Reverse Bayesian poisoning: How to use spam filters to manipulate online elections

Hugo Jonker[1,2], Sjouke Mauw[3], and Tom Schmitz[3]

[1] Department of Computer Science,
Open University of the Netherlands, Heerlen, Netherlands
hugo.jonker@ou.nl, http://www.open.ou.nl/hjo/
[2] Digital Security group, Radboud University, Nijmegen, Netherlands
[3] CSC/SnT, University of Luxembourg, Luxembourg, Luxembourg
sjouke.mauw@uni.lu, http://satoss.uni.lu/sjouke/

Abstract. E-voting literature has long recognised the threat of denial-of-service attacks: as attacks that (partially) disrupt the services needed to run the voting system. Such attacks violate availability. Thankfully, they are typically easily detected. We identify and investigate a denial-of-service attack on a voter's spam filters, which is not so easily detected: *reverse Bayesian poisoning*, an attack that lets the attacker silently suppress mails from the voting system. Reverse Bayesian poisoning can disenfranchise voters in voting systems which rely on emails for essential communication (such as voter invitation or credential distribution). The attacker stealthily trains the voter's spam filter by sending spam mails crafted to include keywords from genuine mails from the voting system. To test the potential effect of reverse Bayesian poisoning, we took keywords from the Helios voting system's email templates and poisoned the Bogofilter spam filter using these keywords. Then we tested how genuine Helios mails are classified. Our experiments show that reverse Bayesian poisoning can easily suppress genuine emails from the Helios voting system.

1 Introduction

System security is typically divided into Confidentiality, Integrity and Availability. Voting systems present an interesting research challenge where confidentiality of the vote (privacy) and verification of integrity of the vote (verifiability) must be combined in one system. As these two requirements cannot be fully satisfied simultaneously [CFS+06], their interplay thus poses an interesting topic for research. Much research attention has been invested in addressing the interplay of confidentiality and integrity in voting. This has led to the development of various notions of verifiability, e.g. [Cha04,JCJ05,CRS05], and privacy [Cha81,BT94,JCJ05]. The efforts towards formalising verifiability culminated in the concept of *end-to-end verifiability*. The name derives from the fact that the voter can verify every step from the input of her choice up to and including the counting of her ballot. As such, end-to-end verifiability ensures verifiability for a voter who casts a vote.

Availability requirements have also been considered: for more than a decade, denial-of-service (DoS) attacks have been considered a serious threat to e-voting systems. Various authors have argued including this class of attacks in the security analysis of e-voting systems [GNR+02,Rub02,JRSW04,Agg16]. Until now, however, DoS attacks have mainly been considered from a generic point of view: a DoS attack is seen as disrupting (some of) the services that are necessary for an e-voting system to operate. The occurrence of such attacks can relatively easily be detected by one or more of the involved parties. It can be expected that the goal of such a DoS attacker is mainly to disrupt the election process and not to directly influence its outcome. Hence, literature views DoS attacks as attacking *availability* of the system and not *integrity* of the election.

In this view on availability, either an attack on availability is noted (and action is taken to rectify this), or every voter who wants to vote, is not impeded by the system from doing so. In the latter case, end-to-end verifiability ensures that the result of the election process matches the collective intent of the voters.

In this paper, we will explore one DoS-related vulnerability that will influence the result without disrupting general availability of the system: using spam filters to suppress voter invitation, thereby impeding voters to vote.

Though various sources have different estimates for the current global amount of spam email, they agree that it is above 50%. As such, spam filters have become a necessity – most incoming emails will be filtered for spam. We investigate the extent to which an attacker can exploit this. We find a novel attack on voting systems: the attacker trains a voter's spam filter to ensure that legitimate emails from the voting system are marked as spam and thus likely remain unseen by the voter. This borrows from the notion of *Bayesian poisoning*, in which an attacker sends emails to ensure that his spam is not detected by the target's filter. Since our attacker seeks the reverse, we call this type of attack *reverse Bayesian poisoning*.

