Assignment 6 - E-mail security (part 3)

Security

Sébastien Vaucher
sebastien.vaucher@unine.ch

13 November 2017

1 Technical introduction

In this assignment, you are going to work with a couple of technologies related to Transport Layer Security (TLS) and the Domain Name System (DNS). You will implement the validation of X.509 certificates using DNS-based Authentication of Named Entities (DANE) [1], secured by Domain Name System Security Extensions (DNSSEC) [2].

Unless you already have a strong grasp of DANE, I highly recommend reading the issue paper published by AFNIC, the entity responsible for French Top-Level Domains (TLDs). It is available in French [3] or in English [4].

The exact specification on how to implement DANE for the Simple Mail Transfer Protocol (SMTP) is described in RFC 7672 [5]. Reading this in-depth technical specification should not be needed to realize this assignment. Should you be interested in the subject, I suggest reading the related summary article written by S. Bortzmeyer (in French only) [6].

2 Assignment instructions

In this assignment, you will develop an SMTP gateway to filter e-mails before forwarding them to a fully-featured SMTP server. From the perspective of the user, your gateway will be the one acting as his/her SMTP server. Your server will analyze the e-mails it receives and modify them according to one or many filters.

In this third part, you will enhance the security of your relay. We will focus on making the link between your program and the fully-featured SMTP server secure.

2.1 Part 3: TLS connection with DANE validation

In this part of the assignment, we will fix an important vulnerability affecting your SMTP gateway. As it stands, connections to the remote fully-featured SMTP server are done in clear text. Any middleman can intercept the connection, including all the e-mails that get transmitted. Moreover, if you use DNS name resolution to look for the endpoint, an attacker could misdirect your connection using various DNS-based attacks. As your gateway does not check the identity of the remote SMTP server it connects to, it is impossible for it to detect that it is under attack. Using the DANE protocol and its friends TLS and DNSSEC, we will be able to fix all of these problems.

2.2 Implementation

The goal of your implementation is to make sure that your gateway only connects to remote SMTP servers using an encrypted connection. Moreover, it will have to check that the server to which it connects is indeed the one designated by the domain of the recipient of the e-mail.

Requirements of your implementation:

1. All DNS queries need to be authenticated using DNSSEC
   a) Validation of DNSSEC-signed data can be done by an external DNS resolver connected through a trusted link. Your program would only check for the presence of the `AD` bit (Authenticated Data) in each reply. Note that you have to enable Extension mechanisms for DNS (EDNS) and set the `DO` bit (DNSSEC OK) in your query if you want the resolver to confirm that DNSSEC validation is performed.
   b) Alternative: *(Better approach, bonus points will be awarded for it)* You may implement the DNSSEC validation within your program. Remember to enable EDNS and set the `DO` bit in your queries so that DNSSEC signatures get sent by the resolver for each reply.
2. SMTP connections emanating from your gateway must be encrypted by the TLS protocol, using the STARTTLS mechanism of SMTP [7]
3. All X.509 certificates presented during a TLS handshake must be validated using DANE
   a) You may only implement certificate usage 3 (Domain-issued certificate), selector 0 (Full certificate) and matching type 1 (SHA-256) of the DANE specification [1]
   b) The classical Certificate Authority (CA)-based certificate validation has to be disabled, as the X.509 certificates in use may be self-signed
4. If any of the verification steps described in this list fails, your program will abort the connection attempt, and log an informative message about the situation in the console

Here is an example of a successful DANE-validated TLS connection:

1. An e-mail with `user@security2017.vaucher.org` as recipient is received by your SMTP gateway
2. The gateway applies the content filters that the user has enabled, exactly as in the previous 2 parts of the assignment
3. The gateway requests the MX records for the domain name of the e-mail address (`security2017.vaucher.org`) from its DNS resolver
   a) 1 MX record is returned, pointing to the mail server at `mailtest.vaucher.org`
   b) The DNS reply is validated using DNSSEC
4. The gateway requests the TLSA records associated to the mail server
   a) The DNS reply is validated using DNSSEC
5. The gateway initiates a clear-text SMTP connection towards the mail server: `mailtest.vaucher.org` on port 10587[1]
6. After the remote mail server sends the greeting and the gateway replies with the `EHLO` SMTP command, the gateway asks the server to switch to an encrypted connection using the `STARTTLS` SMTP command
7. After the TLS handshake has happened, your SMTP gateway will have received the X.509 certificate of the remote mail server
   a) The certificate presented by the mail server is checked against the TLSA records obtained at step 4
8. If all the steps above were successfully completed, then the gateway can forward the e-mail as usual, with the guarantee that the remote mail server is trustworthy

2.3 Test environment

As few public SMTP servers implement the server-side requirements for DANE, we provide one for you. Here are the relevant information for the SMTP service:

Service name[2]	`security2017.vaucher.org`
Test e-mail address	`user@security2017.vaucher.org`
Port number	10587
Authentication	*none*

[1]In a real-life scenario, we would use port 25.

[2]What is on the right-side of the @ character in an e-mail address. It is different than the mail server's fully-qualified domain name (FQDN).

