# Project Proposal

## UbiCrypt: Making ubiquitous encryption compatible with enterprise security

*Weisse, Ofir* *Trippel, Timothy* *Erickson, Jeremy*
*EECS 589, Fall 2015*
*University of Michigan, Ann Arbor*

## 1 Introduction

With recent data breaches[6], many in the security community are calling for ubiquitous encryption to become the norm. That is, all online communications, whether secret or mundane, should be end-to-end encrypted. Several well-known projects[1, 9] are pushing for all web traffic to be encrypted with Transport Layer Security (TLS), also known as HTTPS. As this trend continues, users will become more secure from intermediate entities snooping on their data or being able to modify it in transit. Ubiquitous encryption is very promising.

However, the same defenses that protect users against a malicious third party viewing or modifying data in transit block enterprise security measures from being able to introspect on corporate traffic and provide legitimate security services to the company. For instance, a company may use Snort[4], a common network Intrusion Detection System (IDS), to parse incoming network traffic for signatures indicating malware has been downloaded by an employee. If the network flow is encrypted, Snort will be unable to identify malicious signatures[8].

In enterprise environments, often the need to introspect on corporate network traffic is higher than the need for true end-to-end security. The naive solution in place today is for a company to simply man-in-the-middle its employees' secure connections at the enterprise gateway[7]. It does this by supplying an enterprise-owned root Certificate Authority (CA) to each user and creating two encrypted sessions for each connection — one between the client and gateway and one between the gateway and server.

This has several downsides stemming from the fact that the client must trust the gateway to securely connect to the server on its behalf. The gateway may trust a certificate that the client does not, thus reducing the client's ability to discern untrusted connections, or the gateway may not trust a certificate that the client does, thus either blocking the connection or making the connection without the assurance of authentication the client would be able to achieve on its own. For instance, one of the core tenets of public/private key authentication is that, even without a Public Key Infrastructure (PKI) to distribute and validate keys, the client can manually import a public key as trusted and form a secure connection to the owner of the corresponding private key. Not so if the gateway intercepts the connection.

## 2 Approach

Instead, we propose a new approach in which the client and server are allowed to form a secure end-to-end encrypted connection. To enable the gateway to introspect

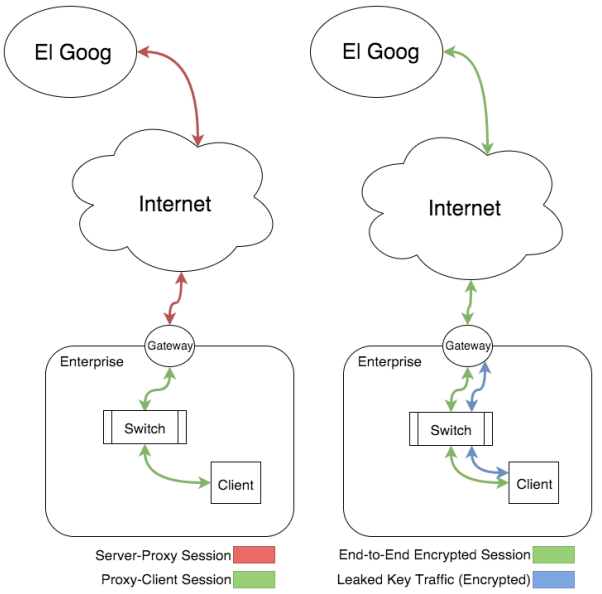


Figure 1: Comparison of existing proxied network flows (left) in an enterprise network vs. proposed proxied network flows (right) in an enterprise network.

on client traffic, we will provide a mechanism for the client to securely leak its session key to the gateway. The session key is unique to the specific connection, so in contrast to other solutions[5] that require leaking private keys, the longevity of the leaked key is short.

With the leaked session key, we will also provide the gateway with a utility to identify encrypted flows, decrypt them, and pass them on to an IDS. This will provide a capability equivalent to the existing solution, yet protect the integrity of a true end-to-end encrypted connection.

Specifically, we propose to build this capability into the QUIC[3] protocol, as it:

- Is implemented at the application layer, and therefore more accessible for modification than HTTPS.
- Is a recently-developed protocol and as adoption is still growing, there is more of a chance to have our proposed functionality incorporated in the mainline.

## 3 Project Plan

Given this project will be focused on the design of a new system, this project will be broken down into several phases. These will consist of four main phases: infrastructure, client application, proxy, and evaluation. For infrastructure setup, we plan to use minimega[2] to deploy a topology of virtual machines for the evaluation of our project. These machines will consist of *a*) one or more clients that attempt to retrieve websites and initiate QUIC sessions and *b*) one gateway that proxies traffic for all clients. We plan to use connections to real websites as the servers in our model. For the client application phase we propose modifying the QUIC source code within Chromium browser. This will allow us to implement our proposed introspection scheme on a common and real-world use case. Our gateway infrastructure will proxy clients' ingress and egress traffic and decrypt QUIC-encrypted flows for Snort to process. Lastly our project will be evaluated as described below. Below we show our project milestones and a projected timeline.

**Project Timeline:**
*Milestone #1* - Infrastructure - Due 10/30
*Milestone #2* - Client Application - Due 11/13
*Milestone #3* - Gateway Deployment - Due 11/20
*Milestone #4* - Evaluation - Due 11/27
*Milestone #5* - Report/Presentation - Due 12/4

## 4 Evaluation

In order to evaluate the implementation we will test it from the perspective of three actors. First is the client. Does the client notice any difference from regular browsing? Does he receive any suspicious warnings from the browser? The second actor is the middle box. Can the middle box inspect the ongoing traffic in a way that is useful for it and allows it to decide if the communication is spurious or not? Last but not least, the third actor is a malicious third party. Can he leverage the new technology to gain information about the encrypted communication?

## References


[1] *Let's Encrypt*. URL: https://letsencrypt.org/.

[2] *minimega: a distributed VM management tool*. URL: http://minimega.org/.

[3] *QUIC, a multiplexed stream transport over UDP*. URL: https://www.chromium.org/quic.

[4] *Snort*. URL: https://snort.org/.

[5] *SSL Decryption*. URL: https://www.gigamon.com/products/technology/ssl-decryption.

[6] Edward Moyer. *NSA disguised itself as Google to spy, say reports*. Sept. 12, 2013. URL: http://www.cnet.com/news/nsa-disguised-itself-as-google-to-spy-say-reports/.

[7] Tim Chiu. *The Growing Need for SSL Inspection*. June 18, 2012. URL: https://www.bluecoat.com/security/security-archive/2012-06-18/growing-need-ssl-inspection.

[8] Joel Esler. *SSL/TLS*. Dec. 4, 2012. URL: http://manual.snort.org/node147.html.

[9] Barry Schwartz. *Google SSL Default, Goodbye Query Referrer Data*. Oct. 19, 2011. URL: https://www.seroundtable.com/google-ssl-drops-query-data-14188.html.