In contrast to DoS attacks as considered in e-voting literature, a reverse Bayesian poisoning attack is stealthy and targeted. It is stealthy in the sense that it is not obvious that the system is under attack, and after the attack it may not even be possible to prove the system has been attacked. It is targeted in the sense that it does not attempt to exceed system capacity, but it is an attack on a particular essential service that maliciously alters that service's behaviour. As this change of behaviour is under the control of the attacker, he can manipulate the outcome of the elections. Thus, this type of DoS attacker focusses on the *integrity* of the elections.

Consequently, the question whether this type of stealthy and targeted attacks are possible in e-voting systems becomes a concern that goes beyond the generic problem of service availability and influences the integrity of the election process.

For the purposes of this paper, we investigate whether the well-known Helios voting system [Adi08] is susceptible to reverse Bayesian poisoning. Helios is an online election system that provides end-to-end verifiability. It offers email facilities for communicating election particulars to voters. The Helios voting system

was used in a number of elections, such as by the International Association for Cryptologic Research for electing board members and directors.

Contributions. In this paper:

- We argue that e-voting security also depends on availability of supporting procedures and systems. We identify and investigate one example of a vulnerability in such a supporting system: email interaction between the voting system and the voter.
- We demonstrate that it is possible and even fairly easy to suppress relevant emails from a voting system by means of a reverse Bayesian poisoning attack. In particular, we are able to manipulate the BogoFilter spam filter such that it suppresses emails from the Helios voting system.
- Finally, we discuss several possible solution directions.

2 Notions of election security

An election has to satisfy a wide range of security and privacy requirements. In literature and in practical systems, the focus tends to be on preventing misuse of the system. As such, there are requirements to prevent double voting and to allow a voter to verify that her vote counts in favor of the candidate for whom she submitted a ballot. Little attention is paid to the setup of the election: determining who is eligible, and distributing voting credentials to authorized voters.

This lack of attention to election setup is natural for voting systems targeting supervised voting (i.e. a setup with a polling station). In supervised voting, such security requirements are supposed to be guaranteed by the supervision. However, online voting systems have no such infrastructure to fall back upon and must therefore address security and privacy aspects of this part as well.

To determine the limits of the scope of end-to-end verifiability, we divide the election into the below administrative processes. These processes are placed into the first election phase where they can be executed. Remark that some processes can be postponed. For example, voter eligibility can be checked before, during or after the election.

- **Pre-election phase.** Processes typically executed in this phase are:
  - **Voter registration.** For some elections, voters need to register themselves. For other elections (e.g. in associations), the register is pre-existing (e.g. the membership list). This process establishes the voter register.
  - **Voter eligibility.** Not all those who are registered are eligible. For example, general elections typically only allow people above a certain age to vote. This process eliminates all people from the register who are found to not satisfy the rules of eligibility.
  - **Announcement and credential distribution.** To cast a vote, voters typically have to show some form of voter credentials. This process distributes the announcement of the election and voting credentials to all voters.

– **Election phase.** Processes typically executed in this phase are:
  - **Vote casting.** In this process, the voters communicate the expression of their choice to the voting system.
  - **Vote eligibility.** The votes received by the system are checked for eligibility, to ensure that only those votes that should be taken into account are considered.
– **Post-election phase.** Processes typically executed in this phase are:
  - **Vote aggregation.** In some elections, the encoded ballots are aggregated before opening them (e.g. systems using homomorphic encryption). In other elections, this process is omitted.
  - **Result determination.** In this process, the result of the election is determined. This can be an exact count per candidate (or option), or a simple determination of which candidate(s) (or options) won.
  - **Result announcement.** In this process, the election result is communicated to the voters and (when applicable) the world at large.

Over the last two decades, research in e-voting security has steadily worked towards designing practically usable voting systems that satisfy privacy and security requirements. Since elections have already been organised before, researchers sought to update parts of the existing process with new techniques. In particular, research was focused primarily confidentiality and integrity of the election and post-election phases. This was highly successful and led to a better understanding of the security and privacy principles involved as well as the design and development of systems that safeguard most if not all of those principles.

