Belenios: the Certification Campaign

Angèle Bossuat[1], Eloïse Brocas[1], Véronique Cortier[2], Pierrick Gaudry[2], Stéphane Glondu[2], and Nicolas Kovacs[1]
{abossuat,ebrocas,nkovacs}@quarkslab.com
{veronique.cortier,pierrick.gaudry}@loria.fr
stephane.glondu@inria.fr

[1] Quarkslab
[2] LORIA - CNRS, Inria, Université de Lorraine

Abstract. This article presents our work on the security evalutation of the Belenios voting solution as a *Certification de Sécurité de Premier Niveau*, the French simple alternative to Common Criteria. We present the steps taken by the LORIA, who designed Belenios, and Quarkslab, who carried out the evaluation, as well as our exchanges with the ANSSI, who coordinated everything.

We also hope to start a more general discussion on how to evaluate the security of e-voting systems in practice, as they are difficult to compare with other types of software, and any vulnerability may have an impact at a large scale.

1 Introduction

Electronic voting solutions have gained significant attention in recent years as a means to enhance the efficiency and accessibility of the voting process. As for all digitalized processes (e.g., e-mails, banking apps, grocery shopping), this evolution and simplification should not come with the introduction of vulnerabilities "worse" than the physical version.

The e-voting solutions possess unique characteristics that distinguish their security assessment from other types of software. Furthermore, their vulnerabilities can have significant consequences on a large scale.

This paper outlines our effort to evaluate the Belenios voting solution in a "Certification de Sécurité de Premier Niveau" (CSPN) process, which is the French (lighter) equivalent of Common Criteria for security certification. This process involved collaboration between LORIA researchers, the creators and maintainers of Belenios, the ANSSI, which sponsored the evaluation and wrote the target of evaluation, Quarkslab, the organization

This work benefited from funding managed by the French National Research Agency under the France 2030 programme with the reference ANR-22-PECY-0006.

responsible for the evaluation, and the certification center of the ANSSI (i.e., *CCN*) for the coordination and the conclusion on the certification.

The paper discusses the limits of the CSPN framework and the currently existing solutions to evaluate the security of an e-voting solution, using the example of Belenios, which is presented in Section 2. A state of the art of the security of e-voting solutions is presented in Section 3. The CSPN framework, which we present in Section 4, offers a lot of possibilities but can easily lead to a too-restrictive perimeter and threat model. It is not a desirable way to model the reality, especially for a voting solution that could potentially have a massive impact if used at the national scale. We illustrate this in Section 5 by detailing how we evaluated Belenios through the CSPN prism. Finally, Section 6 presents our arguments in favor of pursuing this discussion further.

2 Belenios and E-Voting

Belenios [17] is an Internet voting system that aims to provide vote secrecy and verifiability properties. Inspired by the Helios voting system [1], its development started a decade ago. It is a public online platform where anyone can set up their own election, and has been in place for more than 5 years. New features are regularly added, and the software is still maintained and developed.[3] The software is published under the GNU AGPLv3 license, and a precise specification is given so that it is possible to write independent software that will be compatible.

Cryptographic tools. The cryptographic primitives used in Belenios are very classical. The ElGamal encryption scheme in a finite field is used for ballot encryption. The chosen hash function is SHA256, PBKDF2 is used for key derivation, and the signature algorithm is Schnorr's.

ElGamal ciphertexts enjoy a homomorphic property that Belenios uses: multiplying two ciphertexts, one gets another valid ciphertext that encrypts the sum of the two plaintexts. In many places, it is required to prove some statements about ciphertexts, without revealing a secret, and for this Belenios uses zero-knowledge proofs *à la* Chaum-Pedersen [15], that are specific to the discrete logarithm setting, and therefore very efficient.

To prevent security from depending on a single entity, the decryption key is divided into multiple segments. To ensure resilience in case one of

[3] The description given here is for the version 1.19, which is the one that was evaluated; the current version at the time of writing is 2.4.

the decryption key holders becomes unavailable or declines to participate in the decryption process, a *threshold* decryption scheme is implemented.

Participants. The protocol relies on several entities (humans assisted by their computing device, or servers assisted by their administrator). First, the *Voters*, whose opinions are requested for the given election. They vote by Internet, using their computer or their smartphone. For this, they connect to the *Server*. The same server is where the *Election administrator* connects to setup and orchestrate the election. An entity called the *Credential authority* (a.k.a. Registrar) is in charge of sending voting material to the voters. Some *Decryption trustees* hold parts of the key required to tally the election.

