PhishTrace

A C# / .NET 10 forensic toolkit for detecting Business Email Compromise in Microsoft 365 environments

Project Report

Author: Moaaz Ezeddin G Eljamous

Company: i-Force

Academic Year: 2025–2026

PhishTrace

A C# / .NET 10 forensic toolkit for
detecting Business Email Compromise
in Microsoft 365 environments

Project Report

Author: Moaaz Ezeddin G Eljamous
Company: i-Force
Academic Year: 2025–2026

Foreword

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this internship and the accompanying report. First and foremost, I wish to thank my mentors at i-Force, Pieter van der Hulst and Tjebbe Van Quickenborne, both senior digital forensics investigators, for offering me the opportunity to work on a meaningful project in the digital forensics and incident response domain. Their guidance, technical feedback and professional trust throughout the internship were invaluable.

I am equally grateful to the members of the DFIR team at i-Force, Lara Sezgin and Borna Talebi, who welcomed me into their workflow and patiently answered my questions about case procedures, evidence handling and tooling standards. I owe a word of appreciation to my internship coach, Thomas Clauwaert, and the internship coordinator, Kristien Roels, for their support, and lecturer Heidi Terryn for the structured writing guide that shaped this report. Lastly, I dedicate an honorable mention to my family and friends, whose profound belief in me and unwavering encouragement were invaluable throughout my entire academic journey.

Moaaz Ezeddin G Eljamous

Brugge, Belgium, May 2026

AI Use Disclaimer

In accordance with the faculty guidelines on the permitted use of artificial intelligence in bachelor's theses, the following declaration outlines how and where AI tools were used during the preparation of this report.

- **Language editing:** improving sentence formulation, grammar and conciseness of paragraphs written by the author.
- **Technical verification:** cross-checking technical descriptions against the author's own implementation.
- **Design and layout:** assisting in the visual design and page layout of the report.

All AI-generated content was critically reviewed before inclusion. Technical claims were verified against the actual tool implementation.

Abstract

This report describes the design, implementation and validation of PhishTrace, a forensic toolkit written in C# on .NET 10 for detecting Business Email Compromise (BEC) in Microsoft 365 environments. PhishTrace was developed during an internship at i-Force, a Belgian independent forensic audit firm specializing in fraud investigation, computer forensics and fact finding.

PhishTrace consists of two complementary command-line tools. UalBecDetector analyses the Microsoft 365 Unified Audit Log (UAL) for tenant-level BEC indicators by porting and extending the open-source UAL-Analyzer.ps1 PowerShell script by Lethal-Forensics. It retains the original catalogue of 73 detection rules while replacing the execution model with a maintainable, multi-user, single-pass pipeline. Key improvements include an explicit severity model, incident correlation and a severity-routed Excel workbook that automatically classifies findings and presents them in an at-a-glance dashboard.

PstBecDetector scans PST and OST mailbox files offline for message-level BEC indicators, including phishing URLs, file-share lures, malicious SharePoint notifications and domain reputation signals.

Validation on a real suspected BEC case of 3,394 UAL event records produced 454 findings. The severity distribution confirms parity with the original script, while the pre-classified output allows analysts to quickly triage which accounts warrant investigation without manually sifting through thousands of raw log entries. Together, the two tools give analysts both the tenant-wide audit perspective and the per-mailbox content perspective needed for a complete BEC investigation.

Keywords: Business Email Compromise, Microsoft 365, Unified Audit Log, PST, digital forensics, incident response, C#, .NET, MITRE ATT&CK, phishing detection

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
AiTM	Adversary-in-the-Middle
API	Application Programming Interface
ASN	Autonomous System Number
BEC	Business Email Compromise
CLI	Command-Line Interface
CSV	Comma-Separated Values
DFIR	Digital Forensics and Incident Response
ENISA	European Union Agency for Cybersecurity
FBI	Federal Bureau of Investigation
GUID	Globally Unique Identifier
IC3	Internet Crime Complaint Center
IPC	Inter-Process Communication
JSON	JavaScript Object Notation
MFA	Multi-Factor Authentication
OAuth	Open Authorization
OST	Offline Storage Table
PST	Personal Storage Table
SIEM	Security Information and Event Management
UAL	Unified Audit Log
XLSX	Office Open XML Spreadsheet

Glossary

Term	Definition
Adversary-in-the-Middle (AiTM)	A phishing technique in which the attacker proxies the victim's connection to a legitimate service, capturing credentials and session tokens in real time and thereby bypassing multi-factor authentication.
Audit log	A chronological record maintained by Microsoft 365 of user and administrator activities across the tenant, used as the primary evidence source for tenant-level investigation.
Business Email Compromise (BEC)	A class of cyberattack in which an adversary gains access to or impersonates a legitimate business email account to commit fraud, redirect payments or exfiltrate information.
Consent grant	The act of a user authorising an OAuth application to access resources on their behalf; abused in illicit consent grant attacks to obtain persistent mailbox access without a password.
Incident correlation	The process of grouping individually detected findings that relate to the same attack episode, so that analysts review one coherent incident rather than scattered events.
Inbox and transport rule	Mailbox- or tenant-level mail-processing rules. Inbox rules act on a single mailbox while transport rules act organisation-wide; both are abused after account takeover to hide replies, auto-forward mail to external addresses or delete fraud-detection notifications.
Inbox rule	A mailbox-level rule that automatically processes incoming mail; abused by attackers to hide, forward or delete messages and conceal fraudulent activity.
MITRE ATT&CK	A globally accessible knowledge base of adversary tactics and techniques based on real-world observations, used to classify and map detected behaviours.
Multi-Factor Authentication (MFA)	An authentication method requiring two or more verification factors, designed to prevent account

Term	Definition
	access from a stolen password alone; bypassed by adversary-in-the-middle attacks that capture live session tokens.
OAuth	An open authorisation framework that lets applications obtain delegated access to resources without handling the user's password; abused through illicit consent grants.
Phishing	A social-engineering attack that deceives a victim into revealing credentials or authorising access, typically through fraudulent messages or web pages.
Port	The act of rewriting software from one language or platform to another while preserving its behaviour; here, the rewrite of UAL-Analyzer.ps1 from PowerShell to C#.
Severity model	A structured scheme that assigns each detection rule a fixed severity level, replacing implicit or formatting-based prioritisation with explicit, deterministic classification.
Session token	A credential issued after authentication that keeps a user signed in; if stolen, it allows an attacker to access an account without re-authenticating or passing MFA.
Tenant	An organization's dedicated instance of Microsoft 365, encompassing its users, mailboxes, configuration and audit data.
Unified Audit Log (UAL)	The consolidated Microsoft 365 audit trail that records activity across Exchange, SharePoint, Entra ID and other services in a single searchable log.

