificate" https://www.theregister.co.uk/2018/12/06/ericsson_o2_telefonica_uk_outage/