Finally, an *Auditor* role is defined, that consists in checking regularly that the server behaves as expected. In particular, they check that the *Public bulletin board* is consistent. Any of the other entities can also play the role of auditors.

Protocol description. The protocol is split into 3 phases: setup, vote, tally; the election administrator connects to the server, to end a phase and start the next one.

The Setup phase. The election administrator connects to the server and provides the information concerning the election: title, questions, possible answers, and list of voters. They also send a link to the credential authority. From this link, the credential authority gets the list of voters and generates the credentials, that are, for each voter, a pair of keys, one (private) signing key, and one (public) verification key. The credential authority sends the private parts to the voters and the public parts to the server. These public parts are added to the public board by the server.

The decryption trustees collaboratively generate an ElGamal encryption key using a 3-round protocol, so that the global decryption key is never present on a single computer. The server is also involved in this key generation process, in order to schedule the (encrypted) communications between the trustees. The resulting public encryption key is then incorporated into the public board by the server.

The Vote phase. When the election is open, voters can connect to the server and receive a Javascript voting client that does all the ballot preparation on the browser side. The voter has to give their credential and select the answers to the questions; their browser can then prepare

their ballot, encrypt it, and sign it with the credential. More precisely, the ballot contains a list of encrypted bits, one for each answer, telling whether this answer was chosen by the voter.

The ballot is sent to the server that authenticates the voter (typically via a challenge by e-mail), checks the validity of the signature and the well-formedness of the ballot (including verification of the zero-knowledge proofs), and appends it to the public ballot box.

The Tally phase. The server homomorphically agglomerates the ballots present in the ballot box, to obtain, for each answer, the encryption of the sums of the 0 or 1 values in each ballot, that is the encryption of the number of votes for this answer.

Auditors (and decryption trustees) check that everything is in order, and then the decryption trustees collectively decrypt the score of each candidate, together with a zero-knowledge proof of correct decryption. Again, the global decryption key is never present on a single computer.

The server checks the proofs and publishes the result (together with the proofs).

Security claims. An important idea of Belenios's security is that if the attacker controls only one entity, they will not be able to break a security property. This includes in particular the server: if the server is the victim of an (internal or external) attack, it should not be able to change the result of the election, nor to break the privacy of voters.

Up to now, the security of Belenios has been mostly analyzed as a protocol, and not really as a software. Furthermore, the threat model and the security properties studied in the academic literature are rather different from what is analyzed in a CSPN: the properties are usually global properties, and the threat model (the assumptions under which the property holds) may differ from one property to another.

The fact that the result corresponds exactly to the voter's choices is usually split into 4 sub-properties: cast-as-intended, individual and universal verifiability, and eligibility. We detail the meaning of these properties in Section 5.2.

The other important property is *vote secrecy*. In Belenios, this is ensured in two ways. First, individual ballots are never decrypted, therefore, vote privacy does not rely on the fact that the link between voters and ballots does not exist. Thanks to the homomorphic property, the decryption trustees decrypt only the result. Second, no single entity holds the full

decryption key. Therefore, privacy holds as soon as the number of dishonest trustees is below the chosen threshold.

Belenios does not protect voters against coercion or vote buying.

Finally, when it comes to resistance against a dishonest server, let us insist on the fact that the security strongly relies on the auditors who must check that the ballot box remains consistent and that the Javascript served by the server to various entities is the legitimate one.

3 State of the art

Academic perspectives. E-voting protocols proposed in academia are usually evaluated through security theorems and proofs, that clearly state the properties that are achieved, and for which specific threat models and security assumptions are defined. In particular, these security proofs analyze properties such as vote privacy and verifiability. Two main models exist in the literature to analyze security protocols, namely formal (or symbolic) models and computational (or cryptographic) models. Formal models consider a high level of abstraction for the cryptographic primitives and study the logical flow of the protocol, for arbitrary symbolic adversaries. Such models support a high level of automation, hence several tools have been developed, such as ProVerif [6] or Tamarin [23], that can automatically find attacks or provide a security proof. In contrast, computational models consider more precisely the cryptographic primitives, such as homomorphic encryption or zero-knowledge proofs. The attacker is any polynomial probabilistic Turing machine. They can be used to detect flawed interactions between the cryptography and the protocol flow. However, computational proofs must usually be conducted by hand, although some tools also exist such as CryptoVerif [7] and EasyCrypt [5], that require interactions with the user.

