DetecTor

Sonntag 15 September 2013

DetecTor

http://DetecTor.io

Document version 1, published 2013-09-15

Kai Engert - kaie@kuix.de or kaie@redhat.com

Please use mailing list https://www.ietf.org/mailman/listinfo/therightkey to discuss this project's applicability for improving the public Internet TLS/PKI security infrastructure.

Scope

DetecTor is an open source project to implement client side SSL/TLS MITM detection, compromised CA detection and server impersonation detection, by making use of the Tor network.

It will be implemented by extending the NSS security and cryptopgrahy library, to allow existing applications (that already use NSS) to benefit from DetecTor capabilities, by relinking against the DetecTor enhanced library, without requiring application level changes.

The list of applications that could be easily extended include the Mozilla Firefox browser, the Chromium browser, the Mozilla Thunderbird E-Mail client, the Gnome Evolution E-Mail client and the Pidgin instant messaging software. Alternatively the suggested enhancements could be integrated into NSS directly. Optionally, should this project be successful, this technology could also be integrated into additional open source cryptopgraphic toolkits, such as OpenSSL or GnuTLS.

Limitations: The technology presented here doesn't help if a a server's private key has been compromised. Obviously, it also doesn't help when using unprotected connections, without SSL/TLS at all. Also, if a cluster of servers uses multiple different certificates for a single hostname, the presented solution

is probably of limited help. However, best practices for server configuration will be suggested, to achieve compatibility. In the common scenario where a server (or a cluster of servers) uses the same certificate globally, and deploys certificate changeovers within a reasonably short period of time, the presented solution should help.

Other projects have already proposed potential solutions for MITM and hacked CA detection. For example, the Perspectives and Convergence projects propose to query designated network nodes that act as notaries. This doesn't completely solve the problem, because it still requires to trust someone (the notary operators), as the alternative (operating own notary infrastucture that run the software proposed by those projects) is impossible for most users to deploy. The MECAI system that I had proposed requires too many changes to the infrastructure, and is probably impractical even as a mid term solution. The Certificate Transparency (CT) project has been started to detect falsely issued certificates, but it has properties that might prevent it from being effective. It requires that CAs cooperate by publishing data (which might not happen in case of a compromised CA). In order to detect false certificates using CT, it's necessary to constantly scan for changes, which might be impractical for regular user systems.

This paper proposes DetecTor as another alternative, without requiring cooperation from CAs for rogue certificate detection, and which could work even on small systems with limited resources.

DetecTor Description

This project makes use of the Tor network . The Tor network is a global network of nodes running special routing software, that allows anyone to route network connections through Tor. This effectly allows to use a remote location as the starting point for connections to Internet servers. The original intention of the Tor network is to provide anonymity, often by combining the Tor network with privacy enhanced browser software, which attempts to hide the user's identity. However, for the purposes of the DetecTor project, we won't be using Tor for anonymization purposes, but we'll be using the Tor router component, only.

The DetecTor project reuses the notary idea, but it proposes to enable everyone to act as their own notary, by making use of the existing global and distributed Tor network, which is operated by many volunteers, and which already implements consistency checks for network manipulation.

To understand how it works, let's look at an example. A user U wants to use the browser to open the web page of the webmail site M. If the browser supports the integrated DetecTor capability, the browser would perform it's own network analysis prior to connecting to M. The browser would initiate a connection through the Tor network to M, for the sole purpose to check which certificate is being used by M, when contacted from a different network location (the location

of a Tor exit node), but without actually receiving any content from M, because the connection will be ended as soon as M's server certificate has been retrieved.

This concept has already been described in the Doublecheck paper , which had proposed to use the information obtained from the remote network location for Trust-On-First-Use purposes.

DetecTor extends the ideas presented by doublecheck.