Availability concerns are recognised as generic concerns, but so far not as a specific risk for online election systems. However, online election systems also depend on the internet for the pre-election phase. Availability attacks on online election systems may also target the pre-election phase. To highlight the necessity for considering this part of the election process, we explore one stealthy attack upon the announcement and credential distribution process: abusing spam filtering to suppress emails sent by the voting system.

3 Spam, filtering and poisoning

3.1 Spam filtering

A significant fraction of mail traffic is spam – over half, by most reports[4]. Therefore, a thorough spam filter is a necessity for any email address.

Spam filters are filters that distinguish between non-spam emails and spam emails. The output of the filter typically is a classification (*non-spam*, *spam*, or *unknown*) plus a score specifying the extent to which the evaluated mail satisfied filter rules for spam. This output is then used to (automatically or not) delete

[4] E.g. Kaspersky's quarterly spam reports, https://securelist.com/all/?category=442, pegs the amount of spam in email traffic in the first three months of 2017 at 55.19%.

mails classified as spam or divert such mails to a dedicated spam folder. Mails that end up in the category *unknown* can be flagged as such and presented to the user for further processing.

Spam filters use various tests to arrive at their result. Each test evaluates the mail and arrives at its own estimate for the probability of the examined mail being spam. These probabilities are then aggregated by the spam filter to classify the mail.

Spam filters classify emails using two thresholds: a non-spam and a spam threshold. A probability below the non-spam threshold ensures a mail is classified as non-spam. A probability above the (higher) spam threshold ensures that a mail is classified as spam. Probabilities in between are not classified – tagging the message as potential spam, while letting the mail through to the user's inbox.

There is a wide variety in tests spam filters use to determine the probability that a mail is spam, and any given spam filter will use its own subset of tests. Some of these tests focus on the body of the email, others on the header. One test, for example, checks URLs found in the mail body against a database of URLs associated with spam mail. Other tests check if the formatting of the MIME parts in the mail body is suspicious or if the mail body contains obfuscated HTML or HTML trackers. Some header tests check whether the mail was received via known open relays or if the subject refers to a specific medicine (such as viagra).

Remark that the tests are not perfect: some spam mails will not be marked as spam (*false negatives*), while some normal mails will be marked as spam (*false positives*). A false positive is worse than a false negative: a mail that is automatically directed to a spam mailbox might escape the user's notice completely, while a spam mail in a user's inbox will be easily and swiftly dealt with.

3.2 Bayesian classification of spam

One type of test to help reduce false positives is to determine how similar an incoming message is to known spam mails, and how similar it is to known non-spam mails. In effect, this allows the spam filter to learn from previously encountered mails and become "smarter" in separating spam from non-spam.

Such tests are based on Bayes' theorem. Bayes theorem is used to "flip" conditional probabilities: if we know $P(B \mid A)$, the probability that event B occurs if event A has occurred, and we know the probabilities $P(A)$ and $P(B)$, we can derive the "flipped" probability $P(A \mid B)$, the probability that event A would occur if event B occurred. Formally, Bayes theorem is stated as:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B \mid A) \cdot P(A) + P(B \mid \neg A) \cdot P(\neg A)}$$

In terms of spam filters, Bayes theorem can be used to determine the probability that a mail is spam given that it contains a certain word, $P(\mathit{spam} \mid \mathit{word})$, if we know the probability that that word occurs in spam mails, $P(\mathit{word} \mid \mathit{spam})$. Given a corpus of already-classified mails (either spam or non-spam), the probabilities necessary to compute the "flipped" probability can be estimated.

For example, consider the question of whether a mail containing the word "viagra" is spam. The probability that it is, is denoted $P(spam \mid \text{viagra})$. If we have an existing corpus of mails already classified as spam or non-spam, we can estimate how often this word occurs in spam messages, $P(\text{viagra} \mid spam)$, and in non-spam mails, $P(\text{viagra} \mid \neg spam)$. We also know the percentage of incoming messages that is spam, $P(spam)$. The probability that an unclassified message containing "viagra" is spam is then computed as:

$$P(spam \mid \text{viagra}) = \frac{P(\text{viagra} \mid spam) \cdot P(spam)}{P(\text{viagra} \mid spam) \cdot P(spam) + P(\text{viagra} \mid \neg spam) \cdot P(\neg spam)}$$

In actual use, the result is refined further to prevent the filter from acting on insufficient information. For example, if only one email has ever been received with the word "viagra", then the classification of that one email determines how a new mail with that word will be classified. To prevent this, the resulting probability is scaled with the number of received emails (spam and non-spam) and averaged with an assumed probability of any mail being spam (e.g. set at 50%).