Introduction

Business Email Compromise is one of the most financially damaging forms of cybercrime today. In a Microsoft 365 environment, modern BEC attacks exploit session tokens, OAuth consent flows and inbox-rule manipulation rather than relying on stolen passwords alone. Investigating these attacks requires structured analysis at two levels: the tenant-wide audit trail that records who did what and when, and the individual mailbox contents that reveal phishing messages, malicious URLs and social-engineering lures.

This report presents PhishTrace, a forensic toolkit developed during an internship at i-Force, a Belgian independent forensic audit firm. PhishTrace consists of two complementary command-line tools. UalBecDetector analyses the Microsoft 365 Unified Audit Log (UAL) for tenant-level BEC indicators. PstBecDetector scans PST and OST mailbox files offline for message-level indicators such as phishing URLs, file-share lures, fake SharePoint notifications and domain reputation signals. Together, the two tools cover the full evidence spectrum of a BEC investigation.

The primary focus of this report is UalBecDetector, which ports and extends the open-source UAL-Analyzer.ps1 PowerShell script by Martin Willing of Lethal-Forensics. The port retains the original's 73 detection rules and MITRE ATT&CK mappings while introducing a structured severity model, incident correlation, configurable reference data and a deterministic Excel workbook output. PstBecDetector is described at a higher level as the sister tool that completes the PhishTrace toolkit.

The report is structured as follows. Chapter 1 introduces i-Force and the context of the internship. Chapter 2 explains why BEC in Microsoft 365 warrants a dedicated detection toolkit. Chapter 3 describes the reference PowerShell script and the rationale for porting it. Chapter 4 defines the goals and scope of the project. Chapter 5 provides the technical background needed to understand the architecture. Chapter 6 presents the architecture and design of UalBecDetector. Chapter 7 discusses the key implementation choices made during the port. Chapter 8 reports the validation results on a real case. Chapter 9 covers PstBecDetector, the sister tool for mailbox-level analysis. Chapter 10 reflects on the project, the lessons learned and possible future work.

1 The Company and My Role

1.1 i-Force

i-Force is a Belgian independent forensic audit firm that conducts investigations with IT expertise. The firm focuses on three core pillars: fraud prevention and detection, computer forensics, and fact finding. As an independent organization, i-Force handles each engagement impartially and objectively, ensuring that its approach and reporting are designed to be used in legal proceedings.

The i-Force team consists of specialized and certified IT experts who cover the full range of investigative capabilities required to uncover the truth in complex cases. Services span fraud investigation, digital forensics, computer crime analysis and advisory services towards IT security and integrity. The company supports private and public organizations that face cyber incidents such as ransomware, data theft, insider abuse and Business Email Compromise (BEC).

For each case, the Digital Forensics and Incident Response (DFIR) team works with a strictly isolated evidence store. Case data is never copied outside the i-Force environment, and all analysis tooling must respect that boundary. This requirement shapes every technical decision discussed later in this report, in particular the choice to keep UalBecDetector fully offline by default.

1.2 My role and the project assignment

During my internship I was placed on the DFIR team as a junior digital forensics analyst with a cybersecurity profile. My daily work alternated between two activities: assisting senior analysts with active cases and developing internal tooling that the team could reuse on future engagements.

The project described in this report belongs to the second activity. I was asked to design and build a maintainable in house toolkit for BEC investigations in Microsoft 365 environments. The result is PhishTrace, a project containing two complementary tools. UalBecDetector analyses the tenant-wide Unified Audit Log. PstBecDetector scans individual PST and OST mailbox files for phishing indicators. Both are written in C# on .NET 10 and run as stand-alone executables on any Windows workstation inside the i-Force evidence environment.

2 Problem Context

2.1 Business Email Compromise

Business Email Compromise (BEC) is a class of attack in which a threat actor gains control of, or convincingly impersonates, a legitimate mailbox in order to defraud the victim organization. Typical outcomes include diverted invoice payments, fraudulent wire transfers, exfiltration of confidential correspondence and downstream supply-chain phishing through the trusted sender. According to the FBI Internet Crime Complaint Center, BEC remains one of the most financially damaging categories of cybercrime worldwide [1].

In a Microsoft 365 environment, the modern BEC kill chain has moved beyond simple password theft. The dominant technique today, and the one central to the case examined in this report, is Adversary-in-the-Middle (AiTM) phishing. The attack begins with an email containing a link, often disguised as a shared document or file notification. When the victim clicks the link, they are taken to a counterfeit Microsoft sign-in page that is not a static fake but a reverse proxy. As the victim enters their credentials, the proxy forwards them in realtime to the genuine Microsoft sign-in service.

When Microsoft responds with a multi-factor authentication (MFA) challenge, the proxy relays that challenge back to the victim and forwards their response onward, so the victim completes a fully legitimate authentication. At the end of this exchange Microsoft issues a session cookie, and it is this cookie that the proxy captures. The victim is then redirected to the real Microsoft page and sees a normal, successful login, while the attacker replays the stolen cookie from their own infrastructure to access the mailbox. Because the cookie is issued *after* MFA has been satisfied, the attacker bypasses MFA entirely without ever needing the password again [2].

Several related techniques support or extend this access. Session-cookie theft and token replay covers the broader reuse of stolen cookies or refresh tokens from attacker infrastructure [3]. OAuth illicit consent grants trick the victim into consenting to a malicious third-party application that then reads mail or files through Microsoft Graph using a refresh token, invisible to traditional sign-in monitoring [4]. Inbox and transport rule abuse occurs once the attacker is inside the mailbox: rules are installed that hide replies, forward mail to external addresses or delete fraud-detection notifications [5].

Each of these techniques leaves a specific, recognizable trace in the Unified Audit Log. The challenge is that the traces are scattered across dozens of record types and that the raw log is far too noisy to read by hand.

