Security Assessment of the Anonyvoter Voting Platform: Source Code Analysis and Formal Verification

Alex Bedford
Beckenham, UK

May 2026

Abstract

This paper presents a security assessment of Anonyvoter, a web-based voting platform developed by Henson IT Solutions and procured by the UK Labour Party in September 2020 for internal party selections and votes. Through source code analysis, we identify multiple vulnerabilities that allow vote outcomes to be altered without audit. These include but not limited to, per-voter ballot code exposure that allows an organiser to cast votes on behalf of non-voting members, live poll modification without version control, authentication weaknesses including a plaintext password fallback and unsalted SHA-256 hashing, and statistical bias in ballot code generation. These findings are contextualised within documented cases in which selection outcomes were contested or criminally investigated, including the deselection of sitting MPs Sam Tarry (Ilford South, 2022) and Beth Winter (Merthyr Tydfil and Upper Cynon, 2023), criminal proceedings under the Computer Misuse Act arising from the Croydon East selection (2023–2026), and the July 2024 selection of Imogen Walker as Labour candidate for Hamilton and Clyde Valley.

Keywords: electronic voting security, voting system vulnerabilities, Anonyvoter, Labour Party, STV integrity risks, voter weighting, source code analysis, UK politics

1 Introduction

This paper examines Anonyvoter, a voting platform built as an ASP.NET Web Forms application with SQL Server backend, first committed in May 2020 by Mark Henson at Henson IT Solutions, a small IT business based in Croydon, UK, where the author (Alex Bedford) was serving as an apprentice under Henson's direction. The platform was procured without tender by David Evans shortly after his appointment as General Secretary of the Labour Party in June 2020, and became a standard for internal party voting including CLP (Constituency Labour Party) candidate selections [Skwawkbox, 2021].

The analysis began in March 2026 when the author returned to the Anonyvoter codebase and examined it systematically for security and integrity risks. This confirmed that the vulnerabilities identified are present in the platform's core logic and, when combined with 'organiser' level access, allow vote outcomes to be altered without detection from voters.

2 Background and Timeline

2.1 Origins

The Anonyvoter platform was first developed in May 2020 by Mark Henson at Henson IT Solutions, where Alex Bedford was working as an apprentice under his direction. Maddie

Henson, Mark's wife, was a Labour Councillor. The platform supported multiple voting systems including First Past the Post (FPP), Single Transferable Vote (STV), and Instant Run-Off Vote (IRV).

2.2 Procurement

In June 2020, David Evans was appointed General Secretary of the Labour Party. By September 2020, Evans had procured Anonyvoter without tender from Henson IT Solutions for use as an internal voting platform across the party [Skwawkbox, 2021]. Concerns were raised immediately regarding the accuracy of candidate selections conducted using the platform.

3 System Architecture

- **Frontend:** ASPX pages with server-side rendering, dynamic HTML generation for vote interfaces
- **Backend:** C# code-behind files containing all business logic
- **Database:** Microsoft SQL Server with direct ADO.NET access via `SqlConnection` and `SqlCommand`
- **Authentication:** Custom SHA-256 password hashing (ASCII-encoded, unsalted)
- **Email:** AWS SES integration for sending vote codes to voters

The core data model consists of the following primary tables:

Table 1: Anonyvoter Database Schema

Table	Purpose
`tblUsers`	Platform administrators/organisers
`tblDistGroups`	Voter distribution lists
`tblDistUsers`	Individual voters within distribution lists
`tblVoteSession`	Voting sessions (time-bounded containers for questions)
`tblVoteQuestions`	Individual poll questions within a session
`tblVoterAnswers`	Vote codes linked to voters and questions
`tblVoted`	Record of which voter voted (separate from answer)
`tblLinkedCodes`	Links between sequential vote codes for multi-question polls

A key design choice in the vote submission path is the separation of `tblVoterAnswers` (which stores the vote code and the voter's identity link) from `tblVoted` (which records that a vote was cast). When a vote is submitted, the system sets `fldDistUser = fldAutoinc`, replacing the voter's actual ID with the answer record's auto-increment ID. This is described as anonymisation. However, the transformation is reversible by anyone with database access, and the original voter identity is retained in `tblVoted`.

4 Critical Vulnerabilities

4.1 CRIT-1: Plaintext Password Fallback

File: `Utils.cs`
Severity: Critical

```
if (encrypted_password == password || encrypted_password == hash_password)
```

`CheckPassword()` compares the stored value against the plaintext input before checking the hash. In normal operation, every password-writing path (`AddUser()`, `ChangePassword()`, `ForgottenPassword()`) calls `GetEncryptionHash()` before storage, so the stored value is always a hash and the plaintext branch does not fire.

Anyone with direct write access to `tblUsers.fldPassword` can insert a known plaintext string for a target account. On the next login attempt by that account, `encrypted_password == password` evaluates true and authentication succeeds, permanently, with no indication to the account holder that anything has changed. Combined with STRUC-2, the vendor holds persistent, silent login access to any organiser account.

4.2 CRIT-2: No Brute-Force Protection on Login

File: `Login.aspx.cs` (entire file)
Severity: Critical

`Login.aspx` has no rate limiting, no account lockout, no CAPTCHA, and no failed-attempt counter. reCAPTCHA v3 is present on `Register.aspx` and `ForgottenPassword.aspx` but not on the login endpoint. Unlimited password guesses can be submitted against any account without throttling or detection.

4.3 CRIT-3: Open Redirect on Login

File: `Login.aspx.cs`
Severity: High

```
Response.Redirect(Request.Params["redirect"].ToString());
```

The `redirect` query parameter is accepted without validation. A voter receiving a link such as `https://anonyvoter.com/Login.aspx?redirect=http://phishing.site` will authenticate on a genuine page and be silently forwarded to an attacker-controlled one.

