Tor Assignment

Fahad Farid and Samir Farhat Dominguez

April 20, 2022

1 Review Questions

In your own words, what is the difference between data and metadata? What metadata do TLS and encrypted email leak? Give us an imagined scenario for each of these protocols where this information leakages could have serious ramifications.

In TLS, data represents the payload of information that we wish to communicate to a party, while metadata is the encompassing and surrounding data that allows this information to be organized, transferred and interpreted.[1] Data is often hidden away either by encapsulation or encryption, while metadata is exposed on the surface of the channel.[1] As such, the metadata cannot be permitted to expose vulnerabilities or secrets. An enumeration of the possible fields that are leaked by metadata can be found below.[2]

- Layer 3/4 Information: Destination and source IP of the packets, TTL, and initial TLS sequence numbers.
- Size, UNIX time, and protocol description of the packet.[2]
- Ciphersuites and extensions supported by the client and server.[2]
- Server and Client certificates.[2]
- Sender/receiver addresses and public keys in the case of email.[3]
- Hardware/device from which email is sent.[3]

There are several issues and scenarios that can emerge from the leakage of metadata. First and foremost is the replay attack. An adversary can simply choose to repeat the exact same metadata transfer any number of times in any order hoping to achieve something productive.[1] Moreover, in TLS several pieces of information can be uncovered in your typical exchange; geographical location can be uncovered from UNIX time and IP, operating system and machine can be uncovered from packet size, certificates can be taken and used to impersonate at both the server and client side, network enviornment and location can be uncovered from layer 3/4 information and effective attacks can be designed according to the supported ciphersuites and extensions supported.[2] In the case of encrypted email, senders can be impersonated with public keys and sender information.[3]

List and explain 2 benefits of using three nodes in a Tor circuit rather than simply using a single proxy server.

The following two benefits are notable for using 3 nodes for tor circuits rather than a single proxy. First, 3 is deemed sufficient to obfuscate enough in order to achieve anonymity while not incurring higher delays(by say using 4 nodes); and second, the 3 relay model reduces the probability of a circuit where entry and exit points are compromised(the critical points).[4] In terms of the anonymity/obfuscation benefit the 3 node model makes it such that compromising a user is much more complex. In a 2 node system(local and proxy), if an adversary owns or even observes the exit node(the proxy in the case of our example) the adversary would know the exact node to compromise next to fully control the entire circuit and thus the user. This earlier node would be the guard node, which will be utilized for an extended period of time while performing tor operations, so this becomes a very high-value target to deanonymize the target by deciphering there local/native network enviornment or server. Naturally, 4 nodes would theoretically provide further obfuscation and stronger anonymity, but at a certain point a cost-benefit decision needs to be made regarding the acceptable delay. Since an additional hop just adds further propagation in terms of timing and network bandwidth.

In terms of probabilistic advantages, the 3 node model simply makes it unfeasible for an adversary to attempt to use previous gained information to try and compromise the user upon a re-connection. As we know, a tor client will select a few relays at random and choose from this subset as an entry point. Since this is a relatively small subset, an attacker would be able to eventually gather the list of possible nodes, and thus monitor to find the correct exit to compromise. However, by adding the middle node, the cumulative probabilities become much more unlikely for this compromise to take place. Thus protecting the identity and traffic of the client.

2 Setting up a Tor Circuit Write-up

The `telescoping_circuit.py` script is used to setup a tor circuit around the 3-node/hop model and perform an http request. The `main` function consumes the command line arguments and then calls the `get` function. In this function, we start by downloading the tor consensus and if random mode is selected we use that to select unique random nodes for the 3 nodes. We use this selection and begin by creating a circuit base from the initial guard router in the `circuit_from_guard`. Here, the `CreateCell` is created as per the tor specification and a circuit node is made with the fast-key-agreement. As such, we generate the random digest `X` and retrieve the public `Y`, concatenating them to perform the 'ntor' handshake.

With this circuit base, we call the `circuit_build_hops` function with the selected middle and exit routers in order to construct the rest of the 3-node circuit. In the function we call `extend` with both remaining nodes to produce this final circuit. In the `extend function` we retrieve our private key, public key, and public B and use these to construct our onion skin and then create the `ExtendedCell`. We then compute the relevant parameters for the secret input and use that to construct the shared secret as a sanity check for the correct procedure between nodes. We now have our complete circuit and construct and send the HTTP GET request, dumping the output in the file specified at the command line.

3 Connecting to a Hidden Service Write-up

In the `hidden_service.py` we seek to establish a connection with a hidden service. We start by taking a pre-built 3-node circuit as above and establish a TCP stream. We then take the hidden service hostname and call the `stream_prepare_address` function where we receive the hidden service object and its respective address. From there we need to extend our circuit to a 4th hop, namely the one that functions as an introduction point to the hidden service. This is done in `extend_to_hidden`. In this function we begin by getting the hidden service directory and respective introduction point router with the `get_directories`, `get_introductions` and `get_introduction_routers` functions. Then, we ascertain our rendezvous point router with the hidden service.

We then bring the above together and call `set_up_intro_point` to ascertain the introduction point with the hidden service. In this function we run our half of the Diffie-Hellman handshake by computing our private and public key. We then get our `introduce_cell` and receive our `cell_acknowledgment`. We use this to extract the public key and derivative key data which we use to compute the shared key and verify respectively. We store the shared key for all future communications and return the introduction point. This is introduction point is appended to our original circuit thereby connecting to the hidden service and we receive and return the response as a confirmation of the connection.

4 Point your code at live Tor

* Work relevant to this section is contained entirely in the zipped project.

Bibliography

[1] Ford, Bryan. "Metadata Protection Considerations for TLS Present and Future (draft)."

[2] Bauer, K., and McCoy, D. Tor specification proposal 109: No more than one server per ip address. https://gitweb.torproject.org/torspec.git/plain/tor-spec.txt

[3] Bauer, K., and McCoy, D. Tor rendevous specification

https://gitweb.torproject.org/torspec.git/plain/rend-spec-v2.txt

[4] Lerner, Ada, Eric Zeng, and Franziska Roesner. "Confidante: Usable encrypted email: A case study with lawyers and journalists." 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2017.

[5] Kohls, Katharina, et al. "On the Challenges of Geographical Avoidance for Tor." NDSS. 2019.

[6] Dingledine, Roger, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Naval Research Lab Washington DC, 2004.

[7] Huete Trujillo, Diana L., and Antonio Ruiz-Martínez. "Tor Hidden Services: A Systematic Literature Review." Journal of Cybersecurity and Privacy 1.3 (2021): 496-518.