2.2 The Microsoft 365 Unified Audit Log

The Unified Audit Log (UAL) is the central audit feed for a Microsoft 365 tenant. It records events from Entra ID sign-ins, Exchange Online mailbox operations, SharePoint and OneDrive activity, Microsoft Teams, and consent grants, among others [6]. For a forensic analyst, the UAL is therefore the closest thing to a tenant-wide flight recorder.

However, the UAL has three properties that make it difficult to exploit directly. First, schema sprawl: different RecordType values bring entirely different field sets, and most of the payload lives inside a nested AuditData JSON column. Second, volume: even a small tenant produces tens of thousands of records per week, and a typical investigation deals with hundreds of thousands. Third, the lack of severity: Microsoft does not classify individual log events as benign, suspicious or malicious; that judgement is the analyst's job.

A good UAL analysis tool must therefore normalise the schema, condense the volume, and assign a defensible severity to each event.

2.3 Mapping to MITRE ATT&CK

The detection rules implemented in this project align with several techniques from the *MITRE ATT&CK* Enterprise matrix, in particular *T1078.004 (Valid Accounts: Cloud Accounts), T1098.005 (Account Manipulation: Device Registration), T1539 (Steal Web Session Cookie), T1550.001 (Use Alternate Authentication Material: Application Access Token), T1556.006 (Modify Authentication Process: Multi-Factor Authentication) and T1114 (Email Collection)* [7]. Aligning the rule catalogue with this framework makes the output readable for any analyst, even one who has never used the tool before.

3 The Reference Tool and the Motivation for a Port

3.1 What UAL-Analyzer.ps1 already does well

The starting point for this project is UAL-Analyzer.ps1, an open-source PowerShell script of approximately 9 600 lines written by Martin Willing of Lethal-Forensics [8]. The script ingests the UAL-Combined.csv file produced by Invictus' Microsoft-Extractor-Suite [9] and applies a catalogue of 73 detection rules (LETHAL-001 through LETHAL-073, with LETHAL-070 intentionally absent). Each rule maps to a known attacker behaviour and to a MITRE ATT&CK technique. The script's output is a tree of CSV and XLSX files coloured red, orange, yellow or green depending on the suspicion level.

The rule catalogue is excellent. The thresholds are battle-tested. Using the same catalogue as a baseline means that the new tool starts from a recognised, community-validated body of detection knowledge rather than from a green field.

3.2 Why a port was necessary

Despite its strengths, UAL-Analyzer.ps1 has a number of properties that make it less than ideal for daily use inside a DFIR team. First, there is a single-user gate. The script refuses to run on a UAL export that contains more than one user identity. Multi-tenant or multi-mailbox investigations require either splitting the file manually or commenting out the safety check.

Second, the processing phases communicate through file-as-IPC: CSV files are written to disk by one phase and re-read by the next, wasting I/O and making the pipeline hard to reason about. Third, severity is implicit, as it is encoded only through Excel conditional-formatting expressions, leaving no programmatic way to ask which findings are critical. Fourth, false-positive filters are hard-coded. Some rules exclude specific organisation names directly in the script, so adjusting a carve-out requires editing the source. Fifth, Microsoft application identifiers (AppId GUIDs) are sprinkled through the script as string literals with no central reference table.

In short, the script is a high-quality detection catalogue stapled to a shell-style execution model. The port keeps the catalogue and replaces the execution model.

3.3 What the port adds

Translating the script into a maintained C# project on .NET 10 enables a number of improvements that would have been impractical to retrofit into the original. These include an explicit FindingSeverity enum (Critical, High, Medium, Low, VeryLow) surfaced as a column in the output workbook; a stateless rule contract (ILethalRule) so that rules can be unit tested individually and parallelised later; an incident-correlation layer that collapses related findings into operator-facing units; a pre-built investigation dashboard that surfaces Indicators of Compromise immediately upon opening, providing ready-made reporting without manual sorting or filtering; a single-pass, in-memory pipeline with no file-based IPC between phases; and a CSV-driven reference-data layer and blacklist layer that operators can edit without recompiling.

4 Goals and Scope

4.1 Functional goals

UalBecDetector shall: ingest a UAL-Combined.csv file produced by Microsoft-Extractor-Suite; apply all 73 LETHAL detection rules with the documented thresholds; support optional IP enrichment using the IPinfo Lite API; collapse raw findings into incidents using a defined family map; produce a single XLSX workbook with severity-routed sheets, a pre-built investigation dashboard that presents Indicators of Compromise immediately upon opening without requiring manual sorting or filtering, and ready-made reporting output; run on Windows from a single self-contained executable.

4.2 Non-functional goals

The tool shall run fully offline by default, with IP enrichment as the only network call and strictly opt-in. It shall be deterministic and reproducible: the same input must produce the same output. It shall never crash on malformed records but instead log the error and continue. It shall auto-increment the output filename rather than overwrite a prior run.

4.3 Out of scope

UalBecDetector deliberately does not cover live UAL collection from a tenant (handled by Microsoft-Extractor-Suite), real-time alerting or SIEM integration, or a long-lived web dashboard or multi-user UI. Analysis of mailbox contents at the message level is handled by the sister tool PstBecDetector, which is described in chapter 9.

4.4 Success criteria

Two conditions define success for this project. First, the rule engine must produce output that an experienced analyst would consider equivalent to the PowerShell script on the same input. Second, the improvements listed in section 3.3 must be present and demonstrable on a real case.

5 Technical Background

5.1 The shape of a UAL record

Every UAL row carries a small set of top-level fields and a single nested JSON column called AuditData. The top-level columns identify the workload (RecordType), the operation (Operations), the acting user (UserIds) and the rough event timestamp (CreationDate). The vast majority of the forensic value, however, lives inside AuditData: client IP, session identifier, target object, modified properties, and more. A normalised internal schema that flattens the relevant fields into 23 explicit columns is therefore the first thing the tool builds.