3.3 Bayesian poisoning

Bayesian spam filtering is effective, and therefore widely used. Its ubiquity raises the question if the Bayesian filtering could be deliberately exploited by one or more carefully crafted messages to fool the filter in letting through a spam message – a deliberately triggered false negative. This type of attacks on Bayesian spam filtering is known as *Bayesian poisoning*, as effectively, the spam filter is poisoned to not recognise the offending messages as spam. To achieve this, the spam mail must include a sufficient amount of sufficiently "non-spammy" words, such that in aggregate, the spam mail is marked as non-spam.

In 2004, Graham-Cumming presented the first study [GC04] into Bayesian poisoning. His experiment found that randomly adding words was not successful. He was able to devise a different attack which probes the spam filter to determine words which help ensure a non-spam classification.

To their surprise, Wittel and Wu had contradictory findings: adding random words did influence certain spam filters [WW04]. They suspected this contradiction with Graham-Cummings may be due to differences in used spam corpora or filters tested. Stern et al. [SMS04] showed that Bayesian poisoning can affect filter performance, skewing results to generate more false positives. Lowd and Meek [LM05] find that adding about 150 well-chosen words is effective in getting the average spam message through a filter. They also find that frequent retraining the spam filter is effective against this type of attack. This finding matches the intuition that retraining is important: when the spam filter fails to block spam, it should be retrained.

From existing research, we see that it is possible to masquerade spam as non-spam by using Bayesian poisoning. The concept we explore is turning this

around: is it possible to masquerade non-spam as spam by using Bayesian poisoning? That is: instead of using Bayesian poisoning to increase the false positive rate, can it be used to increase the spam filter's false negative rate? In particular, can Bayesian poisoning be used to suppress mails from an online voting system? We call this *reverse Bayesian poisoning*.

Remark that the mails that the attacker sends to poison the filter *are* spam mails. The only aim of these spam mails is to associate words that are likely used in mails of the voting system with spam. Recall that spam filter retraining is important, especially when a false negative occurs, that is: when a spam message is not filtered. Therefore, reverse Bayesian poisoning is a stealthy attack: sending emails is trivial, and the emails sent by the attacker can easily be made to look like spam (since they *are* spam). A regular user who retrains her spam filter when spam gets through (as is normal) is thus aiding the attacker.

4 Experiment setup

In order to investigate the possibility of using reverse Bayesian poisoning to suppress legitimate mails, we set up an experiment with a local spam filter and attempted to suppress emails by poisoning the filter.

The goal of this experiment was to test the feasibility of this attack, not to perform a thorough evaluation of spam filters and their susceptibility to this attack. As the attack relies on the victim training her spam filter on the attacker's poisoning mails, a thorough evaluation must also take into account the human factor.

Our feasibility experiment analyses whether, for one particular system setup in a fully-controlled and minimal environment, using one or more realistic parameter settings, it is possible to construct a limited number of *attack mails* that, when fed to the spam filter lead to the suppression of a particular administrative message.

In particular we will use the Bogofilter, trained with (a part of) the Enron spam corpus, aiming to suppress emails formatted according to the templates used by the Helios [Adi08] online voting system, such as the template depicted in Fig. 1.

As the experiment focuses on the feasibility of poisoning a Bayesian spam filter, we did not set up an email infrastructure (procmail, mailclient, postfix), but applied Bogofilter directly to the emails. We also assume the worst case for victim actions: the victim will always mark attack mails as spam.