4.4 CRIT-4: No CSRF Protection

File: `Login.aspx.cs`; `Status.aspx.cs`; entire mutation surface
Severity: Critical

Session state is set at login with no CSRF token:

```
Session["username"] = user.email;
Session["usernumber"] = user.usernumber.ToString();
```

No anti-forgery token is issued or validated on any page. `Web.config` does not configure a `SameSite` attribute for session cookies; the browser default of `Lax` allows the session cookie to be sent on cross-site GET navigation (links, <`img`> tags, redirects). `Global.asax.cs` contains no `Session_Start` handler. No page in the application sets `Page.ViewStateUserKey`, the standard ASP.NET WebForms CSRF countermeasure.

The primary exposure is a GET endpoint. `Status.aspx.cs` executes the ballot-wipe-and-reissue flow inside the `!IsPostBack` block, meaning the operation fires on a plain HTTP GET request triggered by URL navigation alone. An attacker embedding the following on a page the organiser visits while authenticated:

```
<img src="https://anonyvoter.co.uk/Status.aspx?id=42&stat=PRO&sessionno=7"/>
```

causes the organiser's browser to issue the authenticated GET request in the background. The server runs `ChangeStatus("PRO")`, then `Task.Run(() => SetUpAndEmailResponseCodes(...))`:

every existing vote for poll 42 is wiped, the distribution list is re-read, and new ballot codes are dispatched to whoever is currently in `tblDistUsers`. `SameSite=Lax` does not protect against this; it only blocks cross-site POSTs.

For the POST endpoint that modifies the voter list (`VoterGroup.aspx`), ASP.NET WebForms' `__VIEWSTATE` MAC and `__EVENTVALIDATION` tokens provide some friction against fully blind forgery. However, they are tied to the session, not to an origin or a per-operation nonce. An attacker with authenticated session access makes two requests: a GET to harvest `__VIEWSTATE` and `__EVENTVALIDATION`, followed by a POST using those tokens and a modified voter list. The session is the only credential required and carries no binding to a specific IP address, user agent, or origin.

All state-mutating actions (modifying the voter list, triggering a full ballot reset, changing candidates, re-sending ballot links) are accessible to any party who can direct the organiser's browser to a URL while authenticated, or who holds the session cookie.

4.5 CRIT-5: Broken Password Hashing

File: `Utils.cs`
Severity: Critical

```
byte[] data = System.Text.Encoding.ASCII.GetBytes(password);
data = new SHA256Managed().ComputeHash(data);
string hash = System.Text.Encoding.ASCII.GetString(data);
if (hash.Length > 100) hash = hash.Substring(0, 100);
```

Four defects:

1. **Binary to ASCII cast:** SHA-256 produces 32 binary bytes; casting to ASCII collapses all bytes with values 128-255 (roughly half) to `?`. Different passwords that produce binary hashes differing only in the high-bit positions produce identical stored values.
2. **No salt:** identical passwords produce identical hashes; precomputed rainbow tables apply directly.
3. **Truncation to 100 characters:** the 32-byte SHA-256 output becomes 32 ASCII characters after encoding, so this limit is not reached in practice, but the code indicates a misunderstanding of the hash output format.
4. **Non-standard encoding:** the stored hash is not hex or base64 and cannot be verified by any standard tool, making it unauditable.

The net result is a hash function with substantially less entropy than the raw password, no salt, and collision properties that cannot be predicted without running the function.

4.6 CRIT-6: Test Mode Redirects All Email to Vendor

File: `Utils.cs`
Severity: Critical

```
if (ConfigurationManager.AppSettings["live"].ToString() == "N")
    emailaddress = "[REDACTED]@hensonitsolutions.co.uk";
```

When the `live` application setting is `"N"`, every outgoing email, including ballot links for real voters, is redirected to the vendor's personal address. If this setting is left as `"N"` in a production deployment, or changed by whoever controls the server configuration, the vendor receives all ballot codes for every active poll.

4.7 CRIT-7: Modulo Bias in Ballot Code Generation

File: `Utils.cs`
Severity: High

```
crypto.GetNonZeroBytes(data);   // bytes 1-255 only
foreach (byte b in data)
    result.Append(chars[b % (chars.Length)]);   // chars.Length = 62
```

Two weaknesses:

1. `GetNonZeroBytes` excludes zero, producing values 1-255 rather than 0-255. The alphabet is 62 characters. 255 / 62 = 4 remainder 7, so the first 7 characters of the alphabet (`a` through `g`) are approximately 25% more likely per position than the remainder.
2. The non-uniform byte distribution (1-255) is applied to every character, biasing the entire output.

This affects all ballot codes (24 characters), read-out codes (8 characters), and observer codes (24 characters). The practical impact on 24-character codes is small but present; on 8-character codes the bias is more significant.

4.8 CRIT-8: Fire-and-Forget Email Dispatch

File: `Utils.cs`
Severity: High

```
try
{
    Task.Run(() => SendEmailAndUpdateDB(user.email, subject, mess, user.autoinc, ...));
}
catch { }
```

Ballot emails are dispatched in a `Task.Run` with an empty `catch` block. The inner `SendEmailAndUpdateDB()` method wraps the `SendEmail()` call in its own `try/catch` that silently discards exceptions. `SendEmail()` itself does log SMTP failures to a `tblEmailFailures` table when they occur, providing a partial record of delivery failures. However, this mechanism does not constitute a resilient email delivery system: there is no dead-letter queue, no automatic retry, no exponential backoff, and no alerting. The organiser dashboard provides no indication that any ballot email failed to deliver. A voter whose ballot email is silently dropped by the SMTP relay, rejected by the recipient server, or lost due to a transient network failure has no visible recourse and no way to know that a ballot was issued to them. In a system where the ballot email is the sole mechanism by which a voter receives their right to vote, the absence of guaranteed delivery is a structural integrity risk.

