Securing the Internet, One HTTP 200 OK at a Time

WILFRIED MAYER, KATHARINA KROMBHOLZ, MARTIN SCHMIEDECKER, AND EDGAR WEIPPL

Wilfried Mayer is a Researcher at SBA Research and a PhD student at Technische Universität Wien. His research interests are mainly empirical security measurements. He also likes to make the Internet a better place. wmayer@sba-research.org

Katharina Krombholz is Senior Researcher at SBA Research. Her research focuses on usable security, privacy, and digital forensics. She is currently interested in usable security aspects of crypto deployments, IoT, and usable security research methodology. kkrombholz@sba-research.org

Martin Schmiedecker is Senior Researcher at SBA Research. His research interests include everything related to digital forensics, online privacy, and applied security. mschmiedecker@sba-research.org or @Fr333k

Edgar Weippl is Research Director at SBA Research and Associate Professor (Privatdozent) at the Technische Universität Wien. His research focuses on applied concepts of IT security and e-learning. eweippl@sba-research.org

HTTPS is the most commonly used cryptographic protocol on the Internet. It protects communication content and provides endpoint authenticity at scale. However, deploying HTTPS in a truly secure fashion can be a challenging task even for experienced admins. To explore why this is the case and how these challenges can be fixed in order to support an even wider adoption, we conducted a user study, which was presented at USENIX Security 2017.

Targeting the Long Tail

Nowadays, major online services provide TLS encrypted communication. But is the Web site of your local sushi restaurant secured with HTTPS? Because even your local sushi place needs HTTPS! We need HTTPS as the new standard in the Internet for all Web sites and to finally deprecate unencrypted HTTP. But often the opposite is stated. Even at USENIX Security, somebody in the audience questioned the need for HTTPS for small businesses or static content. The answer was simple: more confidentiality, authenticity, and integrity make the Web a more secure place. We know that the upgrade to HTTPS is a way, and a quite promising way, to improve the security of the entire Internet. This improvement is strongly needed, so let's upgrade to HTTPS.

Shifting to HTTPS is not just the problem of single service providers, and the process of enabling it is not just a problem of some individuals. It is an enormous challenge for all of us. Developers, administrators, and security researchers are working on it, and the situation is improving steadily. We see progress, as Felt et al. showed in their recent work [1]. The number of HTTPS page loads is rising and is now exceeding 50% of strict page loads in Firefox telemetry, and there are similar results with Google Chrome statistics. But their work also identifies weak spots. HTTPS usage on Android devices, the deployments in eastern Asia, and the long tail of Web sites are lagging behind. With our study [2], we focus on this long tail of Web sites that are not supporting HTTPS in contrast to major service providers. These companies—ranging from medium enterprises to your local sushi restaurant—do not have highly specialized security experts in charge, actively checking their Web sites for improvements. Normal IT departments, single administrators, and "IT guys" are solving problems here.

And as we already speculate from anecdotes and personal experience, it is not easy to deploy HTTPS in a secure yet compatible fashion. You probably remember your first time, and maybe your last time, setting up HTTPS. Even after initial deployment, the sheer complexity of keeping it secure can overburden experienced admins. Back in 2015, Yan Zhu created a video of knowledgeable members of EFF, who had never set up HTTPS before, being unable to deploy it within a limited amount of time [3].

Enough anecdotes and speculations. We decided to conduct a user study to analyze these problems. We did this without automated tooling provided by Let's Encrypt and chose the traditional approach of using a non-automated CA to issue certificates.

User Study

For the user study, we conducted a series of lab experiments with 28 participants. We recruited students with expert knowledge in the field of security and privacy-enhancing protocols at our university who fulfilled the criteria to potentially work as an administrator or were actually working as administrators. The participants were invited to the lab where they were briefed about the purpose of our study. They assumed the role of administrator of an SME who is in charge of securing the communication to an Apache Web server with HTTPS in order to pass a security audit.