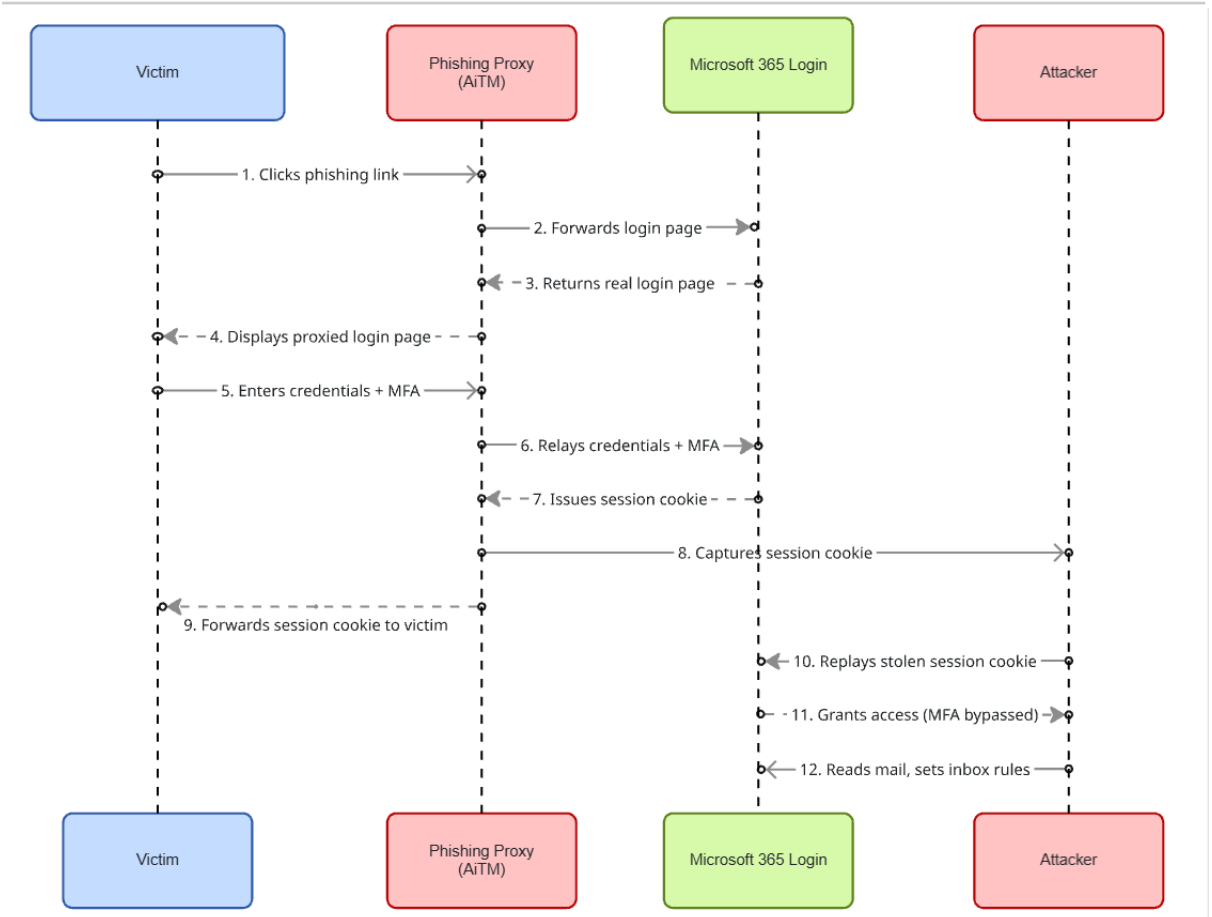
5.2 Adversary-in-the-Middle phishing

The forensic signature of an AiTM compromise in the Unified Audit Log is not, in practice, a distinctive user-agent string [2]. Because the attacker replays the stolen session cookie from an ordinary browser, the recorded user-agent of the malicious session is typically indistinguishable from a legitimate one.

In the case examined in this report, the fraudulent login presented a standard Chrome-on-Windows user-agent. The reliable signal is instead behavioral and is built on two observations [12]. First, the malicious sign-in originates from an IP address the account has never used before, frequently belonging to a hosting or VPN provider such as DigitalOcean or Azure rather than a residential or corporate network.

Second, the session identifier issued to, or reused by, that sign-in can subsequently be traced across further activity in the log. UalBecDetector therefore keys its AiTM detection on session correlation: once a sign-in is judged suspicious, its SessionId is recorded, and every later event carrying that same identifier is flagged as part of the same malicious session.

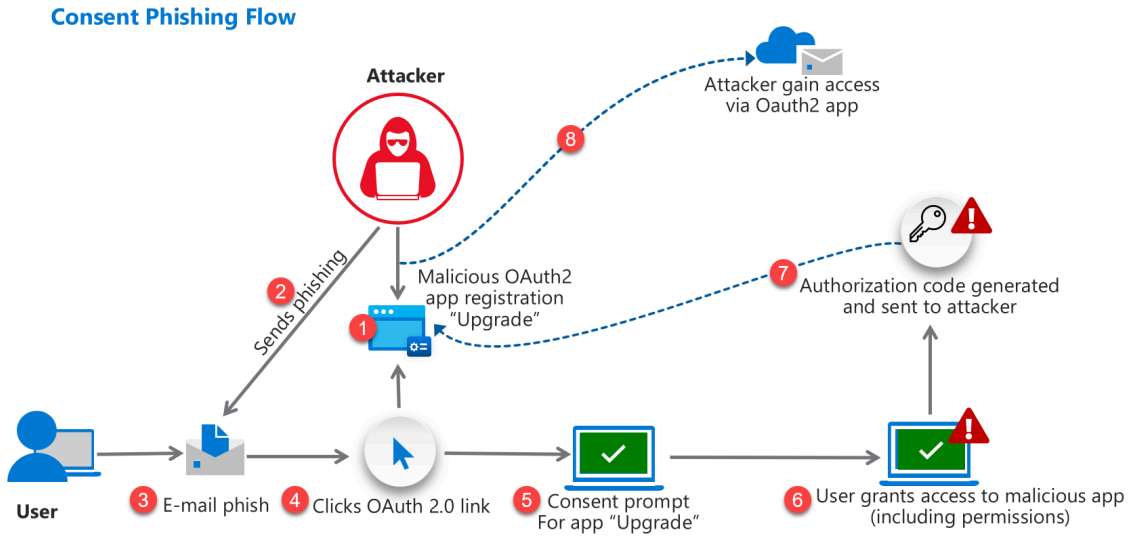
The aforementioned steps are the basis of the tool's session-tracking logic, and they are what links an initial suspicious login to the actions the attacker performs afterwards. Figure 5.1 shows the full sequence of an AiTM attack, from the initial phishing link through to the attacker's use of the stolen session cookie.