5 Structural Deficiencies

5.1 STRUC-1: God Class

File: `Utils.cs` (approximately 4,900 lines)

`Utils.cs` contains 15 classes in a single file. The `Utils` class itself is the dominant component, handling authentication, password hashing, random code generation, email dispatch, database operations, poll lifecycle management, voter list management, PDF generation, and Stripe billing. A static `AnonyvoterApplication` field is initialised in `Global.asax.cs` and

shared across the application, while new `Utils` instances are created freely throughout the codebase with no dependency injection. There is no separation of concerns. Auditing any single capability requires reading the entire file.

5.2 STRUC-2: Single Vendor Controls All Infrastructure

Henson IT Solutions controls:

- Application hosting (IIS)
- SQL Server database (Azure, single vendor account)
- AWS SES sending account
- Domain `anonyvoter.com`

The vendor has database admin access. At the database level, vote anonymisation is not enforced: `tblVoterAnswers.fldDistUser` links a ballot to a voter identity and is only cleared by `SetVoterResponse()` at vote time. Anyone with direct database access can read the association before the vote is cast, and can reconstruct it afterwards from backups or transaction logs. No contractual audit rights, independent oversight, or separation of duties exists.

5.3 STRUC-3: No Audit Logging

No server-side log records:

- Changes to the voter list (which emails were added or removed, when, or from which IP address)
- Poll start, stop, or cancel events
- Candidate list modifications during a live poll
- Results CSV downloads
- Individual ballot code accesses

`SetDistGroupEmails()` performs a full diff-sync with no before/after snapshot. A malicious or erroneous submission is indistinguishable from a legitimate one in the database.

5.4 STRUC-4: Live Poll Modification Without Version Control

File: `VoterQuestion.aspx.cs`; `Utils.cs`

`ChangeVoteQuestion()` allows the organiser to modify the poll title, candidate list, vote type, and other parameters while the poll is in progress (`status = 'PRO'`). There is no version lock, no change log, no voter notification, and no re-validation of already-cast votes against the updated candidate set. A candidate can be added or removed mid-poll with no record in the application layer.

5.5 STRUC-5: Per-Voter Data Visible During Live Poll

File: `Utils.cs`; `VoterQuestion.aspx.cs`

The organiser dashboard displays in real-time:

- Each voter's email address
- Whether they have voted

- Their ballot code (a direct link that can be used to cast their vote if they have not yet done so)
- Whether they were added after the poll started

This data is rendered as live HTML on `VoterQuestion.aspx`. The organiser can see which named individuals have not voted, can selectively re-send ballot links to targeted voters, and holds the ballot codes that, before use, can be submitted on a voter's behalf.

5.6 STRUC-6: No Encryption at Rest

The only cryptographic operation in the codebase is password hashing in `Utils.cs`, which is itself flawed (CRIT-5). No field-level or database-level encryption exists for any other data.

Every sensitive value in the database is stored as a plaintext SQL column:

Table 2: Plaintext Sensitive Data in Anonyvoter Database

Field	Table	Sensitivity
`fldCode`	`tblVoterAnswers`	The ballot token; possession allows voting
`fldReadOutCode`	`tblVoterAnswers`	Observer access token
`fldResponse`	`tblVoterAnswers`	Cast vote choice
`fldDistUser`	`tblVoterAnswers`	Pre-vote voter identity link
`fldEmail`	`tblDistUsers`	Full voter roll
`fldPassword`	`tblUsers`	Organiser credential

The query that drives the live organiser dashboard reads `fldCode`, `fldEmail`, `fldDistUser`, and `fldResponse` in a single unencrypted SELECT. Anyone with SQL read access to `tblVoterAnswers` has every outstanding ballot token for every live poll without touching the application layer.

`fldCode` is the credential that gives a voter their ballot; it is transmitted to them by email as a shared secret. Storing it as a cleartext VARCHAR column means it can be read from any SQL Server backup, Azure DB export, transaction log, or read-replica. The scope of that access is controlled exclusively by the vendor (STRUC-2). There is no cryptographic protection that would render a database copy into anything other than a complete list of every ballot code for every active poll.

6 Multi-Step Attack Scenarios

The following scenarios combine the above findings.

6.1 Scenario 1: Phantom Voter Injection

1. The attacker has organiser-level access (legitimate account or session acquired by other means).
2. The attacker opens the voter list editor and adds attacker-controlled email addresses to the distribution group.
3. Those addresses receive legitimate ballot emails.
4. The attacker votes using those ballot links.
5. After a vote is cast, the platform's anonymisation mechanism replaces the voter's identity link in the ballot record with the record's own internal ID.

6. The ballot is indistinguishable from a legitimate vote in the results.

No server-side control limits the number of addresses that can be added or flags newly added addresses as unusual.

6.2 Scenario 2: Voter Suppression

1. Targeted voters are removed from the distribution group before the poll starts. The voter list editor treats the submitted list as authoritative — any address absent from the new submission is deleted.
2. Those voters are never emailed and cannot vote.
3. Alternatively, a targeted voter's email address is replaced with an attacker-controlled one. The targeted voter receives nothing; the attacker receives the ballot link.
4. No notification is sent to the removed voter. No log entry records the deletion or the substitution.

Note on in-poll protection: When editing a voter list that is linked to a running poll, the voter list editor displays a warning banner stating the group is currently in use. This warning is advisory only — it does not block saving. The voter list is updated immediately regardless of poll status. Voter list modification while a poll is live is therefore accessible through the normal organiser UI with a single click past the warning.

6.3 Scenario 3: Email-Swap Revote

Precondition: A poll is running and the organiser is logged in. Some votes may have already been cast.