We prepared and implemented a fictive Certificate Authority (CA) in order to facilitate the process of getting a valid certificate and to remove any bias introduced by the procedures from a certain CA. The fictive CA was available through a simple Web interface and required the submission of a valid CSR (certificate signing request) for issuing a valid certificate. The user interface was very simplistic, and the browser on the local machine already trusted our CA. We opted for this study setting because we wanted to focus solely on the actual deployment process instead of on the interaction with a CA. There was no existing TLS configuration on the system—hence the participants had to start a new configuration from scratch. We chose Apache for our experimental setup because Apache maintains a clear lead regarding in-usage share statistics.

We instructed the participants to make the configuration as secure as possible, whereas the assignment did not contain any specific security requirements. In order to collect data, we used a think-aloud protocol. While the participants were working on the task, they articulated their thoughts while an experimenter seated next to them observed their work and took notes. We refrained from video recording due to the results from our pre-test during which we filmed the sessions and noticed a severe impact on the participants' behavior. The participants from the prestudy also explicitly reported that they perceived the cameras as disruptive and distracting, even though the cameras were placed in a discreet way. In addition to the notes from the observation, we captured the bash and browser history and the final configuration files. After completing the task, the participants were asked to fill out a short questionnaire with closed- and open-ended questions that covered basic demographics, previous security experience in industry, and reflections on the experiment.

As a result, we had a collection of both qualitative and quantitative data that was further used for analysis. For a qualitative analysis of the observation protocols, we performed a series of iterative coding which is often used in usable security research to develop models and theories from qualitative data. To evaluate the (mostly) quantitative data acquired via the bash/browser history and Apache log files, we applied metrics and measures to evaluate the quality of the resulting configuration.

Results

For the security evaluation, we based our evaluation criteria on Qualys' SSL Test [4]. We consider this rating scheme a useful benchmark to assess the quality of a TLS configuration based on the state-of-the-art recommendations from various RFCs and with respect to the most recently discovered vulnerabilities and attacks in the protocol.

The rating of the evaluation criteria is expressed with grades from A to F and composed out of three independent values: protocol support (30%), key exchange (30%), and cipher strength (40%). Some properties, e.g., support for the RC4 cipher, cap the overall grade. Only four participants managed to deploy an A grade TLS configuration. B was the most commonly awarded grade (15 out of 28). Four participants did not manage to deploy a valid TLS configuration in the given time.

Our qualitative analysis of the think-aloud protocols from our lab study yielded a process model for a successful TLS configuration. All participants who managed to deploy a valid configuration in the given time can be mapped to the stages presented in this model. The four participants who did not manage to deploy TLS in the given time significantly deviate from this model.

We divide the steps from our model into two phases, a setup phase and a hardening phase. The setup phase refers to a set of tasks to get a basic TLS configuration, i.e., the service is reachable via HTTPS if requested. The hardening phase comprises all necessary tasks to get a configuration that is widely considered secure with respect to the metrics defined by the Qualys SSL Server Rating Guide [5]. Participants who achieved at least a basic configuration successfully completed all steps of the setup phase, while better-graded configurations completed some steps from the hardening phase as well. We identified iterative (tool-supported) security testing as a key element for a successful hardening phase since the participants relied on external sources to evaluate the quality of their configuration.

Usability Challenges

With these results, we identified several usability challenges:

- First, searching for information and finding the right workflow. Our participants visited a high number of Web sites and used multiple sources of information. The average number of visited Web sites was 60, and the most visited pages were the Ubuntu wiki and the official Apache documentation. We found that the participants jumped between sources that were not compatible to each other and could not assess the correctness of the used sources. This included outdated recommendations as well as incomplete tutorials that only covered, for example, the setup phase.