4.1 Bogofilter

Bogofilter[5], originally written by Eric S. Raymond, was based on the ideas of Paul Graham [Gra04]. Graham proposed to apply Bayesian filtering of spam messages based on combining the probabilities for spam of the 15 words with the

[5] http://bogofilter.sourceforge.net/

strongest probabilities: the 15 words for which the absolute value of the difference between that word's spam probability and the neutral spam probability (50%) is the largest. Bogofilter applies Bayesian filtering in the classification of spam/non-spam mails.

The program can be trained on a given corpus of spam and non-spam mail. Moreover, explicit user input on the classification of individual emails can be used to fine-tune or correct the system's decision logic. Furthermore, the user (or system administrator) can fine-tune the statistical algorithms by modifying certain configuration parameters of the Bogofilter system. Examples are the earlier mentioned thresholds: *spam-cutoff* for spam and *ham-cutoff* for non-spam.

An interesting and relevant feature of Bogofilter is the *auto-update* feature. When this feature is used and Bogofilter classifies a message as spam or non-spam, the message will be automatically added to the wordlist for recognizing spam or non-spam, respectively. Bogofilter documentation warns that this behaviour may cause classification errors to propagate.

Retraining a Bayesian spam filter is important and can be done without user intervention – for Bogofilter by using the auto-update feature, for other spam filters by using similar automated updating features or publicly available scripts.

As the Bogofilter documentation warns, automatic retraining may cause classification errors to be exacerbated. This is exactly the goal of our attack: to exacerbate misclassification. To understand the risk that auto-update features pose with respect to this attack, we will conduct two types of experiments: one in which the user will have to classify the attack mails as spam, and one in which the auto-update feature is used to automatically classify the attack mails as spam. In the first case, the number of attack mails should be rather limited, as it involves user interaction. In the second case, we could use a larger number of attack mails.

4.2 Enron mail corpus

The Enron [KY04] corpus[6] is a widely-used dataset consisting of e-mail communication among executives from the (now-defunct) Enron corporation. It contains over 600.000 emails generated by 158 employees. For the experiments, we used the corpus that was enriched by Metsis et al. with spam from various sources [MAP06].

4.3 Helios

Helios [Adi08] is an end-to-end verifiable online voting system with strong privacy guarantees (receipt-free but not coercion-resistant). Its strong security features apply to the election phase and the post-election phase. However, our

[6] http://www-2.cs.cmu.edu/~enron/

attack is upon the announcement and credential distribution process, which is out of scope of Helios' security features – and therefore not protected.

For the purpose of this paper, we omit all technical details of Helios and only focus on the features it offers to support announcement and credential distribution.

Helios is initialized by the voting officials by providing the relevant information, the register of voters and their email addresses, and customizing various emails to be sent out (announcement, credential distribution, etc.).

All voters receive an automatically generated voting invitation from Helios (see Fig. 1). To vote, the voter visits the emailed election URL and authenticates using the credentials from the email. After having voted, the voter receives an email confirmation that her ballot has been received and stored. After the elections close, the stored votes are tallied and the voter receives an email that the tally has been released. The voter can inspect the tally page through the URL provided in the tally email and she can verify that her vote is correctly counted through the provided administrative information.

Dear <*voter.name*>,

<*custom_message*>

Election URL: <*election_vote_url*>
Election Fingerprint: <*voter.election.hash*>

Your voter ID: <*voter.voter_login_id*>
Your password: <*voter.voter_password*>

Log in with your <*voter.voter_type*> account.

We have recorded your vote with smart tracker: <*voter.vote_hash*>
You may re-vote if you wish: only your last vote counts.

In order to protect your privacy, this election is configured
to never display your voter login ID, name, or email address to the public.
Instead, the ballot tracking center will only display your alias.

Your voter alias is <*voter.alias*>.

IMPORTANTLY, when you are prompted to log in to vote,
please use your *voter ID*, not your alias.

–
Helios

Fig. 1. Template of a Helios invitation.

Remark that if a particular voter does not vote (e.g. because she did not receive the invitation), this does not lead to any further action from the Helios system. Further, notice that it is impossible to vote without the email containing the authentication credentials.