Belenios has been proven to preserve vote privacy and to offer individual, universal, and eligibility verifiability in a computational model [16]. A machine checked proof has also been conducted in EasyCrypt [14]. Proofs in symbolic models, using ProVerif, can also be found in [11, 13].

These security proofs allow analyzing vote privacy and verifiability on e-voting protocols. However, they only consider the protocol "on paper". Such analyses do not cover the implementation or the deployment of an e-voting system.

CNIL recommendations. The CNIL (Commission Nationale de l'Informatique et des Libertés) is the main institution in France that

provides guidelines for the organization of Internet voting. Most election organizers wish to comply with the CNIL recommendations.[4] Its last version [25] has been published in 2019 and defines three security levels: from 1, the lowest security level, to 3, the highest security level. Typically, elections of security level 1 are for low-stake, small elections (like parent representatives in a school), while elections of level 3 stand for large professional elections, in a conflictual context. The CNIL requires vote privacy from level 1 and introduces verifiability at level 2, with the use of third-party tools at level 3. However, many recommendations are organizational and there is no clear trust model. For example, it remains unclear whether the voting server should be trusted, and for which properties. Moreover, the CNIL does not make any recommendation w.r.t. publishing the specification of the protocol, while an e-voting system without public specification does not offer verifiability.

It should be noted that the CNIL advises against the use of Internet voting for political elections. Consistently, the recommendations do not apply to high-stake elections and rather focus on professional and associative elections.

The Swiss example. The Swiss Chancellery [12] has probably the most demanding regulation. It is specific to the Swiss context, for political elections. For example, voters are supposed to receive their material by postal mail and the main flow of the protocol is fixed. The requirements cover many aspects. They precisely describe the threat model. The voting server is untrusted for all properties, and the trust is split among 4 control components that are online during the voting phase (only 1 out of 4 is trusted). The voting device is trusted for privacy but untrusted for verifiability: it may try to alter the voter's vote. On the other hand, the component in charge of generating the printed material is fully trusted, as well as the postal services in charge of delivering the material to voters. The Swiss Chancellery requires vote privacy as well as end-to-end verifiability, which includes cast-as-intended: a voter should be able to detect if their voting device attempted to modify their vote. External auditors should be able to check the tally.

The regulation relies a lot on *public scrutiny*: the detailed specifications, as well as the code, must be published. A generous bug bounty program rewards a wide variety of potential bugs. Moreover, a public intrusion test is regularly organized, again with a bug bounty program. Finally, this

[4] The CNIL is the data protection agency, and it is under this authority that it has written the e-voting recommendations.

regulation also asks for symbolic and computational proofs of the security of these properties, following academic standards.

Thanks to these regulations, non-trivial cryptographic flaws were discovered in 2019 [20] that could allow election authorities to alter the result without being detected, if they all collude (a threat model way beyond usual trust assumptions in other countries). After this attack, Internet voting was suspended until 2023.

Other countries. Estonia has used e-voting for politically binding elections for a decade. The system in use is public [22] as well as its code (more or less updated). Severe flaws are still regularly discovered [24]. Similarly, for the 2015 elections in New South Wales in Australia, a high-level description of the system is provided [8] but again, severe flaws were discovered [21].

4 What is a CSPN evaluation?

Created in 2008 by the ANSSI, the First Level Security Certification (CSPN [5]) is the French Fixed-Time Cybersecurity Certification. It assesses the product's resistance against moderate attacks. Conducted by an IT Security Evaluation Facility (ITSEF) certified by the ANSSI, this certification is a cheaper and quicker alternative to the *Common Criteria* certification process.

An evaluation model that tends to be a new standard in Europe. The CSPN evaluation is conducted in a constrained amount of time, usually 35 days (with cryptography included), in an environment as close as possible to the actual usage of the product by its different users (administrator, main user, maintenance...). The goal is dual: analyze the product's conformity with respect to its security specifications, and evaluate the strength of the security functions.