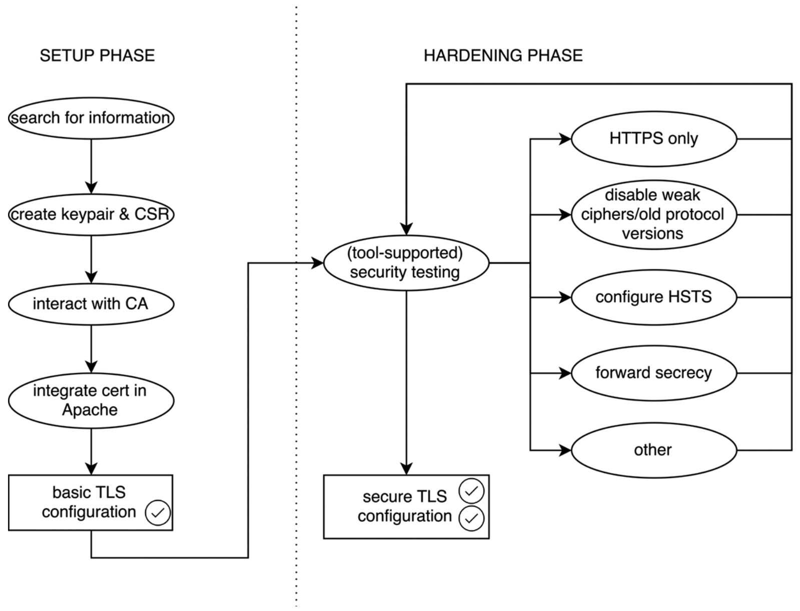

Figure 1: Schematic representation of a successful workflow

- Second, creating a Certificate Signing Request (CSR). The creation of a key pair and the CSR is part of the setup phase. We found that many participants had problems understanding the concept and struggled manually creating the CSR correctly on the first try.
- Third, choosing the appropriate ciphersuites. Ciphersuites define the underlying cryptographic primitives used. A sane configuration defines a limited set of ciphersuites with strong authentication and encryption. This is enabled via one specific Apache directive, "SSLCipherSuite," with the correct values. However, a deep understanding of the underlying algorithms is necessary in order to make an informed decision. All participants trusted online sources because of their missing knowledge, which implies that the quality of these sources is crucial.
- Fourth, strict HTTPS. After finishing a valid HTTPS configuration, most participants tried to enforce HTTPS via redirects and HTTP Strict Transport Security (HSTS) as the first step of the hardening phase. Most participants were initially confused with the default HTTP response when they entered the URL without protocol prefix. They spent a significant amount of time configuring this step correctly.
- Fifth, multiple configuration files. All but six participants struggled with the configuration file structure, regardless of their experience with Apache. Several participants did not understand how to enable the SSL module or where to configure the entry "SSLEngineOn."
- Last, finding the right balance between security and compatibility. In our scenario we didn't specify which level of security the participants should deploy but stated they should make it *as secure as possible*. About 15 of the participants expressed concerns regarding compatibility when configuring SSL/TLS versions, but the majority opted for the more secure options.

Our results reveal that these usability challenges are a serious issue to work on, and that they are the main reason for weak configurations. Our results also show that there is a high demand for improved tool support of the configuration process and more secure default configurations. This would prevent administrators from dealing with mechanisms they cannot fully understand. Fortunately, there are already tools out there. The impact of these tools ranges from generating sane configurations, to comprehensively changing the TLS ecosystem as a whole. We discuss four tools that are all tackling these usability challenges.

Tool Support

One of the projects with the biggest impact on the TLS ecosystem, especially the long tail, is Let's Encrypt (letsencrypt.org), "a free, automated, and open Certificate Authority." While the cost-free issuance of certificates takes away any economic reason to not implement HTTPS, and the open design facilitates transparency and continuous monitoring, the automated fashion of Let's Encrypt highly improves the usability of certificate issuance. This corresponds to the setup phase (the first column in the image) of our identified TLS deployment process model. All setup phase steps are replaced with a single tool-supported step. The certificate issuance is completely automated with the ACME protocol, so no further manual input is needed and the major Web server software, e.g., Apache, is also integrated. Let's Encrypt changed the TLS ecosystem significantly, with now more than 100 million certificates issued.

