The Tor Network - Strengths and Weaknesses

Computer Security 1 - CS5460

LaTeX

Jesse Victors

I. Introduction

The Tor network is a second-generation onion routing system that aims to provide anonymity, privacy, and Internet censorship protection to its users. Tor routes TCP traffic through a worldwide and volunteer network of over four thousand relays. Its encryption, authentication, and routing protocols are designed to make it exceptionally difficult for an adversary to identify an end user or reveal their traffic.

II. Design

Tor provides an anonymity and privacy layer by relaying all end-user TCP traffic through a series of *relays* on the Tor network. Typically this route consists of a carefully-constructed three-hop path known as a *circuit*, which changes over time. These nodes in the circuit are referred to as *entry guard*, *middle router*, and the *exit node*, respectively. Only the first node can determine the origin of TCP traffic through Tor, and only the exit node can examine the contents and its destination. Nodes in the middle are unable to determine either. No single node can determine the origin, the contents, and the destination of traffic through the network. Tor's architecture is designed to make it exceptionally difficult for a well-resourced adversary to uncover the identity of the end-user and their network activities, even if nodes are compromised.[1]

A. Routing

Routing information is distributed by a set of authoritative directory servers.[1]

Tor achieves anonymity by routing a user's traffic through a circuit of several random relays, wherein each relay is only capable of decrypting the address of the next relay in line. Hence, even compromised relays will be unable to deduce what the traffic is or where they came from, and an outsider is faced with multiple layers of public-key – either RSA or elliptic curve – encryption on top of TLS protocols. Currently, Tor is one of the most secure tools to use against network surveillance, traffic analysis, and to bypassing information censorship.

B. Encryption

Encryption is a necessary component for privacy within the Tor network. Traffic passing between nodes must be secured from outsiders, and even compromised nodes must not be able to observe the traffic in cleartext. Privacy is also important for traffic outside the network; encrypted connections between the exit node and the target web server must also be achieved. Furthermore, two layers are needed: one for inside the network and one for outside it.

Within the network, nodes talk to each other via the Transport Security Layer (TLS) protocol. When a circuit is built, each pair of nodes within it must first come to an agreement over an encryption key and the specific cipher to use for encryption. There are several mechanisms by which this can be done, all of which use Diffie-Hellman (DH) for key exchange. The most common methods include: TLS_RSA, which rely upon public and private keys generated with RSA; TLS_DHE, ephemeral DH; TLS_ECDH, DH based on elliptic curves, and TLS_ECDHE, ephemeral elliptic curve DH. It should be noted that only TLS_DHE and TLS_ECDHE provide perfect forward secrecy, so it is no surprise that Tor exclusively prefers them. Once encryption keys and the specific cipher (as well as its mode) have been agreed upon, communication of traffic between two nodes can be encrypted and exchanged.

Tor users typically use the Tor Browser Bundle, (TBB) a custom build of Mozilla Firefox with a focus on security and pricacy. The TBB not only provides special handling of client-side scripts such as Javascript, but also offers the HTTPS Everywhere extension, which uses regular expressions to rewrite web requests into ones that use HTTPS. Thus, if the web server is capable of handling SSL or TLS connections, HTTP communications will be encrypted under them. Like any typical browser, the TBB negotiates with the web server for the cipher and keys required for SSL/TSL, except that this negotiation entirely occurs through the Tor circuit. As previously noted, this information, along with subsequent HTTP data, is encrypted between each node in the circuit. During transmission, each node in the circuit decrypts its layer in turn, until the final node (the exit) passes the data to the target server. The TLS/IP connections remain open, so the returned information likewise travels back up the circuit to the end user.

In September 2013, Robert Graham of Errata Security analyzed 22,920 incoming connections to his exit node and found that 89.9 of the circuits agreed to use the Advanced Encryption Standard (AES) block cipher in cipher-block chaining (CBC) mode. This is the most common mode for AES, and allows for parallel decryption of the data blocks. The other 10.1 of the circuits relied upon the Data Encryption Standard (DES) cipher for encryption. Furthermore, 75.7% of the circuits used elliptic-curve DH, with the remaining 22.3% relying upon traditional RSA DH.[6] Both protocols are discussed below.

1) RSA: RSA is an algorithm for public-key cryptography. Its security is based on the infeasibility of factoring the

product of two large primes. RSA, like all other public-key cryptography algorithms, relies upon two keys: one public and the other private. The public key may be published and is used for encryption and for verifying digital signatures. The private key is used for decryption, the generation of digital signatures. Thus, through RSA, only the owner of the private key can decrypt incoming messages, and only they can digitally sign outbound messages. This makes RSA extremely powerful for both authenticity and privacy.