This fixed-time evaluation scheme is increasingly adopted by national security agencies as a standardized process for assessing the basic security of products. For example, the German Federal Office for Information Security (BSI) created a similar certification process in 2021: *Beschleunigte Sicherheitszertifizierung* (BSZ) [9]. The BSI and the ANSSI have established an agreement for the mutual recognition of certificates of the CSPN and the BSZ [4]. At the "continental" level, the EU has published a new

[5] CSPN stands for *Certification de Sécurité Premier Niveau* in French.

methodology for fixed-time cybersecurity evaluation [10] to standardize these national processes into a uniform one.

CSPN Evaluation: a three-step workflow. The sponsor provides a security target specifying the product's security features and its execution environment. The ANSSI reviews this document and accompanying administrative papers to accept or reject the CSPN request. Then, the ITSEF conducts its audit and provides an Evaluation Technical Report (ETR) with the evaluators' opinions and recommendations. Based on the ETR, the ANSSI decides on the certification output and publishes the certificate upon validation [2]. The complete process is detailed in Figure 1.

The Security Target: a key point for the certificate legitimacy. The Security Target is a document that defines the audit perimeter. We can divide this document into four parts that contain precise elements required by the procedure [2].

The *Product Description* should include the product's exact version, configuration, and the technical environment in which it should be installed. It should detail the product's usage (or pinpoint to the appropriate documentation) throughout its life-cycle, including a presentation of the different users and the intended contexts.

The *Security Perimeter* details the product's architecture and specifies what is included.

The *Security Audit* section regroups:

— The hypotheses made for the audit.
— The list of critical assets and their respective security needs (integrity, confidentiality, authenticity, and availability).
— The threat actors considered for the audit as well as the definition of the threats.
— The list of security functions.

Finally, if the product embarks some cryptography, it should also be evaluated. In that case, the sponsor should provide a cryptographic specification document entitled *Cryptography Mechanisms* listing:

— The algorithms, modes of operation, and objectives.
— The protocols and their functionalities.
— Key generation, transmission, and storage.
— Random generation (sources and handling).

Usability, Conformity, Security: the audit focuses The audit should verify the different following points, that will be recorded in the ETR [3].