Tool support for the second phase—the hardening phase—is also quickly improving. For choosing the appropriate ciphersuites and associated compatibility issues, Mozilla published their Mozilla SSL Configuration Generator [6]. With selected server software and a chosen compatibility mode, the tool generates a valid and sane configuration. This can easily be copied and pasted into the correct server configuration file. Complicated decisions are replaced with few more easily understandable options. The compatibility level has three profiles: old, intermediate, and modern. The tool shows the oldest compatible clients upon selection and also adds correct HSTS headers to the configuration. So it also unburdens the use of HSTS.

Further tool support exists for testing the deployed security configuration. For publicly reachable domains this enables iterative configuration with repeated security testing until a specific grade is reached. The most established testing tool is the Qualys SSL Server Test, the same tool we graded the participants' configurations. With Hardenize (hardenize.com) this concept is widened to DNS, email, and application security.

In our study, we addressed the ecological validity of our results by conducting additional expert interviews with experienced security consultants. One expert mentioned the Web server software Caddy (caddyserver.com). It comes with a secure TLS default configuration and automatically uses Let's Encrypt to retrieve certificates. Initially delivering HTTPS with your software is a good example of the paradigm of secure defaults. We have to shift this paradigm to other major Web server software.

This tool support, although not yet fully integrated, provides important assistance for administrators, but there is still a lot of work to do to shift the Internet to HTTPS.

Outlook

As mentioned, the Firefox telemetry data showed that more than 50% of Web pages are loaded over HTTPS. This is an enormous success, and the overall trend is definitely pointing in the right direction. We see a lot of effort to shift the ecosystem towards HTTPS. With TLSv1.3, not only the security but also the performance of TLS is increased, making HTTPS even more attractive. We see more and more hosting platforms switching to HTTPS for their customers, which is increasing HTTPS usage for the long tail at scale. New upcoming standards, like the ACMEv2 standard [7] are improving the automation of certificate issuance. And the future support of wildcard certificates with Let's Encrypt [8] will form the ecosystem even more.

Conclusion

Administrators of the long tail of the Internet should sometimes also be seen as users. Some configuration tasks still require a deeper understanding of security mechanisms or even underlying cryptographic methods. If we want these security mechanisms to work, we also have to support administrators in enabling them. With this in mind, we should all work on shifting the ecosystem, until even the long tail supports HTTPS so that we can finally move on to the Internet where HTTPS is the norm.

References

[1] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, "Measuring HTTPS Adoption on the Web," in *Proceedings of the 26th USENIX Security Symposium (Security '17)*, pp. 1323–1338: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-felt.pdf.

[2] K. Krombholz, W. Mayer, M. Schmiedecker, and E. Weippl, "'I Have No Idea What I'm Doing'—On the Usability of Deploying HTTPS," in *Proceedings of the 26th USENIX Security Symposium (Security '17)*, pp. 1339–1356: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-krombholz.pdf.

[3] Y. Zhu, "(mis)adventures in setting up HTTPS": https://www.youtube.com/watch?v=Q0VdlLG7t1w.

[4] Qualys SSL Labs, SSL Server Test: https://www.ssllabs.com/ssltest.

[5] I. Ristić, SSL Server Rating Guide: https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide.

[6] Mozilla SSL Configuration Generator: https://mozilla.github.io/server-side-tls/ssl-config-generator/.

[7] ACMEv2 API Endpoint Coming January 2018: https://letsencrypt.org/2017/06/14/acme-v2-api.html.

[8] Wildcard Certificates Coming January 2018: https://letsencrypt.org/2017/07/06/wildcard-certificates-coming-jan-2018.html.