1. The organiser navigates to the voter list editor for the distribution group linked to the poll. A warning banner states that the group is currently in use by a running poll. The organiser clears the email address field, pastes in replacement addresses, and clicks Save. The banner does not block the save; the change takes effect immediately.
2. If the organiser now returns to the poll dashboard, the display reflects the change. Voters who were removed are shown as **(REMOVED AFTER POLL STARTED)** — a greyed-out placeholder row. The label confirms that voters were removed after the poll began, but does not record or display which email addresses were deleted. Replacement addresses have not yet been issued ballot codes and do not yet appear in the dashboard.
3. The organiser constructs the poll-start URL directly — the same URL the Start button generates for a poll in "not yet started" state — and navigates to it:

   ```
   Status.aspx?id=X&stat=PRO&sessionno=Y
   ```

   The application does not check the current poll status before processing this request. It executes a full poll start: every existing ballot record for the poll is deleted — including any votes that had already been cast — and new ballot codes are generated for every address currently in the voter list. Ballot emails are dispatched to the replacement addresses.
4. The attacker votes using the new ballot links.

Post-operation state: The poll dashboard shows all replacement addresses with ballot codes issued. The intermediate **(REMOVED AFTER POLL STARTED)** labels from step 2 are gone — the ballot table was wiped and rebuilt from the current voter list, so there are no orphaned records to display. No audit record remains of the original voter list, the deletion, or the poll restart. Original voters' ballot links return a "code not found" error with no explanation.

The advisory warning at step 1 is the only signal this operation produces in the organiser UI. It is not logged. It is not visible to participants. An organiser reviewing the poll results after the fact sees a complete, apparently normal result.

Note on the start URL: The poll-start URL is not a secret or privileged endpoint. It is the URL the normal Start button would produce. That button is only shown when the poll is in "not yet started" state, but the URL itself accepts direct navigation regardless of current status, and the application executes the transition without validation.

6.4 Scenario 3a: Clean Voter Swap via NYS/CAN Intermediate State

This variant achieves the same outcome as Scenario 3 with one additional property: the intermediate **(REMOVED AFTER POLL STARTED)** label at step 2 does not appear at any point. The operation produces no visible signal to the organiser or anyone reviewing the poll afterwards.

Precondition: A poll is running. Votes may or may not have been cast.

1. The organiser navigates to the poll status URL with `stat=NYS`:

   ```
   Status.aspx?id=X&stat=NYS&sessionno=Y
   ```

   The poll is moved back to "not yet started." No participant is notified. The poll dashboard no longer shows an active poll.

2. The organiser navigates to the voter list editor. Because the poll is not currently running, the advisory warning does not appear. The organiser replaces the voter list and saves. The page redirects to the voter groups list with no indication that anything unusual has occurred.

3. The organiser navigates to the poll-start URL:

   ```
   Status.aspx?id=X&stat=PRO&sessionno=Y
   ```

   The application starts the poll as if for the first time: all prior ballot records are deleted, new ballot codes are generated for every address currently in the voter list, and ballot emails are dispatched to the replacement addresses.

4. The attacker votes.

Post-operation state: The poll dashboard shows all replacement addresses. There are no **(REMOVED AFTER POLL STARTED)** rows — the prior ballot records were wiped before the view was rebuilt — and no **(ADDED AFTER POLL START)** labels, because ballot codes issued through the poll-start path are never flagged as post-start additions regardless of when the addresses were added to the voter list. The confirmation reads "All X e-mails sent" with the replacement count. No prior vote total is visible. No application log records the NYS/CAN transition, the voter list change, or the poll restart.

Why the audit markers do not appear:

The platform does have a designed mechanism for marking changes to a running voter list. When an organiser adds an individual voter to an already-running poll through the per-voter resend interface, that voter's entry is flagged **(ADDED AFTER POLL START)** in the organiser dashboard. When a voter is removed from the distribution group while a poll is running, their entry in the dashboard changes to **(REMOVED AFTER POLL STARTED)**. Both of these labels are the platform's intended audit trail for mid-poll voter list changes.

Neither label appears after the NYS/CAN route because neither condition is met. The voter list is edited while the poll is inactive (NYS/CAN), so no removal label is generated during editing. The ballot codes are issued through the poll-start path, which does not apply the "added after start" flag to any row. The result is indistinguishable from a poll that was set up normally and started once.

In Scenario 3 the **(REMOVED AFTER POLL STARTED)** label does appear briefly, in the window between step 1 (voter list edit while PRO) and step 3 (poll re-trigger). At that point the label is visible to the organiser in the poll dashboard. However it does not record which email addresses were removed — it shows only a placeholder for the count of removed rows. After the poll re-trigger the label disappears entirely.

Full automation chain:

The complete operation requires four requests with an active organiser session:

1. `GET Status.aspx?id=X&stat=NYS&sessionno=Y` — returns poll to inactive state
2. `GET VoterGroup.aspx?autoinc=Z` — loads the voter list editor and returns the form tokens required for the next request
3. `POST VoterGroup.aspx` — submits the replacement voter list using the tokens from step 2
4. `GET Status.aspx?id=X&stat=PRO&sessionno=Y` — starts the poll with the replacement voter list

Steps 1 and 4 are GET requests that execute the state change on arrival; no form submission is required. The form tokens in step 3 are bound to the session, not to a specific origin, referrer, or per-operation nonce, and can be scraped and replayed programmatically. All four requests are indistinguishable from normal organiser navigation in any record the application produces.

Reproduction (4 May 2026):

This sequence was executed against a locally deployed instance of Anonyvoter on 4 May 2026. A poll was configured with three voters (test1@test.com, test2@test.com, test3@test.com) and started. After performing the NYS/CAN transition, replacing the distribution group with five new addresses (test1c@test.com through test5c@test.com), and re-triggering PRO, the organiser dashboard showed all five replacement addresses with no audit markers and "All 5 e-mails sent." No trace of the original three voters remained in any application view.