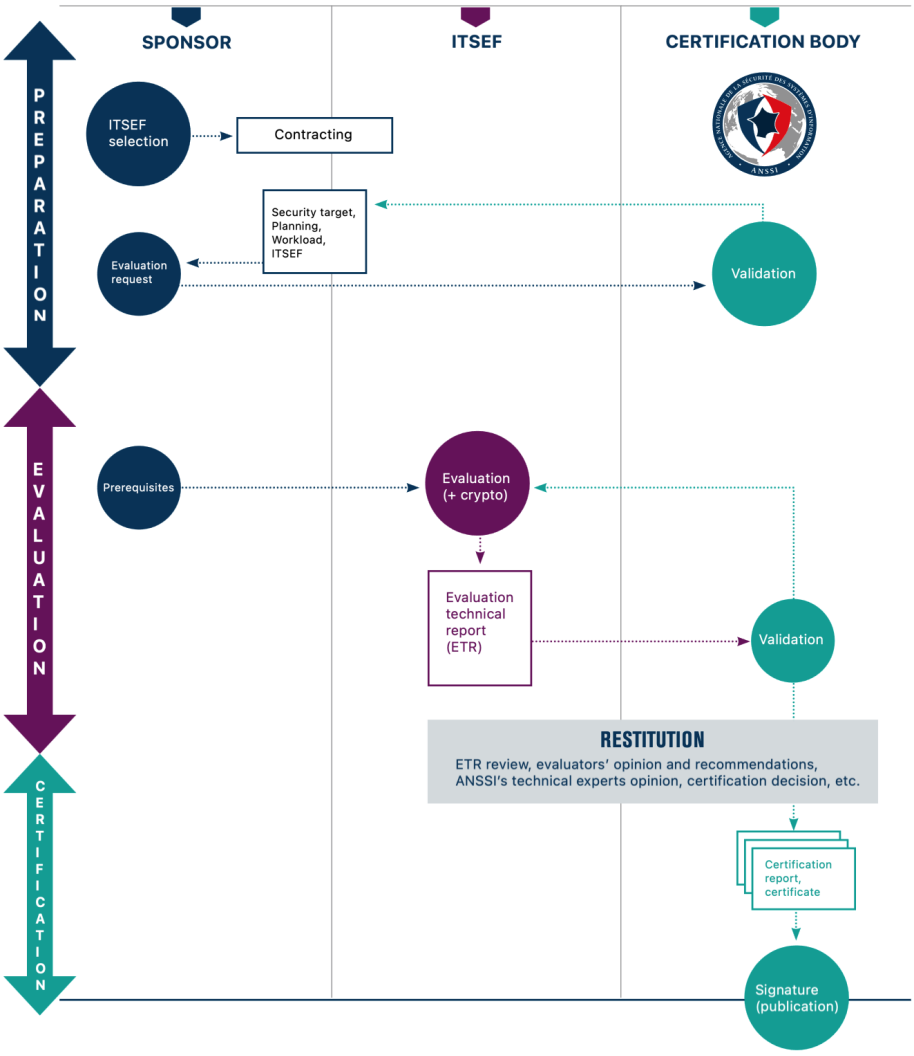

Fig. 1. The CSPN Evaluation Workflow.

First, the *Usability* is evaluated, either during the product installation or usage, considering each user type's point of view. The evaluators will also assess if any information is missing in the installation/usage documentation that could lead to a security issue.

Then, the *Conformity* of the documentation, the source code (if available), and the installed product are evaluated. Each security function is tested to check if its behavior corresponds to the specification. On the cryptography side, the experts will check the conformity of the implemented mechanisms concerning the documentation and the ANSSI guides. More generally, the experts will assess that the security functions use state-of-the-art mechanisms, both for the primitives in isolation and with respect to other similar products.

Finally, the *Security* is evaluated by testing the strength of the security functions and the vulnerabilities found by the auditors during their dynamic and source code reviews. They will focus on trying to obtain an asset, or to break a security function.

All of these results are recorded in the ETR along with the aforementioned expert opinion (as a conclusion) that will be used by the ANSSI to determine if the certification is passed.

5 Journey to certification

5.1 Belenios as a target.

In this section, we will quickly summarize how Belenios is modeled in the target document. We focus on the *security functions* and will omit details in other parts that are less relevant to the point of this article.

The functions concern voters' privacy, confidentiality (before publication) of the result, as well as its integrity and verifiability. Additional security functions were introduced by Quarkslab, related to the web interfaces.

Vote Privacy and Result Confidentiality (1) votes are encrypted, (2) the decryption key is shared between trustees and never reconstructed, (3) the voting client can be recompiled and verified independently.

Result Integrity (1) ballots are signed, (2) voters can verify the handling of their ballots, (3) tally can be independently verified, (4) only valid votes are tallied, (5) from only registered voters, (6) complete verifications are done by auditors.

Web Interfaces (1) proper session management, (2) proper authentication management, (3) input validation and sanitization.

The threats that these functions are meant to thwart are rather straightforwardly deduced (e.g., votes are encrypted so that they are not linked and leaked, or signed so that they cannot be forged, etc.). As for the potential attackers, we consider external attackers, i.e., someone who is not part of the election and only has access to the public information, as well as legitimate voters, and, to be fully realistic, election and server administrators. This last part is crucial in most studied products yet often omitted, and in this specific case, since e-voting protocols are already designed to be as close to "real-life" voting as possible, it is entirely probable that voting authorities are corrupt.

The target does, however, have hypotheses on what is seen as secure and/or trusted: (1) the registration authority correctly generates and sends the credentials, which are also securely stored by the voters, (2) every party's browsers are trusted, and the election was properly installed and setup, (3) the *threshold* of honest trustees is met, and there is at least one honest auditor.

5.2 A well-studied protocol.

As explained in Section 4, the CSPN methodology includes specific steps when a product contains enough cryptography for it to be studied in detail. We can see in Section 2 that it was indeed the case for Belenios, as cryptographic mechanisms play a crucial part in ensuring the security properties are met.

For the evaluation, the document that was submitted to be considered as *Cryptographic Mechanisms* was a shortened version of Belenios's specification [19], excluding options and features that were not part of the considered configuration. In comparison with other products that went through the CSPN process, we can already see that the presence of this extensive crypto-oriented documentation shows the designers' understanding of the domain.

Moreover, as early as in its introductory paper [17], and as mentioned in Section 3, Belenios has been proven to meet a variety of security features, which we detail below. We can see that the various properties proven in those articles are similar to the target's security functions in Section 5.1, though with more nuance.