[Figure 5.1: Sequence of an Adversary-in-the-Middle phishing attack]

5.3 OAuth illicit consent grants

Modern Entra ID applications can request OAuth permissions on behalf of a user. An attacker who is unable to steal a password can still trick the user into consenting to a malicious application, for example an app requesting Mail.ReadWrite and offline access. Once consent is granted, the attacker reads mail through Microsoft Graph using a refresh token that is invisible to traditional sign-in monitoring [4]. The UAL records the consent event under the operation Consent to application, which the tool flags whenever the target application is unfamiliar.



[Figure 5.2: OAuth illicit-consent grant flow.]

5.4 Inbox and transport rule manipulation

The tool decodes inbox-rule parameters against a localised list of suspicious target folders in multiple languages, since Outlook folder names are translated by locale. A rule that automatically moves incoming messages from the finance department to a rarely checked folder such as RSS Feeds, marks them as read and forwards them to an external address is the canonical example. The UAL operations New-InboxRule, Set-InboxRule, New-TransportRule and Set-TransportRule capture these events, but only the rule's raw parameters appear in the log. After initial access, attackers routinely create rules that hide their activity from the victim [13].

Inbox rules are not only used to hide the attacker's own correspondence. A more deliberate use is the suppression of breach-notification messages.

This tactic exploits the way Business Email Compromise spreads through the supply chain. The mailbox that sends the phishing email to the victim is often itself a legitimate account at a partner organization that was compromised earlier. When that partner discovers the intrusion, it will typically warn its contacts not to trust recent emails from the affected address. If the attacker already controls the victim's mailbox by this point, they can pre-empt that warning. A rule is created that matches any incoming message from the original compromised sender domain and diverts it again, commonly to the RSS Feeds folder, so that the victim never sees the alert. The fraud is thereby kept alive even after the source of the phishing has been identified elsewhere.

This behavior is recorded in the same New-InboxRule and Set-InboxRule operations, with the sender address of the originally compromised mailbox appearing as the rule's matching condition.

5.5 IP enrichment

Geolocation and Autonomous System Number (ASN) information adds crucial context to a ClientIP field. A login from a residential proxy in a country the user has never visited is far more significant than the IP address alone suggests. The tool uses the IPinfo Lite API for this enrichment, with one HTTP call per distinct IP and a small concurrency limit to stay within the free-tier quota.

6 Architecture and Design

6.1 Pipeline overview

The processing pipeline has five phases that run in strict order:

Phase 1 is Parse : UalCsvParser reads the CSV and produces a list of UalRecord objects with the JSON-nested fields already flattened.

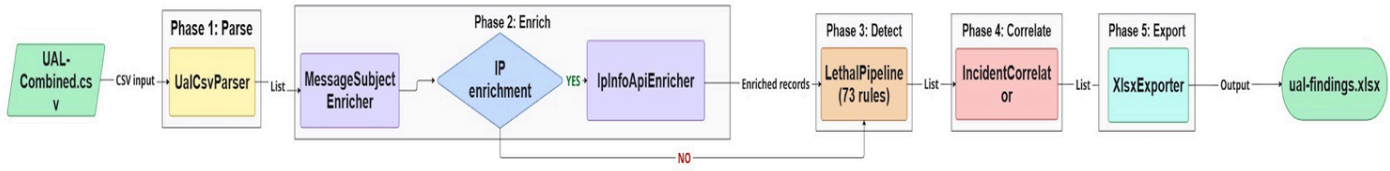
Phase 2 is Enrich : MessageSubjectEnricher recovers missing subjects on MailItemsAccessed records by joining them to send and create records on their shared InternetMessageId, surfacing which message was opened.

Then, if enabled, IpInfoApiEnricher adds country, region, ASN and organisation per distinct ClientIP, turning a raw address such as 46.101.234.200 into DigitalOcean LLC, AS14061, Netherlands

Phase 3 is Detect :LethalPipeline runs each ILethalRule against the enriched records and produces LethalFinding objects.

Phase 4 is Correlate : IncidentCorrelator groups findings by (UserId, UTC-day, RuleFamily) and merges multi-rule hits on the same record into a single Incident.

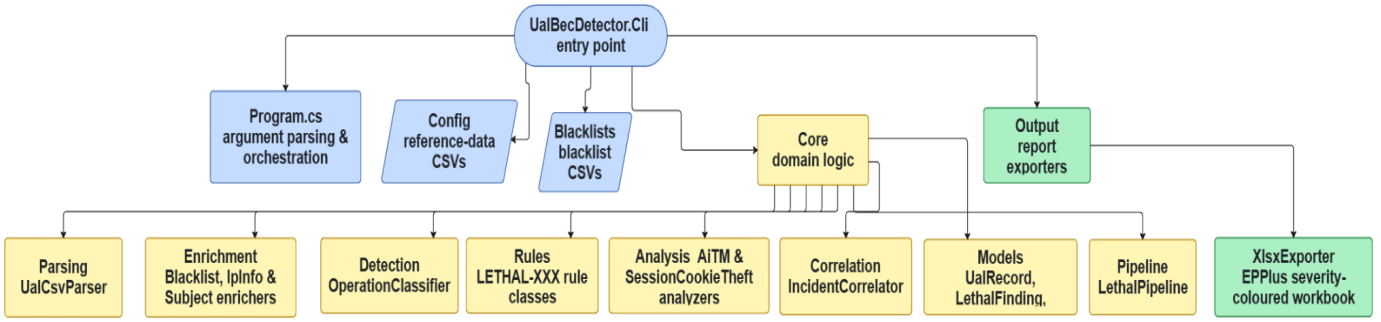
Phase 5 is Export : XlsxExporter writes a severity-routed workbook using EPPlus.



[Figure 6.1: Left to right flowchart of the five pipeline phases, with the data type flowing on each arrow.]

6.2 Solution structure

The solution is organized around three top-level folders. UalBecDetector.Cli is the entry point, handling argument parsing, and hosting the reference-data and blacklist CSV files. Core contains the domain logic split into Parsing, Enrichment, Detection, Rules, Analysis, Correlation, Models and Pipeline namespaces. Output contains XlsxExporter, isolated from the rest of the codebase so that an alternative exporter (for example, a JSON report) could be added later without touching the detection logic.



[Figure 6.2: Organization chart of the C# namespaces.]

6.3 The rule contract

Every detection rule implements the ILethalRule interface, which exposes a RuleId property (for example, LETHAL-018), a RuleFamily property (for example, InboxRuleManipulation), and an Evaluate method that receives the record list as a read-only collection and returns findings.