- (A) One remote network location isn't sufficient, because one random exit node could also have been manipulated by an adversary. To reduce the likelihood of being manipulated by an adversary, the DetecTor software will use five separate connections to the Tor network, where each connection requests specifc rules for the selection of ExitNodes. The Tor network software allows to request that only exit nodes in a given set of countries are used. DetecTor proposes to divide the world into five spheres, where each country belongs to one sphere. The number five is an arbitrary odd number, a reasonable middle ground between the desire to gather multiple data points, and the desire to limit the amount of network activity. The composition of the spheres can be user defined. For example, the spheres could be defined by contintents, to enable observations based on geographical location. Alternatively spheres could be defined based on the concept of Cultural Spheres of influence , where the countries within each sphere are more likely to collaborate for the purposes of surveillance or jurisdiction, but countries belonging to different spheres are less likely to collaborate for network manipulation purposes.

- (B) While doublecheck used a browser extension, the DetecTor functionality should rather be built into the general purpose SSL/TLS protocol and certificate validation implementation, to enable multiple applications to easily benefit from the work. Also, by implementing at this software layer, it allows for additional flexibility during certificate validation. For example, should the evidence gathered through the Tor network be ambiguous, it could be automatically decided that additional proof of certificate status becomes mandatory, such as the availability of very fresh OCSP information.

- (C) Don't settle on Trust-On-First-Use (TOFU). While it's helpful to know that a given server certificate has been used by a majority of servers at some time in the past, the certificate could have been compromised in the meantime. Or worse, the previously obtained information could have been observed during a global DNS attack to the server, and the remembered certificate is in fact controlled by an adversary. Therefore it still makes sense to query the Tor network for a global consensus of certificate use. While the knowledge gained through the Tor network could be stored in a local database (maintained by the DetecTor software library) for caching and short-term speed optimization purposes, it should

reguarly be reverified. Only if the Tor network is temporarily unavailable, the previously gained information could be treated as sufficient, if the cached information is reasonably fresh, and if fresh OCSP information is available. However, this behaviour is optional and should be a configuration option. Security conscious users should decide that current status information obtained through the Tor network is always mandatory, and that connections should always be rejected, if it hasn't been possible to contact the Tor network for very recently.

- (D) The DetecTor project suggests to use an implementation that works without any user interface confirmation or interaction. If no consistent network view can be established, the connection will be rejected and the user will receive an error message in the application. The user can retry later after the Tor network becomes available again. If detailed user feedback is desired, the DetecTor library could broadcast status information (limited to user's local system, e.g. using the D-Bus technology) for failures and rejected connections, which a separate DetecTor status application could display to the local user.

- (E) By integrating directly into the SSL/TLS network protocol implementation code, increased compatibility with servers can be achieved. A single SSL/TLS server may use multiple different certificates, based on properties of the handshake at the start of connections. For example, the server could select the presented certificate based on the cipher suites supported by the client software, or based on the use (or non-use) of the server-name-indication extension. As a result, different client software could see different certificates. Because of the direct integration, each DetecTor connection for notary purposes can use identical connection properties, which will result in retrieval of the same server certificate with an increased likelihood. This might work better than querying external notary servers, as proposed by other projects, because the notary servers might have used different connection properties when connecting to the destination server.

The solution described may help to detect, whether a directly connected server uses a different certificate than what can be seen from the majority of remote Tor network locations. This helps to detect a targeted attack against oneself or on the local network. Even if an adversary is able to hack or compell a CA, to obtain an apparently valid certificate, DetecTor can reject it because of inconsistency.

However, the above assumes that the adversary isn't physically close to the server which an adversary attempts to impersonate. The above also assumes that the adversary isn't able to run a global DNS manipulation attack, which would redirect all client requests to servers controlled by the adversary. In both scenarios, client side DetecTor technology is insufficient, because any or most connections would connect to the adversary, regardless of the client's location in the network.

However, a complementary solution is proposed for the server impersonation and network manipulation scenarios, that makes use of the DetecTor technology, too. It should become the responsibility of each server administrator to monitor their own server for such attacks. This can be achieved by regularly connecting to oneself's server from remote network locations through the Tor network, and alerting the administrator if mismatches are detected.