Encryption mandatory STARTTLS

You can check the e-mails received by the test address using the following webmail access:

URL `https://mailtest.vaucher.org:10401/roundcube/`

Username `user`

Password `security2017`

You need to use a DNSSEC-aware DNS resolver to be able to implement DANE. Most recent DNS resolvers should be able to process requests asking for DNSSEC signatures (`DO` bit set in the query). If your resolver is too old, you can use Google Public DNS[3] as your resolver.

You can validate that your DNSSEC validation implementation works by trying to query `dnssec-failed.org`. If the resolver you use correctly validates DNSSEC signatures, you should get a `SERVFAIL` error.

You can send e-mails to `user@security2017broken.vaucher.org` to check that your DANE validation steps work. The MX records of that domain point towards a different domain name (`mailtestbroken.vaucher.org`) that voluntarily contains a TLSA record that does not match the certificate presented by the server. If your implementation is on point, your gateway will refuse to connect to that server.

2.4 Open questions

After you have finished implementing this assignment, please answer the following open questions by providing your answers in your Readme file:

1. Imagine that you are an attacker. You want to perform a man-in-the-middle attack against the DANE-verified TLS connection. Your goal is to tamper with the e-mails sent through that channel. You have the ability to inspect, change and redirect all the network traffic between the endpoints.
   a) How would you proceed?
2. Even with the security protocols that we implemented in this assignment, a middleman can still tell which domains our relay sends e-mails to.
   a) What kind of traffic should the middleman intercept to get that information?
   b) How could we fix this leakage of information?
   c) *(Bonus)* Implement the fix in your assignment

[3] `https://developers.google.com/speed/public-dns/docs/using`

3 Hand-in

The time allotted for this assignment is 2 weeks. The deadline is on 2017-11-29T13:59:59 local time. Late submissions are not accepted.

- To be submitted to Ilias[4]:
  - Source code of **your** assignment
  - Readme file briefly mentioning how to compile and run your program, which dependencies it requires, etc.
    - For this assignment, your Readme also has to contain your answers to the open questions asked in subsection 2.4
  - All the files have to be packed in an archive in a standard format[5], named following this exact pattern (in lowercase letters only):
    `security17-as<assignment number>-<your family name>.<extension>`.
    For example, if your name were to be *Homer J. Simpson*, you would use the following filename for this assignment: `security17-as6-simpson.tar.gz`
  - Please use the "Upload File" button when handing-in your assignment in Ilias. Do **not** use "Upload Multiple Files as Zip-Archive".
- You have to present a demonstration of the program in class (to the TA).
  - It is **mandatory** for each student to demonstrate his or her submission!
  - The sooner you present your assignment, the better (even before the deadline).

Your grade will depend on the presentation, the code and your answers to the questions of subsection 2.4.

4 Notes

You can use your favorite programming language for the assignments of this course, so long as it is a programming language readily available on the GNU/Linux operating system[6]. The suggested language for this assignment is Python. I recommend using the `dnspython`[7] package to perform DNS queries.

Should you have additional questions, please direct them to the TA at sebastien.vaucher@unine.ch.

[4]Or sent by e-mail for external students

[5]`.tar`, `.tar.gz`, `.tar.bz2`, `.tar.xz`, `.zip`

[6]You can use any of the languages in the following list. If you want to use another language, please check with the TA first. List in alphabetical order: Bash, C, C++, Go, Java, Kotlin, Perl, PHP, Python, Ruby, Rust, Scala

[7]`http://www.dnspython.org/`

References

[1] P. Hoffman and J. Schlyter, “The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA,” RFC Editor, RFC 6698, Aug. 2012. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc6698.txt`.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “DNS security introduction and requirements,” RFC Editor, RFC 4033, Mar. 2005. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc4033.txt`.

[3] Association Française pour le Nommage Internet en Coopération, “Sécuriser les communications sur Internet de bout-en-bout avec le protocole DANE,” French version, in *Dossiers thématiques*, 12, Nov. 2013. [Online]. Available: `https://www.afnic.fr/medias/documents/Dossiers_pour_breves_et_CP/dossier-thematique12_VF1.pdf`.

[4] ——, “Securing Internet communications end-to-end with the DANE protocol,” English version, in *Issue papers*, 12, Nov. 2013. [Online]. Available: `https://www.afnic.fr/medias/documents/Dossiers_pour_breves_et_CP/dossier-thematique12_VEn1.pdf`.

[5] V. Dukhovni and W. Hardaker, “SMTP security via opportunistic DNS-based authentication of named entities (DANE) transport layer security (TLS),” RFC Editor, RFC 7672, Oct. 2015. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc7672.txt`.

[6] S. Bortzmeyer, “RFC 7672: SMTP security via opportunistic DANE TLS,” *Blog Stéphane Bortzmeyer*, Oct. 2015. [Online]. Available: `https://www.bortzmeyer.org/7672.html`.

[7] P. Hoffman, “SMTP service extension for secure SMTP over transport layer security,” RFC Editor, RFC 3207, Feb. 2002. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc3207.txt`.