6.5 Scenario 4: Vote Re-identification

The platform's vote anonymisation mechanism (`fldDistUser = fldAutoinc`) is described in Section 3 and Appendix A.1. As established there, the transformation is reversible by anyone with database access: `tblVoterAnswers.fldAutoinc` can be joined to `tblVoted.fldDistUser` to reconstruct the original voter identity for every cast ballot. This means full de-anonymisation of every vote is achievable via a single SQL query by anyone with read access to the database.

At the organiser UI level, the re-identification surface is narrower but still present:

1. The organiser dashboard displays each named voter's email address and whether they have voted, refreshing automatically via a timer (`Timer1_Tick`).

2. The organiser can observe in real-time which named individuals have voted and the order in which votes are cast.
3. The anonymised ballot CSV does not include response timestamps and is re-randomised on each download (`ORDER BY NEWID()`), limiting direct correlation through the CSV export alone.
4. However, in a low-turnout poll where votes are cast sequentially, the organiser's real-time observation of who voted when, combined with knowledge of the running vote total, can narrow the set of possible ballot assignments for each named voter.

The more significant exposure is at the vendor and database level. The vendor holds direct SQL access to both `tblVoterAnswers` and `tblVoted` (STRUC-2). No timing correlation is required: the join between these two tables reconstructs the full mapping of voter identity to ballot content for every vote ever cast on the platform. The "anonymisation" mechanism provides no protection against the party that controls the database.

7 Client-Side Voter List Modification

7.1 Vector: Textarea POST Modification

Files: `VoterGroup.aspx`; `VoterGroup.aspx.cs`; `VoterSession.aspx.cs`; `Utils.cs`

Voter list entry is performed by typing or pasting email addresses into a plain HTML <asp:TextBox TextMode="MultiLine"> textarea. On form submission, the content of this field is transmitted as a URL-encoded POST parameter. `SetDistGroupEmails()` treats the submitted string as the authoritative voter list and performs a full diff-sync against the database. There is no CSV file import; the textarea is the only entry method.

ASP.NET WebForms does not protect against pre-POST modification of textarea content:

- `__VIEWSTATE` MAC: validates only the serialised ViewState blob; does not include textarea content.
- `__EVENTVALIDATION`: validates that the control identity (the button clicked) was from a rendered form; does not validate the content of text inputs.
- `TextBox.Text`: reads directly from the POST body parameter; there is no server-side record of what was displayed to the user when the page loaded.

Any component with write access to the browser DOM or to the HTTP stream between browser and server can modify the textarea value after the user has composed it but before the POST fires:

- A malicious browser extension
- Injected JavaScript (XSS, compromised CDN, or direct injection on an infected machine)
- A locally installed proxy or VPN with TLS inspection

Example of a DOM-level modification:

```
document.querySelector('form').addEventListener('submit', function () {
    var t = document.querySelector('[id$="txtEmails"]');
    // Remove a targeted voter:
    t.value = t.value.replace('target@member.org.uk\n', '');
    // Or inject a phantom voter:
    t.value += '\nattacker@controlled.com';
}, true);  // capture phase fires before default submit
```

The server receives the modified list. `SetDistGroupEmails()` inserts phantom addresses and deletes suppressed ones. No error is raised. No log records the modification. The on-screen confirmation to the organiser reflects the post-submission database state, not their intended input.

If voter lists were uploaded as files, a file would exist on the client machine with its original name, timestamps, and content. A POST-body modification of a textarea leaves no such artefact. The only forensic evidence in the application layer is the database state after the transaction, which shows the result but not the method.

All high-consequence mutations (voter list, poll start/stop, candidate list, results export) are single-step authenticated operations. There is no re-authentication challenge, confirmation code, or out-of-band verification step. On a compromised organiser machine, every capability the organiser holds is accessible to whoever controls that machine without further interaction.

8 Organiser Privilege Summary

The following capabilities are available to any authenticated organiser account without special elevation, accessible through the normal UI:

Table 3: Organiser-Level Capabilities

Capability	Page	Method
Add voters to voter roll	`VoterGroup.aspx`, `VoterSession.aspx`	`SetDistGroupEmails()`
Remove voters from voter roll	`VoterGroup.aspx`, `VoterSession.aspx`	`SetDistGroupEmails()` → `DeleteDistUser()`
See each voter's email, voted status, and ballot code	`VoterQuestion.aspx`	`GetUsersForVote()`
Re-send ballot link to individual voter	`VoterQuestion.aspx`	`EMailResponseCodes()`
Modify candidate list while poll is running	`VoterQuestion.aspx`	`ChangeVoteQuestion()`
Start / stop / cancel poll	`VoterQuestion.aspx`	`cmdNextStage` / `cmdCancel`
Download anonymised ballot CSV	`VoteResults.aspx`	`ExportCSVVotes()`
Grant observer access	`VoterSession.aspx`	`txtObservers` → observer link

None of these actions are logged. None require re-authentication. None are rate-limited or require confirmation.

9 Contested Deployments

Each of the following selections alleged to have used Anonyvoter was disputed by losing candidates.

9.1 Ilford South, October 2022

Sam Tarry, the sitting Labour MP for Ilford South, was deselected in October 2022 following a selection that was alleged to have used Anonyvoter. Tarry reported winning 57% of in-person votes while receiving only 35% of Anonyvoter votes — a 22 percentage-point discrepancy across the two channels [Online, 2024a]. No factor has been identified that would account for a differential of this magnitude between in-person and online voting populations drawn from the same membership roll. Tarry filed an official complaint with the Labour Party alleging vote rigging, threatened legal action to compel disclosure of Anonyvoter records, and stated: "What's gone on in Ilford South, and allegedly in other selections too, is unlike anything I have seen in 20 years in the Labour Party." The Labour Party stated it had full confidence in the integrity of the selection and declined to investigate [Guardian, 2024].