4.4 Generating attack mails

The essence of the reverse poisoning attack is in generating the poison. To generate the attack emails, we alternated words from the genuine Helios administrative mails with words from typical spam messages. An example of an attack mail is shown in Fig. 2. Each attack mail contains 115 words, which is equal to the average size of a genuine Helios mail. Regular users will probably mark such emails as spam, even if they are expecting a communication from Helios.

We readily acknowledge that this is a rather rudimentary approach to generating attack emails. However, this experiment aims only to test feasibility of Bayesian poisoning attacks, for which purpose the thusly generated attack emails seem sufficient. Thus, we keep in mind that an attacker can likely generate stronger attack emails.

From: Luxury@experience.com
Subject: Lower monthly payment passwords
Remuneration Election Subsidiary Link: payment Dear
Usury – Reapportionment Helios Reply How Syndicate
to Wholesale Vote Return ======== Computer
Election roots URL: Coattail Your Challenger voter
Believe ID: Decide Your Permit password: Advertisement
Log Pamphlets in Broadcast with Downsize your
...

Fig. 2. Fragment of an example attack mail. The underlined words are from genuine Helios mails.

5 Experiments and analysis

5.1 Experiments

To execute our experiments, we used a standard laptop and installed Bogofilter. Bogofilter's initial database was cleared and trained with part of the Enron corpus. We conducted the experiments with various sizes of the training set.

To test the susceptibility of Bogofilter to reverse Bayesian poisoning, we constructed 50 attack emails. We then iterated over these mails. Each iteration trained Bogofilter on one attack mail and then tested Bogofilter's classification

of Helios emails. We used four fictional elections, each of which generated three genuine emails. In total, each time the classifier was tested on 12 genuine emails from Helios, see Fig. 3.

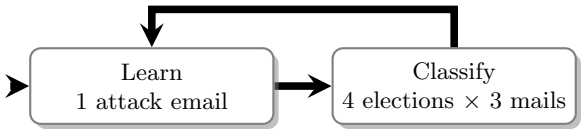

Fig. 3. The process of the experiments.

5.2 Analysis of results

The results of the experiments are depicted in Fig. 4. The figure shows, per election, the averaged probability of genuine Helios mails being marked as spam against the number of attack emails processed by Bogofilter.

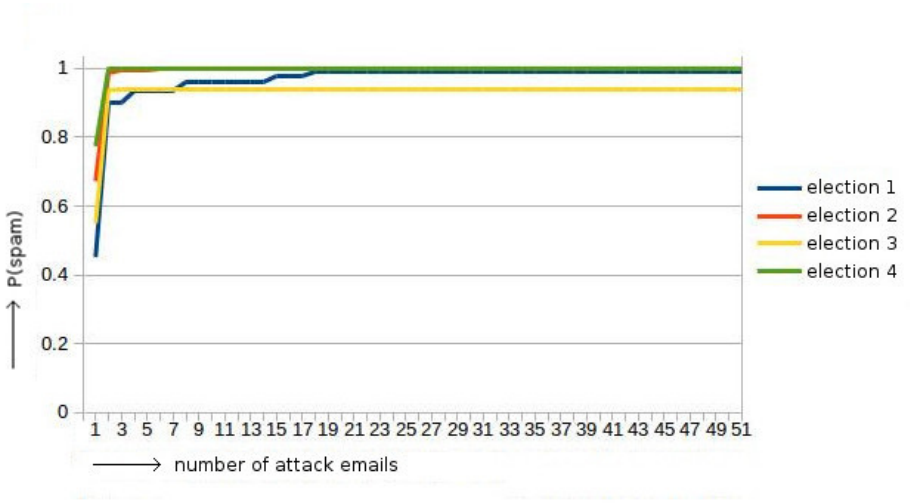

Fig. 4. Average classification of the three genuine Helios emails for each of the four hypothetical elections.

Fig. 4 shows the results when training the spamfilter with 11000 messages from the Enron corpus. Each line represents the average probability of emails being spam for one of the four hypothetical elections. For each election we average the results of the three administrative Helios messages. The vertical axis shows

the classification of the Helios messages as a number between 0 (non-spam) and 1 (spam). The horizontal axis shows the number of attack messages sent before feeding the genuine Helios messages to the spam filter.