— *Cast-as-intended*: The encrypted ballot that is sent to the server contains the voter's intention. This is not guaranteed by Belenios.

No provision is taken for the case where the voter's device behaves badly, for instance, because it is infected by a malware.

— *Individual verifiability*: The voter can check that their (encrypted) ballot is present in the (public) ballot box. In Belenios, this is enforced by the voting client showing the hash of the ballot, the presence of which can be checked on the web page serving the ballot box.

— *Eligibility*: Anyone can check that the ballot box contains only ballots coming from legitimate voters. In Belenios, this is true unless the attacker controls both the server and the credential authority. Voters are authenticated dynamically, during the voting phase, by the server, and their ballots are authenticated offline, thanks to the signature with their credentials.

— *Universal verifiability*: Anyone can check that the claimed result corresponds to the list of encrypted ballots present in the ballot box. For this property, zero-knowledge proofs are added: the voter proves that their ballot is well formed, and the decryption trustees provide proof of correct decryption.

The proofs referenced in that article can be found in another paper [14], they were conducted with EasyCrypt, an interactive theorem prover that can be used to write security proofs in the computational model. Another tool was also frequently used [11, 13] to prove specific features in Belenios: ProVerif, an automatic verifier in the formal (Dolev-Yao) model.

We also note that the aforementioned articles discuss the need for more generic frameworks and properties so as not to have one specific model for each voting protocol. We will discuss a similar issue in the Section 6.

The cryptographic evaluation. The existing proofs mentioned above meant that we could focus on conformity (with the ANSSI documentation and with the state of the art) and implementation (choices of libraries, verify that it does what the documentation says, how randomness is handled, etc.), and less on the protocols.

Indeed, the two vulnerabilities related to cryptography that were found during the initial evaluation, and patched afterwards, were of the nonconformity category (too few PBKDF2 iterations with respect to the OWASP recommendation, and a too-short salt according to the ANSSI guides). These vulnerabilities were not exploitable, as other elements of the protocol meant that the "attacks" would take more than 6 months (the highest limit in evaluating the *elapsed time* in a CSPN).

For all other aspects that must be studied in the cryptographic evaluation, we found Belenios to be satisfactory: the primitives used are adequate, and as for the implementation, the chosen of external libraries are secure, and randomness is correctly generated and handled.

5.3 From theory to practice.

Then came the evaluation based solely on the target of everything that was not cryptography-related. During the evaluation, it was necessary to proceed in several steps, as described in Section 4: analyzing the documentation and using the product, then going deeper by analyzing the source code and imagining and testing attack scenarios.

Evaluation perimeter extension. The original security target of Belenios is mostly centered around assets and functions inspired by the academic articles, and thus mostly relating to the cryptographic aspects. However, the server used to install Belenios and the Web interface are important components to analyze. Therefore, the evaluators believe that this part should be properly included in the evaluation, and tested.

Although, even if you can prove that the votes have not been altered, it will remain difficult for a non-technical audience to understand that a web interface modification by an attacker does not mean that the ballot box has been compromised. That's why it's important to ensure that the various functionalities offered by the web part of the solution are sufficiently robust to avoid compromising the application and potentially the server.

To that end, the evaluation was broadened (by the evaluators) to include, within the restricted timeline, the following:

— The verification that potential security functions related to the Web component of the TOE are (1) following security best practices and (2) cannot be bypassed easily.
— The verification that threats related to the Web component of the TOE cannot degrade the existing security functions or existing assets.

These considerations highlight the need for both a theoretical and a fully practical side to the evaluation.

After testing Belenios and the functionalities offered by its web interface, the evaluators identified several new security functions based on traditional Web best-practices like providing robust cookies for election administrator, or having brute-force resistant authentication forms.

Extensive target evaluation. The difficulty here was in particular trying to identify attack vectors that were not foreseen in the target and trying to be exhaustive to cover all the functionalities proposed. Overall, it was necessary to ensure that a vulnerability in the web interface did not impact service availability during an election, data confidentiality (votes), or data integrity (manipulation of votes and election results). The major risk is that an election could be invalidated.

Several attack scenarios can then be developed. They could be achieved by compromising the server, the web interface, the voter's machine, the browser, or the network. On the voter's machine and the browser, the security target stated that such attacks were out of scope. However, it was found that Belenios does not implement any communication encryption methods. The administrator must set up a sufficiently secure TLS configuration to protect against various attacks affecting the integrity or confidentiality of communications.