9.2 Merthyr Tydfil and Upper Cynon, 2023

Beth Winter, the sitting Labour MP for Merthyr Tydfil, was deselected in favour of another sitting Labour MP, Gerald Jones, for the newly merged seat of Merthyr Tydfil and Upper Cynon in Wales. Winter reported winning most in-person and postal votes before Anonyvoter was alleged to have been applied, and pressed the Welsh Labour executive for an independent inquiry. Winter stated that "ongoing controversy around Anonyvoter understandably leads to a lack of trust and confidence in Labour Party procedures" [Online, 2024a].

9.3 Hamilton and Clyde Valley, July 2024

The selection of Imogen Walker (wife of Morgan McSweeney, founder of Labour Together) as the Labour candidate for the new constituency of Hamilton and Clyde Valley in July 2024 is the alleged deployment of Anonyvoter with the most direct organisational connection to the platform's procurement chain. Walker narrowly won the selection against local contender Gavin Keatt by 62 votes to 55, with reports indicating Keatt won the hustings but Walker took the nomination "by winning decisively with online votes" [Croydon, 2024d, Express, 2024].

9.4 Croydon East, 2023–2026

The 2023 parliamentary candidate selection for Croydon East resulted in a Metropolitan Police cyber crime investigation after evidence emerged that the constituency's membership database had been tampered with: on one list, 71 members had their home address changed from an earlier 2023 version; 26 had their phone number changed; and 40 had been assigned new email addresses. The Labour Party confirmed that an internal investigation found an attempt at electoral fraud, and reported itself to the Information Commissioner over the compromise of approximately 600 members' personal data. Anonyvoter was alleged to have been the online voting system used for the selection [Croydon, 2024a].

In September 2025, the Metropolitan Police handed its case files to the Crown Prosecution Service. In April 2026, four individuals were charged with conspiracy contrary to Section 1(1) of the Criminal Law Act 1977 and with offences contrary to Section 3 of the Computer Misuse Act 1990. Among those charged was the local party official responsible for membership and the selection process, who had been imposed on the CLP by Labour's London region rather than elected by local members [News, 2025].

Mark Henson, co-creator of Anonyvoter, was serving as interim treasurer of the Croydon East CLP at the time the investigation was opened.

10 Internal Concerns: Mark Henson, May 2024

In May 2024, Mark Henson came forward at a Labour CLP meeting to raise concerns about data integrity relating to Anonyvoter, before continuing to provide the software [Croydon, 2024b].

This confirms that concerns about data integrity were known within the organisation that built and maintained the platform. The technical analysis in this paper establishes that those concerns were well-founded and the vulnerabilities are material.

11 Risk by Access Level

The vulnerabilities described above can be grouped by the access level required to exploit them.

Organiser-level access (available through the web application):

1. Add phantom voters or remove targeted members by editing the distribution group in the voter list editor. An advisory banner appears if a poll is currently running but does not block saving. The voter list is updated immediately.

2. Cast votes on behalf of non-participating members. The live organiser dashboard shows each named voter's ballot link for as long as they have not yet voted. The organiser can navigate to that link directly and submit a vote on the voter's behalf.

3. Modify candidates or poll title mid-poll. The edit controls are hidden in the organiser dashboard once ballot emails have been sent, but the underlying form accepts modifications without a server-side status check. The absence of CSRF protection (CRIT-4) means this can also be triggered without any UI interaction.

4. Retain access indefinitely: no brute-force protection (CRIT-2), no bound on session reuse (CRIT-4).

5. None of the above actions are logged, rate-limited, or visible to participants.

Vendor and database-level access (requires direct SQL or server access):

1. **Set voter weights:** `fldVoterWeight` in `tblDistUsers` can only be written via direct SQL. `AddDistUser()` never writes this field and no web page exposes a setter. A vendor with database access can assign any weight to any voter, causing their ballot to count as multiple votes in the `ProcessAnswer()` weighted total, with no visible indication to the organiser or participants.

2. **Redirect all ballot emails to the vendor:** the `live` application setting is controlled by whoever manages the server's `Web.config`. Setting it to `"N"` causes `SendEmail()` to replace every recipient address with `[REDACTED]@hensonitsolutions.co.uk`, giving the vendor every ballot code for every active poll (CRIT-6).

3. **Access any organiser account:** because `CheckPassword()` compares the stored value against the plaintext input before checking the hash (CRIT-1), writing a known plaintext string into `tblUsers.fldPassword` for a target account grants permanent login access to that account regardless of what the account holder believes their password to be.

12 Lack of Testability and Maintenance

12.1 Untestable Architecture

`Utils.cs` is a single file of approximately 4,900 lines containing 15 classes. The `Utils` class itself is not static but is instantiated freely throughout the codebase via `new Utils()`, with a static `AnonyvoterApplication` field initialised at startup in `Global.asax.cs`. It has no dependency injection, no interfaces, and every method reaches directly for `SqlConnection`, `SmtpClient`, or `ConfigurationManager` at the point of invocation. It is not unit-testable: there is no mechanism to run a method in isolation, inject mock state, or observe a return value without standing up a live SQL Server instance and an SMTP relay. The same applies to `Login.aspx.cs` and the ballot lifecycle across the code-behind files.

This lack of testability has critical security implications:

- **No regression testing:** Changes cannot be verified against prior behavior. Any modification risks introducing undetected regressions that could alter vote counts or authentication logic.

- **No security testing:** Vulnerabilities cannot be systematically hunted via automated testing. Security flaws remain hidden until exploited in production.

- **No audit trail:** Without tests, there is no specification of expected behavior to audit against. Third-party security audits cannot verify that the system behaves as intended.

- **No formal verification:** The codebase cannot be subjected to static analysis or formal methods that require isolated, testable components.

12.2 Abandoned Maintenance Cadence

The commit history reveals a concerning pattern of declining maintenance. Activity was regular through 2021, with multiple commits per month addressing bug fixes, accessibility improvements, and feature additions. By 2022, commits dropped to a handful. In 2023, only a single commit was made. Activity remained minimal in 2024, with only three commits in 2025, and one so far in 2026.