Rules are stateless: they never mutate shared state and never touch the file system. This is a deliberate precondition for two things: unit testing each rule in isolation, and a possible future parallel evaluator.

6.4 The LETHAL detection rule catalogue

UalBecDetector implements the full catalogue of 73 LETHAL detection rules developed by Martin Willing of Lethal-Forensics. The rules are organised into fifteen functional families, each targeting a distinct category of attacker behaviour. Two further detection components extend the catalogue beyond the shipped 73 rules: the BEC Chain Correlator, which links related rule hits across families into a single attack narrative, and the Novel Login IP rule, a behavioural anomaly detector that is still in progress and not yet counted in the 73. Table 6.1 summarises all of these, listing for each the rule identifiers it covers, the source file that implements it, the primary severity tier assigned to its findings, and the corresponding MITRE ATT&CK technique [7].

Table 6.1: LETHAL ruleset summary by family and severity tier

Rule Family	Rule Range	Source File	Primary Severity	MITRE Technique
Inbox Rule Manipulation	LETHAL-001 to 015	InboxRuleRules.cs	High / Critical	T1114
Transport and Mailbox Forwarding	LETHAL-016 to 022	TransportMailboxRules.cs	High / Critical	T1114.003
Mailbox Permissions	LETHAL-023 to 027	MailboxPermissionRules.cs	High	T1098
OAuth App Abuse	LETHAL-028 to 031	OAuthRules.cs	Critical / High	T1098.005
Anti-Forensics	LETHAL-032 to 036	AntiForensicsRules.cs	Critical	T1562
Blacklist Enrichment	LETHAL-037 to 040	BlacklistEnrichmentRules.cs	High / Critical	T1078.004
Mailbox Actions (Single-event)	LETHAL-040 to 045	MailboxActionRules.cs	Low / Medium	T1114
Volume Threshold Rules	LETHAL-046 to 051	VolumeMailboxRules.cs	Medium / Low	T1114
SharePoint Exfiltration	LETHAL-052 to 059	SharePointRules.cs	High / Medium	T1213
Authentication Anomalies	LETHAL-060 to 062	AuthAnomalyRules.cs	High	T1078.004
AiTM and Session Theft	LETHAL-063 to 066	AiTMRules.cs	Critical	T1539 / T1550.001
Mail Items Accessed	LETHAL-067 to 068	MailItemsAccessedRules.cs	High / Critical	T1114
Update Message	LETHAL-069	UpdateMessageRules.cs	Low	T1114
Teams Initial Access	LETHAL-071 to 073	TeamsRules.cs	High	T1566
Novel Login IP	LETHAL-075 (in progress)	NovelLoginIpRule.cs	High	T1078.004
Defensive Actions	green-tier	DefensiveActionRules.cs	Defensive	N/A
Device Code Authentication	extra	DeviceCodeRules.cs	High	T1078.004
BEC Chain Correlator	correlator	BecChainCorrelatorRule.cs	Critical	multiple

6.5 Incident correlation

The raw output of the rule engine is a flat list of findings. On a real case, this list can run into the thousands, and the operator does not want to triage them one by one. The IncidentCorrelator turns findings into actionable incidents.

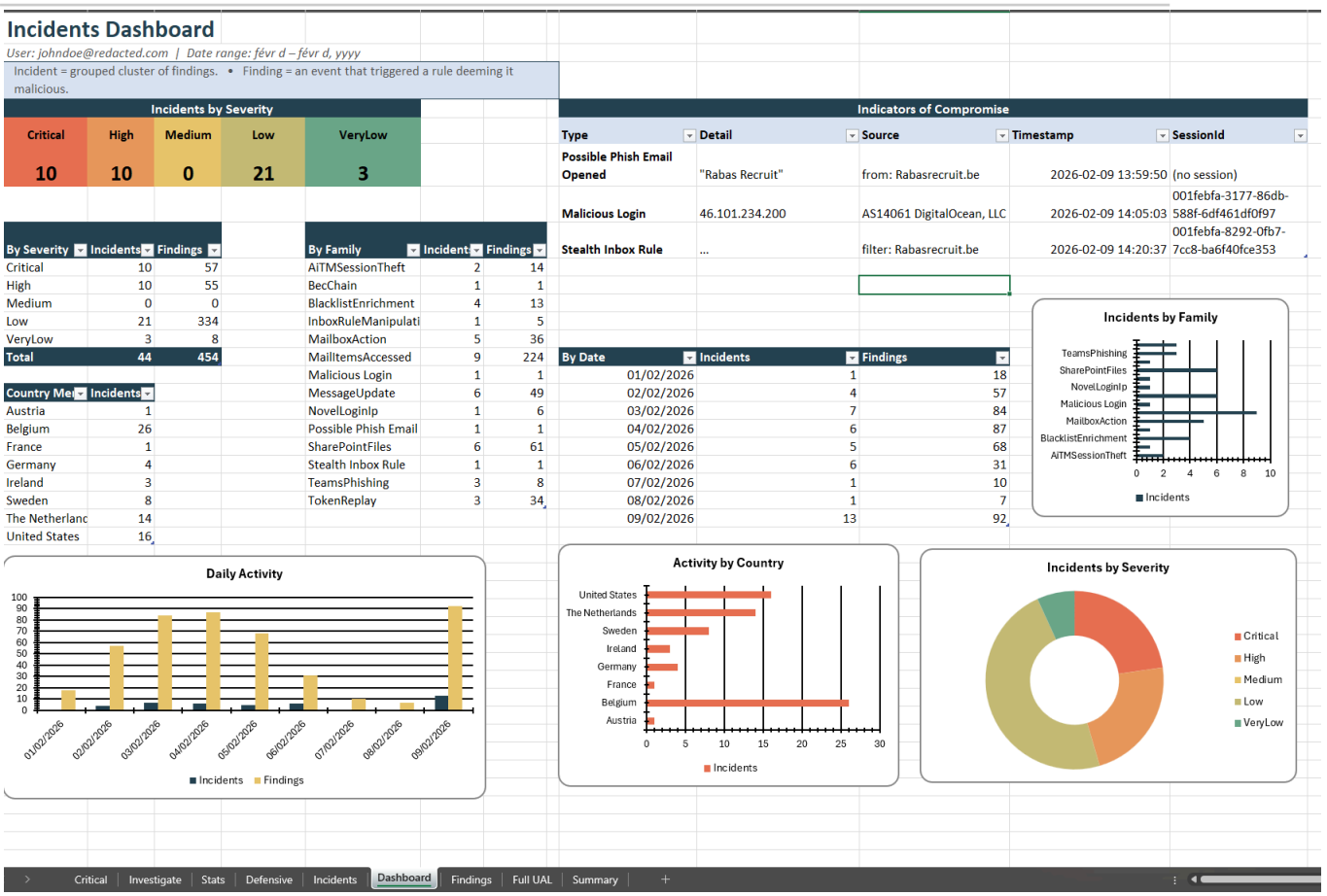
The correlation key is (UserId, UTC-day, RuleFamily). The family map contains seventeen named families, the fifteen catalogue families plus the BEC Chain Correlator and Novel Login IP components, such as InboxRuleManipulation, MailboxForwardingConfig, OAuthAppAbuse and AiTMSessionTheft. As a result, a single Set-Mailbox event that triggers three rules (LETHAL-018, LETHAL-020 and LETHAL-021) becomes one incident with all three rule identifiers attached, rather than three separate alerts. The incident's MaxSeverity is the worst-case verdict and drives the sheet on which the incident is rendered.