As can be seen in the figure, classification of mails from every test election reacted strongly to the presence of only a few attack emails. Bogofilter has a default spam threshold of 99%. We found that it was possible to rate the official emails above this threshold. We also found that the efficacy depended on the size of the training corpus. Nevertheless, even when using the full Enron corpus for training, we found cases where less than 50 attack emails sufficed to have a genuine election email have a spam probability over 99%. When Bogofilter's auto-update feature is switched on, this number of attack emails will be processed without any user interaction.

In other words, notwithstanding the crudeness of our attack, fifty attack emails can be sufficient to reverse Bayesian poison a spam filter to suppress official election emails.

6 Conclusions

Our simple experiments show that reverse Bayesian poisoning is indeed a feasible attack on the administrative processes surrounding a voting system. We expect that the attack mails can be further optimized as to achieve the attacker's goal with fewer attack mails.

Discussion. As the experiments were aimed at testing feasibility, we only tested one particular spam filter (Bogofilter) and made various assumptions: the attacker knows whom to target, the attacker's guess of the format of the genuine emails is sufficiently close to the actually sent emails, and the victims will mark the attack emails all as spam. Moreover, for reverse Bayesian poisoning to constitute an actual attack upon online elections a few more assumptions are made: victims ignore the fact that they did not receive voting credentials, election officials are not alerting voters to check their spam filters, and elections are not repeated if a voter finds an election email in her spam mailbox.

Remark the stealthy nature of this attack. The attack could be executed slowly, over time. The attacker would send attack emails that recipients use to train the spam filter and possibly even discard afterwards. This means that it is possible for a reverse Bayesian poisoning attack to occur without leaving a visible trace on the victim's side.

Moreover, while our experiments focused on local spam filtering by the user, spam is also filtered at the email service provider. Such filters may be trained upon user-classified mails from all users. Such a set is almost perfectly classified[7] and therefore used to classify mails for *all* users.

The downside is that an attacker can sign up for a popular email service and carry out the reverse Bayesian poisoning attack by himself, poisoning the

[7] https://www.quora.com/What-does-the-Report-Spam-feature-really-do-in-Gmail

service's spam filter without sending mails to anyone but his own mail account. The consequences of such an attack can affect many or all users of the service. It is for this reason that we did not dare to execute our experiments on GMail: Helios is used in elections (e.g. by the IACR), and even our crude experiments may train GMail's spam filters to classify genuine election emails as spam. As such, we believe that any experiment with reverse Bayesian poisoning must be done in a controlled environment. If not, others relying on the shared spam filter will be affected without adequate means of reversing the experiment's effects.

Note that reverse Bayesian poisoning of shared spam filters allows an attacker to target groups of voters that share the same spam filter, such as all voters from one institute. As large institutes typically have a generic, shared spam filter, an attacker could thus prevent any voters from a given institute from voting.

Mitigation approaches. Mitigation approaches against reverse Bayesian poisoning can be classified by who should implement the measure:

- User-side mitigation measures (e.g. whitelisting),
- Centrally taken mitigation measures (e.g. multi-channel communication).

Users can mitigate the effects of reverse Bayesian poisoning in various ways. On a filter level, they could whitelist the election email address. This trumps any other spam test, and so would prevent the emails from being suppressed. Another option is to use a reminder service (e.g. a calendar service) to remind them of when credentials are to arrive, and contact election authorities if the credentials did not arrive.

Election officials can also mitigate this attack on different levels. On a technical level, they can use alternative channels (e.g. SMS messages) to notify the users that credentials have been mailed. Remark that these other channels might be attacked as well – for example, the SMS channel in the Norwegian voting system was subverted [KLH13]. They can also address this on a social level, by effectively campaigning that credentials have been sent out and spam filters should be checked.