This represents a system that has effectively ceased active development while remaining in production for high-consequence electoral processes. The combination of untestable architecture and abandoned maintenance creates a situation where:

- Security vulnerabilities discovered post-2021 cannot be reliably patched without introducing regressions, as there is no test suite to verify fixes.
- The codebase cannot be audited or verified by third parties without extensive manual code review.
- No security updates or modernization have occurred for critical infrastructure.
- The system remains vulnerable to newly discovered attack vectors in ASP.NET Web Forms, SQL Server, and cryptographic primitives.

A voting system used for parliamentary candidate selections should maintain a rigorous security update cadence, automated testing infrastructure, and formal verification processes. Anonyvoter exhibits none of these characteristics.

13 Recommendations

13.1 Immediate

1. **Freeze all Anonyvoter polls** pending independent audit.
2. **Export and preserve all database records** including `tblVoterAnswers`, `tblVoted`, `tblDistUsers`, and all voter weight fields.
3. **Engage an independent electoral audit firm** to review all polls alleged to have been conducted using Anonyvoter since September 2020.
4. **Re-run all contested selections** using a paper-based or independently audited electronic system.

13.2 Structural

1. **Mandatory tender process** for all voting software used in political party selections.
2. **Independent observation rights** for all party-internal votes, including access to raw vote data.
3. **Open-source requirement** for voting platforms used in democratic processes, with third-party code review.
4. **Cryptographic audit trails** using append-only logs with hash chaining to detect post-hoc modifications.

5. **Voter-verifiable receipts** allowing each participant to confirm their vote was counted as cast.

13.3 Technical

1. **Eliminate voter weights** or require explicit, voter-visible justification for any weight different from 1.0.
2. **Lock voter lists** at poll start with a cryptographic commitment.
3. **Soft-delete with audit trail** for all vote modifications.
4. **Salted, keyed password hashing** (e.g., Argon2id or bcrypt).
5. **Server-side enforcement** of all voting constraints (maximum votes, voter eligibility, etc.).
6. **Automated testing infrastructure** with 100% code coverage for all security-critical paths.
7. **Continuous security updates** with formal vulnerability disclosure and patch management processes.

14 Conclusion

The vulnerabilities identified in this analysis are confirmed through source code examination. Several of them are not defects that could be fixed in isolation but reflect choices that are structural to the platform's design.

The voter weight mechanism (`fldVoterWeight`) is a representative example. It is not a bug; it is an implemented feature. No web UI exposes a setter for it, but the application reads it on every vote count. A voter's weight can be set to any value via direct database access, causing their ballot to count as multiple votes in the final tally with no visible indication to the organiser or other participants.

The platform was procured without tender, operated by a single vendor who controls all infrastructure with no separation of duties, and deployed for candidate selections where the outcome was contested. The exploits documented in this paper are objective and reproducible. The system produces no logs, no audit trail, and no cryptographic commitment to any prior state. The database records are subject to the same trust failure: the vendor holds full write access, starting a poll permanently destroys all prior ballot records with no archive, and transaction logs are controlled exclusively by the vendor. There is no evidential basis on which to attribute integrity to any poll conducted on this platform. The correct position is not that integrity cannot be verified — it is that there are no grounds to assume it was present.

References

Inside Croydon. Scotland yard's cyber crime unit investigating croydon labour. *Inside Croydon*, March 2024a. URL https://insidecroydon.com/2024/03/18/scotland-yards-cyber-crime-unit-investigating-croydon-labour/.

Inside Croydon. Labour vote-fixing whistleblower outs himself at clp meeting. *Inside Croydon*, May 2024b. URL https://insidecroydon.com/2024/05/24/labour-vote-fixing-whistleblower-outs-himself-at-clp-meeting/.

Inside Croydon. Union leaders join calls for labour to abandon anonyvoter. *Inside Croydon*, March 2024c. URL https://insidecroydon.com/2024/03/28/union-leaders-join-calls-for-labour-to-abandon-anonyvoter/.

Inside Croydon. Mcsweeneys new no10 job puts croydon clique in charge. *Inside Croydon*, October 2024d. URL https://insidecroydon.com/2024/10/07/mcsweeneys-new-no10-job-puts-croydon-clique-in-charge/.

Scottish Daily Express. Who is morgan mcsweeney, the clyde valley man who's the new power behind starmer? *Scottish Daily Express*, October 2024. URL https://www.scottishdailyexpress.co.uk/news/politics/who-morgan-mcsweeney-clyde-valley-33837454.

The Guardian. Deselected labour mp sam tarry submits 'vote rigging' complaint. *The Guardian*, April 2024. URL https://www.theguardian.com/politics/2024/apr/03/sam-tarry-deselected-labour-mp-complaint-vote-rigging.

The Guardian. Peter mandelson released on bail after arrest on suspicion of misconduct in public office. *The Guardian*, February 2026. URL https://www.theguardian.com/uk-news/2026/feb/23/peter-mandelson-arrested-on-suspicion-of-misconduct-in-public-office.

BBC News. Welsh labour: Frontbench mp beats left-winger in seat battle. *BBC News*, June 2023. URL https://www.bbc.co.uk/news/uk-wales-politics-65760166.

Parliament News. Fraud allegations in croydon east selection sent to cps. *Parliament News*, September 2025. URL https://parliamentnews.co.uk/fraud-allegations-in-croydon-east-selection-sent-to-cps/.

Morning Star Online. Fears grow that up to 40 labour parliamentary selections may have been rigged. *Morning Star Online*, March 2024a. URL https://morningstaronline.co.uk/article/fears-grow-that-up-to-40-labour-parliamentary-selections-may-have-been-rigged.

Morning Star Online. Labour general secretary quits. *Morning Star Online*, September 2024b. URL https://morningstaronline.co.uk/article/labour-general-secretary-quits.