6.6 Output workbook layout

The workbook is designed to put the most relevant information first. The sheets, in order, are as follows. Figure 6.3 shows the resulting Dashboard sheet for a validated case.

Table 6.2: Output workbook sheet order and purpose

Sheet	Purpose
Dashboard	Visual summary with charts for the operator's first glance.
Critical	Only Critical-severity incidents.
Investigate	High and Medium incidents for manual review.
Stats	Low and VeryLow incidents aggregated by family, rule and distinct users.
Defensive	Green-tier administrative response actions for change-log purposes.
Incidents	Every incident, full table.
Findings	Every raw finding, kept for traceability.
Full UAL	Every parsed record, painted by maximum severity of any finding on that row.
Summary	A RuleId × Severity pivot table.



[Figure 6.3: The Dashboard sheet of the output workbook, showing the severity-routed incident summary and IoC table and charts for the operator's first glance.]

7 Implementation Choices

7.1 CSV parsing and AuditData flattening

CSV parsing uses CsvHelper version 30. The trickiest part is the nested AuditData JSON: the column itself is double-encoded inside the CSV escape rules, and the JSON shape varies per RecordType. The parser deserialises AuditData once with Newtonsoft.Json, then copies the fields that the downstream rules need (SessionId, ClientIP, UserAgent, MessageSubject and others) onto the flattened UalRecord. The original JSON is kept as a string for the two rules that have to inspect rare fields.

7.2 The SessionId priority chain

Session identifiers are the linchpin of AiTM and session-theft detection, but Microsoft does not place them in the same field for every workload. The parser therefore extracts SessionId through a priority chain: DeviceProperties, then the top-level field, then AppAccessContext.AADSessionId.

This change alone unblocks rule LETHAL-063 on roughly 2 700 events per typical case, events whose session marker only lives in AppAccessContext, such as SharePoint, OneDrive, Teams and Outlook-on-the-web activity. Without the priority chain, those events appear sessionless and the time-correlator misses them.

7.3 Message-subject recovery

MailItemsAccessed records do not carry the subject line of the message that was opened. They do, however, carry an InternetMessageId. Conversely, send, create, update and delete records on the same message do carry the subject. The MessageSubjectEnricher performs an in-memory join across these records and back-fills the subject. The result is that downstream rules can flag access to a message with a specific subject even when only the read event survives in the log.

7.4 Explicit severity instead of conditional formatting

This is the single cleanest improvement over the PowerShell script. Each rule assigns one of five severities (Critical, High, Medium, Low, VeryLow) to every finding it produces. The exporter then maps the enum to a fill colour at write time and routes the row to the appropriate sheet. The benefit is that severity is now a first-class field that other code can sort, filter and count on, rather than a property hidden inside an Excel formula.

7.5 Resolving the LETHAL-056 threshold drift

During the documentation phase a small but interesting discrepancy surfaced. The comment for rule LETHAL-056 (SharePoint anonymous link creation by volume) states a threshold of ten or more events per day, but the actual PowerShell code uses five or more. This discrepancy went unnoticed in the original script because the comment is never executed.

Table 7.1: LETHAL-056 threshold comparison: comment versus code versus C# port

Source	Threshold	Status in C# port
PowerShell comment	≥ 10 events/day	Not adopted
PowerShell code	≥ 5 events/day	Adopted (matches validated behaviour)

The C# port resolves the drift in favour of the code value. The reasoning is that the code is what produced the validation results the community has been relying on, and raising the threshold to the comment value would silently weaken the detection. The decision is documented in UAL-Analyzer-spec.md so that it can be revisited later.

7.6 Dropping the single-user gate

The PowerShell script aborts when the input UAL contains more than one distinct user identity. This gate was originally added as a safety check, since rules with per-day volume thresholds need a clear subject to group by. The C# port replaces the gate with a GroupBy(UserId) upstream of those rules, applying the same volume logic per user. The net effect is that the tool can be used for multi-tenant sweeps without manual file splitting.

7.7 IP enrichment, opt-in by design

IpInfoApiEnricher is asynchronous, opens at most ten concurrent HTTP connections and issues exactly one request per distinct IP. Results are cached in memory and back-written to every record that matched the same IP. The component is opt-in: the token must be supplied either through the --ipinfo-token CLI flag, the IPINFO_TOKEN environment variable, or a .env file. If none of these are present, or if the operator passes --no-ipinfo, the component is skipped entirely. This is consistent with the offline-by-default constraint inherited from the i-Force environment.

7.8 EPPlus and reproducibility

The workbook is generated with EPPlus 7.5.2. Two design rules keep the output reproducible. First, stable ordering: findings are sorted by (MaxSeverity, UserId, CreationTime, RuleId) before they are written, so running the tool twice on the same input produces rows in the same order.

Second, auto-increment file names: if ual-findings.xlsx already exists in the output folder, the next run writes ual-findings_1.xlsx, then _2.xlsx, and so on. The prior runs are never silently overwritten, which is essential when the same evidence is re-analysed multiple times.