To generate the RSA keys, two distinct prime numbers of similar bit-length are chosen at random, p and q. n is computed as the product of p and q, and it is used as the modulus for both the public and private keys. The bit-length of n is the RSA key length. Then let (n) = (p) (q) = (p 1) (q 1) where is Euler's totient function, which counts the number of positive integers less than or equal to n that are relatively prime to n. Then e is chosen such that e and (n) are coprime. d is also chosen such that it is the multiplicative inverse of e modulo (n). The RSA public key then consists of the modulus n and the exponent e. The private key likewise consists of n and the exponent d.

A message M is encrypted by first blocks of length less than n using an agreed-upon padding scheme. The ciphertext c is then computed as mê mod n, which can be calculated quickly using the method of exponentiation by squaring. The recipient (the owner of the private key) can then recover the plaintext M by computing cd̀ mod n, and then reversing the padding scheme. The decryption process is often accelerated by optimizations such as the Chinese Remainder Theorem. Furthermore, to sign a message, the owner of the private key first uses an agreed-upon hash function to compute a hash value of the message, raises it to the power of d modulo n and then sends that in conjuction with M. The recipient, who only has access to the public keys, then raises the signature to the power of e modulo n and compares the result with his hash of the message using the same hash algorithm. If the two match, then he can prove the authenticity and integrity of the message. These two capabilities make RSA an extremely powerful tool for securing authenticity, integrity, and privacy of communication between two parties.

2) Elliptic-curve: Elliptic Curve Cryptography (ECC) is an approach to public-key cryptography in which elliptic curves are used instead of RSA. In contrast to RSA, ECC relies upon the infeasiblity of finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point. This is known as the elliptic curve discrete logarithm problem, or ECDLP.

Elliptic curves over the real number system can be defined through the equation, $y\hat{2} = x\hat{3} + a\,x + b$ Where a and b are real numbers. This formula describes a plane curve through the Euclidean plane, although elliptic curves do exists in other spaces. However, current cryptographic purposes only work with elliptic curves within the context of the real number systems.

A smaller key size is one of the most significant benefits introduced by ECC. Current NIST recommendations state that a key size of 160 bits for ECC offers a similar level of security to 1024 bits of RSA/DH. Likewise, 224 bits for ECC is analogous to 2048 bits of RSA/DH. It should be noted that the required key sizes for ECC increases significantly slower than the key sizes for RSA and Diffie-Hellman increase at a much faster rate than the required key size for ECC. Thus ECC offers more security per increase in key size than for RSA. The smaller key size also directly correlates to an increase in computational efficiency; as the bit size increases, the speed difference between Diffie-Hellman and ECC grows superlinearly.[5]

Tor 2.4.x introduced ECC into its DH key exchange protocols, known as *NTor*.

C. Authentication

Authentication...

III. Features

A. SOCKS

The Tor client software offers a Secure Sockets (SOCKS) interface on the system's localhost. SOCKS is different than HTTP proxying, although it uses handshake protocols for authorization purposes, it does not interpret or rewrite any of the subsequent traffic. The Tor Browser Bundle uses this interface as an input to the Tor network, although other applications may also utilize it. SOCKS then multiplexes traffic through the Tor circuit.

B. Hidden services

IV. Adversaries

It was recently revealed that the National Security Administration (NSA) has been targeting Tor, albeit with marginal success in breaking the anonymity and privacy of its users. However, in early October, the FBI successfully identified and arrested the owner of the Silk Road, a black market operating as a hidden service within the Tor network, and seized the service.

V. Challenges

TBB - javascript and cross-scripting Does the web server support SSL/TLS?

To achieve its low-latency objective, Tor does not explicitly re-order or delay packets within the network.[1]

References

[1] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, Douglas Sicker, *Shining Light in Dark Places: Understanding the Tor Network*. Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-2969, 2008.

[2] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, Douglas Sicker *Low-resource routing attacks against tor*. ACM, 2007

[3] Abdelberi Chaabane, Pere Manils, Mohamed Ali Kaafar *Digging into Anonymous Trafc: a deep analysis of the Tor anonymizing network*. IEEE, 2010

[4] Tor FAQ

[5] National Security Agency *The Case for Elliptic Curve Cryptography*. NSA, 2009

[6] Robert Graham *Tor is still DHE 1024 (NSA crackable)*. Errata Security, 2013