Skwawkbox. Exclusive: 'anonyvoter' system contract awarded by labour to evans's friend-of-a-friend with no competitive bidding. *Skwawkbox*, March 2021. URL https://skwawkbox.org/2021/03/10/exclusive-anonyvoter-system-contract-awarded-by-labour-to-evanss-friend-of-a-friend-with-

Wikipedia. Morgan mcsweeney - wikipedia, 2026. URL https://en.wikipedia.org/wiki/Morgan_McSweeney.

A Key Source Code Excerpts

A.1 Vote Submission with Anonymity Mechanism (Utils.cs)

```
SqlCommand command = new SqlCommand(
    "UPDATE tblVoterAnswers SET fldResponse = @response, fldAbstain = NULL, " +
    "fldDateTime = @response_date, fldDistUser = fldAutoinc " +
    "WHERE fldCode = @code AND fldVoteQuestion = @votequestion " +
    "AND fldDistUser = @distuser;", util.conn, util.tran);
```

The `fldDistUser = fldAutoinc` clause replaces the voter's actual ID with the answer record's auto-increment ID. This is described as anonymisation, but the transformation is reversible by anyone with database access: `tblVoterAnswers.fldAutoinc` can be joined to `tblVoted.fldDistUser` to reconstruct the original voter identity for every cast ballot.

A.2 Voter Weight Application (Utils.cs)

```
public void ProcessAnswer(VoterAnswers answer)
{
    if (is_abstain)
    {
        if (answer.abstain == "Y")
            tot += answer.voter_weight;
    }
    else if (this.vote == answer.response ||
             answer.response.Contains(this.vote + "~"))
        tot += answer.voter_weight;
}
```

No validation that `voter_weight` equals 1.0. No logging when a non-standard weight is applied. The weighted total is indistinguishable from multiple genuine votes in the results display.

Access requirement: `fldVoterWeight` in `tblDistUsers` is only ever read by the application; it is never written through any web UI. `AddDistUser()` in `Utils.cs` inserts only the group reference, name, and email; it never sets `fldVoterWeight`. Setting a voter's weight to a value other than the default of 1.0 requires direct SQL access to `tblDistUsers`. This is a vendor/database-level capability, not an organiser-UI capability.

A.3 Answer Deletion (Utils.cs)

```
protected void DeleteAnswerForQuestion(int questionno)
{
    bool is_db_open = OpenDBConnection();
    SqlCommand command = new SqlCommand(
        "DELETE FROM tblVoted WHERE fldVoteQuestion = @questionno;", conn);
    command.Parameters.Add(new SqlParameter("questionno", questionno));
    command.ExecuteNonQuery();
    command.Dispose();

    command = new SqlCommand(
        "DELETE FROM tblVoterAnswers WHERE fldVoteQuestion = @questionno;", conn);
    command.Parameters.Add(new SqlParameter("questionno", questionno));
    command.ExecuteNonQuery();
    command.Dispose();
    CloseDBConnection(is_db_open);
}
```

Hard delete from both tables. No archive, no audit log, no soft delete. Complete erasure of all votes for a question.

Call chain: This deletion is triggered when the organiser starts a poll (transitions to PRO status) via `Status.aspx`. All existing answer records for the question are deleted and immediately re-created with new ballot codes; new ballot emails are dispatched to whoever is currently in the voter list at the moment of invocation.

A.4 Author's Assessment of Risk Factors

The following timeline represents the author's assessment of events that, taken together, constituted sufficient risk to warrant a systematic audit of the Anonyvoter codebase and subsequent disclosure of the findings covered in this document.

Table 4: Timeline of Anonyvoter Deployment and Surrounding Events

Date	Event
Oct 2001	Morgan McSweeney joins Labour Party; works with Mandelson's Excalibur database [Wikipedia, 2026]
2010	Peter Mandelson establishes Global Counsel
Jun 2017	Morgan McSweeney founds Labour Together think tank
Sep 2017	Author joins Henson IT Solutions as apprentice (age 16)
May 2020	First commits to Anonyvoter codebase
Jun 2020	David Evans appointed Labour General Secretary; promoted by Morgan McSweeney [Online, 2024b]
Sep 2020	Anonyvoter procured by Labour; concerns raised immediately
Oct 2022	Sam Tarry deselected in Ilford South; files official complaint with Labour alleging vote rigging [Guardian, 2024, Online, 2024a]
Jun 2023	Beth Winter deselected in favour of Gerald Jones for Merthyr Tydfil and Upper Cynon; calls for independent review [Online, 2024a]
Nov 2023	Metropolitan Police cyber crime unit opens investigation into Croydon East selection; 71 members' addresses altered in CLP database [Croydon, 2024a]
Mar 2024	Four Labour-affiliated union leaders (ASLEF, CWU, TSSA, FBU) write to General Secretary Evans demanding moratorium on Anonyvoter [Croydon, 2024c]
May 2024	Mark Henson raises concerns at Labour CLP meeting about data integrity in Anonyvoter; continues providing software [Croydon, 2024b]
Jul 2024	Imogen Walker (Morgan McSweeney's wife) selected as Labour candidate for Hamilton and Clyde Valley; won 62-55 after losing hustings but winning online votes [Croydon, 2024d]
Oct 2024	David Evans resigns to enter House of Lords
Dec 2024	Peter Mandelson appointed UK Ambassador to the US
Sep 2025	Metropolitan Police hands Croydon East case files to Crown Prosecution Service [News, 2025]
Feb 2026	Peter Mandelson arrested on suspicion of misconduct in public office [Guardian, 2026]
Feb 2026	Morgan McSweeney resigns as Downing Street Chief of Staff following Mandelson arrest
Mar 2026	Author begins systematic analysis of Anonyvoter codebase
Apr 2026	Four individuals charged under s.3 Computer Misuse Act 1990 and conspiracy (s.1(1) Criminal Law Act 1977) over Croydon East selection