In another example, the security target mentions that availability was not to be considered, as an external actor could conduct Distributed Denial of Service network attacks aimed at making an election inaccessible. The Belenios application has no method of preventing such attacks, and a specific architecture must be set up if this is desired for a given election. However, no consideration has been given to the case where an application function alone could generate a Denial Of Service. This was notably the case with a possible denial of service by uploading documents to the platform without needing to be authenticated, allowing disk space to be saturated.

This is always a risk in these types of evaluations: formulating hypotheses that exclude more scenarios than what was aimed for. It is important to be thorough and as close to reality as possible.

Third-parties. Belenios relies on an external framework, Eliom, for the Web part. As it is a third party, this framework was not integrated into the security target, and vulnerabilities may exist that could affect it. We can see there is another important point to keep in mind when discussing certifications like CSPNs, only the product itself is certified not its dependencies. This is precisely why, in an evaluation, all external libraries/frameworks used by the solution must be clearly defined, and a process for updating and monitoring vulnerabilities must be put in place.

Outcome. While Belenios was ready for a CSPN based on the original security target, the evaluation experts found, as explained at the beginning

of this section, that the scope of security target required to be enlarged, which precluded the certification. Redoing another evaluation from scratch based on the refined security target was decided to be the way to go.

6 Takeaway

Evaluating e-voting solutions. Throughout the previous sections, we have seen that evaluating an e-voting solution using a generic framework requires some flexibility and reinterpretations to make said solution look more like other products normally evaluated through a CSPN. Even the CNIL recommendations for e-voting protocols, which are (obviously) specific are not satisfactory, as explained in Section 3. We are also aware that the security-oriented public is generally convinced that e-voting should not be used for major elections – but they are, and that means we *have* to ensure that they are as secure as possible, and for that, we need the appropriate framework.

We have also seen that, as always, it is imperative to study both the theoretical and practical aspects of e-voting, and in this specific case, we have seen that the theoretical part is several steps ahead. Indeed, the properties proven in academic papers are nuanced, precise, and capture most facets of e-voting *protocols*.

Moreover, the attackers found in e-voting have different capabilities and incentives compared to most products: ultimately, we might consider that "passive" attackers could want to either know the result of an election before the end or know how a specific user voted, and active attackers could want to change the result of an election (either directly at the end or by changing the vote of a (group of) user(s)). Although this problem is somewhat similar to attackers wanting to read/modify the content of encrypted messages or payloads, the ballots' treatment and handling before being tallied are not comparable to anything else.

What do we want? Ultimately, our wish would be to have a specific framework for evaluating and certifying the security of e-voting solutions. As we have seen, many audits are carried out once the solutions are in use (or afterwards), including for major elections like for the French Legislative E-Voting Protocol, for which two vulnerabilities were uncovered [18]. In the article presenting those flaws, the authors had to reverse the voting client to obtain a full specification as only a partial one was available. This also raises the question of openness: in a Belenios election, anyone has the possibility to become an auditor ; this principle should be applied

to e-voting solutions as a whole. We agree with the authors of [18] that transparency is very important (as seen with the reliance on public scrutiny in Switzerland, explained in Section 3) both in terms of the code itself and of what threats and attackers are considered in the model.

Perspectives. We have many tools at our disposal that we can enhance to build a satisfactory framework. Discussions must be held with participants of the e-voting community and anyone concerned with this issue, to find a way to improve upon, for example, the CNIL recommendations and CSPN methodology *specifically* for e-voting. We have seen that state actors are more than happy to participate in these discussions.

We also know that there is a rich pool of academic articles on the subject that can have a strong focus on the theoretical aspect *and* a comprehensive examination of the implementations.

Though it may take some time, we believe that it is a necessary subject to tackle on a larger scale; it is no longer possible to simply argue that it is "not secure enough" and dismiss the topic entirely.

Acknowledgement

We would like to thank the ANSSI for sponsoring and accompanying both of our teams in this certification and for being open to further discussions on the matter.