Future directions. In this work, we investigated a practical attack on the pre-election phase of the Helios voting system. We have not formalised this type of attack, though we see several approaches to doing so. In particular, our attack touches upon the interaction between humans and computers in voting. The security implications of such interactions have been considered before, e.g. Ryan for Prêt á Voter [Rya11], and Kiayias et al. for Helios [KZZ17]. Both these works consider the privacy and verifiability requirements on voting. In terms of the work by Kiayias et al., the reverse Bayesian poisoning attack constitutes an attack upon the setup ceremony. It is possible to formally define a requirement in their framework which would catch any shenanigans with voter credential distribution. Such a requirement could ascertain that credentials were not correctly received, but not whether this is due to lossy communication channels or

due to an active attack such as reverse Bayesian poisoning. To formalise a requirement such that active attacks can be distinguished from regularly occurring circumstances requires further work.

References

Adi08. Ben Adida. Helios: Web-based open-audit voting. In *Proc. 17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

Agg16. Kanupriya Aggarwal. Denial of service attack on online voting system. *Journal of Engineering Science and Computing*, 6(5):5585–5587, 2016.

BT94. Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th ACM Symposium on Theory of Computing (STOC)*, pages 544–553. ACM, 1994.

CFS+06. B. Chevallier-Mames, P. Fouque, J. Stern, D. Pointcheval, and J. Traoré. On some incompatible properties of voting schemes. In *Proc. IAVoSS Workshop On Trustworthy Elections*, 2006.

Cha81. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

Cha04. David Chaum. Secret-ballot receipts: True voter-verifiable elections. *Proc. 25th IEEE Symposium on Security and Privacy (S&P)*, 2(1):38–47, 2004.

CRS05. David Chaum, Peter Y. A. Ryan, and Steven A. Schneider. A practical voter-verifiable election scheme. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS)*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.

GC04. John Graham-Cumming. How to beat an adaptive spam filter. In *Presentation at the MIT Spam Conference*, 2004.

GNR+02. E. Gerck, C.A. Neff, R.L. Rivest, A.D. Rubin, and M. Yung. The business of electronic voting. In *Proc. 5th International Conference on Financial Cryptography (FC'01)*, volume 2339 of *LNCS*, pages 243–268. Springer, 2002.

Gra04. Paul Graham. *Hackers and Painters: Big Ideas from the Computer Age*, chapter "A plan for spam", pages 121–129. O'Reilly, 2004.

JCJ05. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proc. ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 61–70. ACM, 2005.

JRSW04. David Jefferson, Aviel D. Rubin, Barbara Simons, and David Wagner. A security analysis of the secure electronic registration and voting experiment (serve). Technical report, 2004.

KLH13. Reto E. Koenig, Philipp Locher, and Rolf Haenni. Attacking the verification code mechanism in the Norwegian internet voting system. In *E-Voting and Identify - 4th International Conference, Vote-ID 2013, Guildford, UK, July 17-19, 2013. Proceedings*, volume 7985 of *Lecture Notes in Computer Science*, pages 76–92. Springer, 2013.

KY04. Bryan Klimt and Yiming Yang. The Enron Corpus: A new dataset for email classification research. In *Proc. 15th European Conference on Machine Learning*, pages 217–226, 2004.

KZZ17. Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. Ceremonies for end-to-end verifiable elections. In *Proc. 20th IACR International Conference on Practice and Theory in Public-Key Cryptography (PKC'17) Part II*, pages 305–334. Springer Berlin Heidelberg, 2017.

LM05. Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *Proc. Second Conference on Email and Anti-Spam (CEAS 2005)*, 2005.

MAP06. Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive Bayes - which naive Bayes? In *Proc. 3rd Conference on Email and Anti-Spam (CEAS'06)*, 2006.

Rub02. Aviel D. Rubin. Security considerations for remote electronic voting. *Communications of the ACM*, 45(12):39–44, 2002.

Rya11. Peter Y. A. Ryan. Prêt à voter with confirmation codes. In *Proc. 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE'11)*. USENIX Association, 2011.

SMS04. Henry Stern, Justin Mason, and Michael Shepherd. A linguistics-based attack on personalised statistical e-mail classifiers. Technical report, Dalhousie University, 2004.

WW04. Gregory L. Wittel and Shyhtsun Felix Wu. On attacking statistical spam filters. In *Proc First Conference on Email and Anti-Spam (CEAS 2004)*, 2004.