This can enable quick detection of hacked or compelled CAs, and allow a server operator to quickly communicate with the abused CA, for the purpose of immediate revocation of rogue certificates. CAs that don't respond immediately to compromise alerts will become known during such incidents and risk their inclusion status in the default set of trust shipped by client software. This creates pressure on CAs to not allow themselves to be abused for MITM purposes, because detection and non-responsiveness can end their business, and it supports CAs that act exclusively in the interest of security of server operators and users.

While the above fosters detection of CA abuse, we still require CAs to do the verification work. We still must trust someone - until it has been shown that someone isn't trustworthy. However, detection is insufficient. We still need the other properties of PKI for global deployments, and most urgently we require that availability of fresh OCSP information becomes mandatory. Applications and servers should all use OCSP stapling, giving well-behaving CAs the ability to quickly respond to compromises by revoking certificates and clients to learn about revocation.

There are few edge cases where the proposed solution doesn't work yet, but it should be possible to fix that by suggesting best practices for server configuration, and by asking server operators to comply. While most small servers and small clusters will use a single certificate for all connections (per set of client security properties), global deployments exist that use multiple certificates. For example, some global banks use several certificates based on physical network location and based on the data center the client software connects to. This makes it nearly impossible to observe consistency when probing the server. It also causes a problem for server administrators that wish to the DetecTor probing tool to monitor their own infrastructure for global attacks. For these reasons it should become a best practice that even global clusters use a consistent certificate. It needs to be discussed whether tolerance should be added to the DetecTor logic to allow for such deployments. For example, it could be acceptable if all sites use a certificate with identical subject name and are issued by the same CA, and very fresh OCSP data is stapled.

This paper demonstrates another legitimate purpose for the Tor network besides the established anonymity purpose, and it suggests that a larger number of people in additional geographical locations run Tor network nodes and exit nodes. The Tor network used in the described way could improve the security of all Internet users.

The proposed solution doesn't require new server software to be developed or de-

ployed, it asks for wider deployment of existing technologies (OCSP and OCSP stapling) and for wider support for operating Tor network nodes. Although global use of DetecTor would create a major increase of network activity on the Tor network, the bandwidth increase will be limited to the small data exchange that happens at the beginning of secure connections, because no data payload will be transmitted through Tor (except for clients under attack that decide to reroute connections).

Additional ideas

The DetecTor technology could be used to dynamically re-reoute connections. If the client is located in a compromised part of the network, but is able to connect to the Tor network, and the status information obtained using the DetecTor probes suggests that the local connection is compromised, the DetecTor SSL/TLS implementation could dynamically route the connection through the Tor network, thereby accepting the slower Tor network in favor of security.

Using the DetecTor technology could enable an additional increase of security, by broadcasting information about revoked certificates more quickly. Imagine that an adversary could steal a valid certificate, and the adversary controls a subset of the global network. When probing the global network using DetecTor, the use of certificates might be inconsistent, making it difficult to decide which certificate to trust. However, if some of the network connections present a fresh stapled OCSP response, while the direct route is lacking fresh OCSP status, the DetecTor client can decide that a compromise is likely and reject the certificate. It's also imaginable to introduce a new TLS protocol enhancement similar to multi-OCSP-stapling . Servers that aren't controlled by the adversary could include an additional OCSP response, which informs about the recently compromised certificate. The DetecTor client could retrieve it while probing connections to the non-compromised servers, thereby learning about the revoked certificate, and learning that connections using that certificate are compromised - even if the attacker is able to replay a reasonably fresh OCSP status response on those connections, the revoked status would take precedence.

Deployment

A local controller software will be provided that can be used to startup the five Tor spheres, based on a configuration file that can optionally be adjusted by the user. The software will provide five local network ports on the local IP address of the user's computer (localhost). By default, the DetecTor enhanced TLS/TLS library will attempt to connect to five defined standard port numbers on localhost (proposed 9160, 9162, 9164, 9166, 9168), unless a configuration file or an enviroment variable is used to override that information. Each port is expected to behave as a regular SOCKS5 proxy, as provided by the Tor software.

In a corporate environment, if it's undesirable to allow each employee to open their own connections to the Tor network, the corporation could run the Tor software on behalf of their employees, and all employees could connect to the corporate gateways to the five desired Tor spheres.