References

1. Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 335–348. USENIX Association, 2008.
2. Agence nationale de la sécurité des systèmes d'information. Certification de sécurité de premier niveau des produits des technologies de l'information.
3. Agence nationale de la sécurité des systèmes d'information. Méthodologie d'évaluation en vue d'une CSPN - contenu et structure du RTE.
4. Agence nationale de la sécurité des systèmes d'information and Bundesamt für Sicherheit in der Informationstechnik. Mutual Recognition Agreement of Cybersecurity Evaluation Certificates issued under a Fixed-time Certification Process.
5. Gilles Barthe, Francois Dupressoir, Benjamin Grégoire, Cesar Kunz, Benedikt Schmidt, and Pierre-Yves Strub. EasyCrypt: A Tutorial. In *Foundations of Security Analysis and Design - FOSAD 2013*, 2013.
6. Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135, 2016.

7. Bruno Blanchet and Charlie Jacomme. CryptoVerif: a Computationally-Sound Security Protocol Verifier. Technical Report RR-9526, 2023.
8. Ian Brightwell, Jordi Cucurull, David Galindo, and Sandra Guasch. An overview of the iVote 2015 voting system. New South Wales Electoral Commission, Australia, 2015.
9. Bundesamt für Sicherheit in der Informationstechnik. Beschleunigte Sicherheitszertifizierung (BSZ)—a fixed-time cyber security certification scheme.
10. CEN-CENELEC. New EN 17640 "Fixed-time cybersecurity evaluation methodology for ICT products" helps evaluate the cybersecurity of ICT products.
11. Vincent Cheval, Véronique Cortier, and Alexandre Debant. Election Verifiability with ProVerif. In *CSF 2023 - 36th IEEE Computer Security Foundations Symposium*, Dubrovnik, Croatia, July 2023.
12. Swiss Confederation. Technical and administrative requirements for electronic vote casting. Annex to the FCh Ordinance of 13 December 2013 on Electronic Voting (OEV, SR 161.116), 2018.
13. Véronique Cortier, Alexandre Debant, Pierrick Gaudry, and Stéphane Glondu. Belenios with cast as intended. In *Financial Cryptography and Data Security. FC 2023 International Workshops - Voting, CoDecFin, DeFi, WTSC, Bol, Brač, Croatia, May 5, 2023, Revised Selected Papers*, volume 13953 of *Lecture Notes in Computer Science*, 2023.
14. Véronique Cortier, Constantin Catalin Dragan, François Dupressoir, and Bogdan Warinschi. Machine-checked proofs for electronic voting: Privacy and verifiability for Belenios. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, 2018.
15. Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Distributed elgamal à la pedersen: Application to helios. In Ahmad-Reza Sadeghi and Sara Foresti, editors, *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, pages 131–142. ACM, 2013.
16. Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. Election verifiability for Helios under weaker trust assumptions. In *Proceedings of the 19th European Symposium on Research in Computer Security (ESORICS'14)*, LNCS. Springer, 2014.
17. Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Belenios: A simple private and verifiable electronic voting system. In *Foundations of Security, Protocols, and Equational Reasoning*, volume 11565 of *Lecture Notes in Computer Science*, pages 214–238. Springer, 2019.
18. Alexandre Debant and Lucca Hirschi. Reversing, breaking, and fixing the French legislative election e-voting protocol. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, 2023.
19. Stéphane Glondu. Belenios specification. Available from `https://www.belenios.org/specification.pdf`.
20. Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. In *IEEE Symposium on Security and Privacy (SP'20)*, pages 644–660. IEEE, 2020.

21. J. Alex Halderman and Vanessa Teague. The New South Wales iVote system: Security failures and verification flaws in a live online election. In *E-Voting and Identity - 5th International Conference, VoteID 2015*, LNCS. Springer, 2015.

22. Sven Heiberg, Tarvi Martens, Priit Vinkel, and Jan Willemson. Improving the verifiability of the Estonian internet voting scheme. In *Electronic Voting (E-Vote-ID'16)*, pages 92–107. Springer, 2017.

23. S. Meier, B. Schmidt, C. Cremers, and D. Basin. The Tamarin Prover for the Symbolic Analysis of Security Protocols. In *25th International Conference on Computer Aided Verification (CAV'13)*, volume 8044 of *LNCS*, pages 696–701. Springer, 2013.

24. Johannes Mueller. Breaking and fixing vote privacy of the Estonian e-voting protocol IVXV. In *Workshop on Advances in Secure Electronic Voting*, 2022.

25. Journal Officiel. Délibération n° 2019-053 du 25 avril 2019, 2019. Available at the link `https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000038661239`.