The described technology doesn't work for probing intranet servers, as an Intranet service cannot be reached from outside the intranet, such as from Tor exit nodes. In intranets It will be necessary to use a configuration on each client computer, that instructs the DetecTor software to exclude the intranet domain from probing. Since an intranet should be tightly controlled by a small group staff, it should be possible to use other mechanism to monitor intranet servers for compromises. When providing external gateways into a corporate network, staff can still monitor the externally reachable intranet gateways using DetecTor.

Alternative configuration: Because of the described architecture, the use of Tor isn't strictly required for using the proposed probing technology. If desired, instead of routing DetecTor probes into the Tor network, a global organization could decide to run SOCKS5 network proxies at different global geographical locations, and configure clients to use those. However, while this alternative solution is independent of the Tor network, it might be less preferable, because it might be easier for an adversary to manipulate the network view observed by fixed corporate observation proxies.

Although a primary property of the DetecTor proposal is direct integration into Internet client software, it also depends on locally executing Tor router software. This dependency should be easy to fulfil for desktop or workstation computers. It might be a challenge to implement it on mobile devices and tablet computers. However, as demonstrated by the Orbot project for mobile devices running the Android operating system, it should be possible. An alternative could be to run the Tor software on a separate computer or smart device in a home network, and configure the tablet device to use that device as the gateway for performing DetecTor probes.

Implementation status

An initial version of the separate server probing tool (detector-probe), implemented using NSS and the C programming language , is already available. It can be used to query any publicly reachable SSL/TLS server by hostname and port number. It will run six connections - one using the local network and five through connected Tor spheres - and will report the consistency status. This could already be used for monitoring oneself's server for network manipulation attacks. It could also be manually used by users, by executing the tool and and comparing the certificate fingerprint reported by the probing tool with the certificate fingerprint reported by the local browser or email software.

A minimal sphere controller is already available, which uses the regular Tor

software and a small controller implementation to startup the required Tor spheres, based on a configuration file. It's implemented as a simple command line application using Python and the Stem controller library. An example sphere configuration is provided, which is based on continents. The five spheres used are Asia, Europe, North America, Africa and South America combined, Oceania and Antartica combined. At the time of writing this document, the number of Tor exit nodes available in Africa, South America and Oceania is very small, and are the bottleneck for the concept of using continental spheres, but it is working, albeit slowly.

The transparent extension of the NSS SSL/TLS library code still needs to be done. However, the work has started. As a first step, a patch is available that enables the NSS software to connect through a SOCKS5 proxy. The patch has been made available and is waiting integration into the NSPR/NSS libraries. The provided server probing tool already makes use of it.

Implementation challenges

It should be possible to implement the proposed SSL/TLS probing integration into the NSS library without much problems. However, the implementation will likely appear slow for each connection to a new site. If a site is accessed for the first time after having started an application, it might take from 5 to 30 seconds until a Tor network probes have completed. If a connection to a site requires loading content from multiple sites, that delay might be experienced for each dependent site. For subsequent connections within the same session, there won't be a delay if the same server certificate is being used, until the DetecTor decides that re-probing should be done. The validity period of probing results should be a configuration option.

The proposed solution will immediately work with many common protocols that fully wrap the application protocol inside SSL/TLS, such as https (regular browsing of secure sites), SMTP/SSL (port 465), IMAP/SSL (port 933), POP3/SSL (port 995), etc. However it probably cannot transparently work for the protocols that use a STARTTLS variant, which require a protocol specific plaintext handshake until the connection gets upgraded to use TLS. Examples are XMPP, IMAP/STARTTLS, LDAP/STARTTLS, SMTP/STARTTLS. In order to use DetecTor certificate probing for those protocols, application support will probably necessary, as each probing connection needs to be provisioned with the appropriate plaintext protocol handshake, prior to starting the generic SSL/TLS probing inside the DetecTor library.

Discussion

Feedback on this proposal is welcome. Please send your comments to the IETF mailing list named "therightkey" available at https://www.ietf.org/mailman/listinfo/therightkey

for initial discussions. Separate mailing lists dedicated to the DetecTor project might be setup at a later time.