8 Validation and Results

8.1 Test corpus

The tool was validated against two real cases from the i-Force evidence store. The first was a baseline case whose input file contained 3,394 UAL records spanning approximately 9 days of activity on a single tenant. This case was chosen because the PowerShell script had already been run against it during the active investigation, which provided a directly comparable baseline. The detailed results reported in the remainder of this chapter are drawn from this baseline case.

The second was a live case encountered in the final week of the internship, in which a company was actively compromised and the audit log was substantially larger, at approximately 1,354,341 records. This case was not chosen as a controlled baseline but served as a real-world test of the tool under operational conditions and at a scale roughly four hundred times that of the baseline.

UalBecDetector parsed the full log successfully, the analysis identified the initial phishing email, and the severity-routed dashboard surfaced the indicators of compromise without manual sorting. The case confirmed that the in-memory single-pass design holds up well beyond the record volume of the baseline case and that the triage workflow remains effective on a genuine, large-scale incident.

8.2 Outcome

Running UalBecDetector on the same input produced 454 findings across the 73 rules. The findings broke down by severity as shown in Table 8.1.

Table 8.1: Severity breakdown of the 454 findings

Severity	Count
Critical	57
High	55
Medium	0
Low	334
Very Low	8
Total	**454**

8.3 Interpretation

The shape of the distribution is what an experienced analyst would expect. The bulk sits in the Low tier, which is dominated by high-volume but individually unremarkable events such as typical sign-ins, mail-item access and file reads. The Critical and High tiers together total 112 findings across 20 incidents, a number low enough to triage manually but high enough to indicate that the rule catalogue is firing on real behaviour rather than producing silence.

Upon opening the output workbook, the investigation dashboard shown in Figure 6.3 immediately presented the 10 Critical and 10 High incidents at the top of the severity-routed view, with Indicators of Compromise surfaced without any manual sorting or filtering.

An analyst opening the workbook for the first time could triage which accounts warranted investigation within minutes rather than the several hours previously required to manually review raw PowerShell output. This ready-made reporting output is the most operationally significant improvement over the predecessor tool and was validated directly on a real BEC case at i-Force.

After incident correlation, the operator-facing unit count dropped from 454 raw findings to 20 incidents, a reduction of more than twenty to one, which matches the design intent described in section 6.5. The 5-second processing time on 3,394 records, compared to 42 minutes for the equivalent PowerShell workflow on the same input, confirmed that the in-memory single-pass pipeline delivers the performance improvement required for practical daily use in a DFIR environment.

8.4 False-positive sources

Two false-positive sources surfaced during validation. First, corporate proxy ranges: volume rules (LETHAL-046 through LETHAL-051) flagged the corporate Zscaler roaming gateway as a high-volume single source. The OrgName-AllowList.csv carve-out file was introduced to suppress this without weakening the rules for other organisations.

Second, the automated tooling user-agents: a small number of internal scripts produced user-agent strings that overlapped with patterns on the blacklist. The blacklist is now CSV-driven so that the team can extend the exclusion list without recompiling.

9 PstBecDetector: Mailbox-Level Analysis

9.1 Purpose and position within PhishTrace

While UalBecDetector answers the question of what happened at the tenant level, a BEC investigation also requires examining the actual messages that were sent, received or accessed. PstBecDetector fills this gap. It is the second tool in the PhishTrace project and scans PST and OST mailbox files offline for message-level BEC indicators. The two tools are designed to be used together: the analyst runs UalBecDetector first to identify suspicious users, timeframes and sessions, then runs PstBecDetector on the relevant mailbox exports to inspect the phishing messages themselves.

9.2 Design principles

PstBecDetector inherits the same core design principles as UalBecDetector. It runs fully offline by default, with URLScan.io enrichment as the only optional network call. It produces a reproducible Excel workbook as the primary deliverable. Rules are stateless and implement the IBecRule interface, following the same pattern as the ILethalRule contract in UalBecDetector.

The main architectural difference is the streaming pipeline. Because PST and OST files can be multiple gigabytes in size, PstBecDetector processes messages one at a time using a lazy IEnumerable rather than loading the entire mailbox into memory. Each message is analysed, its findings are recorded, and then its body cache is cleared before the next message is loaded. This keeps the working set bounded at roughly 500 MB to 1 GB even on very large mailboxes.

9.3 Detection rules

PstBecDetector ships with four detection rules, each targeting a distinct BEC indicator pattern.

SuspiciousHostingRule detects URLs hosted on commonly abused CDN and hosting platforms such as Firebase, Netlify, Vercel and GitHub Pages. It also flags high-entropy subdomains (Shannon entropy of 3.5 or above) and phishing-related path keywords.

FileShareLureRule identifies messages that combine file-sharing platform mentions (covering 29 platforms) with urgency phrases in English, Dutch and French, dangerous file-type hints such as .exe, .docm or invoice keywords, and free-email senders.

SharePointNotificationAnomalyRule detects fake SharePoint sharing notifications. It flags messages with generic call-to-action document names, external or first-contact senders, and a lack of thread context that would normally be present in genuine SharePoint notifications.

DomainReputationRule uses URLScan.io verdicts to flag domains with a malicious reputation. This rule is opt-in and includes a learning phase: the first 50 messages bootstrap a set of safe domains: those that appear in 15 or more messages or account for 2 percent or more of all messages are considered safe.

9.4 URL extraction and unwrapping

A critical capability of PstBecDetector is its URL extraction pipeline, which automatically unwraps SafeLink, ProofPoint, Mimecast, Barracuda and Trend Micro URL protection wrappers. These services encode the original URL inside a redirection wrapper, and nested wrappers are recursively unwrapped. The extracted URLs are then filtered against an allowlist (common benign domains such as w3.org, aka.ms and teams.microsoft.com) and matched against a configurable flaglist of suspicious file-sharing domains.

9.5 Output

PstBecDetector produces a set of CSV files and a consolidated XLSX workbook. The CSV outputs include url_findings (every URL found with context), domain_stats (aggregate per-domain statistics), domain_occurrences (chronological URL occurrences grouped by domain), flagged_findings (suspicious findings only) and rule_findings (detection pipeline results with severity and confidence scores). The XLSX workbook contains one formatted sheet per CSV, with table styling, auto-filters and date formatting applied via ClosedXML.

10 Reflection

10.1 What went well

Two design choices proved themselves quickly. Writing a porting specification (UAL-Analyzer-spec.md) before touching C# saved a great deal of time later in the project. Every rule, every threshold, every conditional-formatting expression and every blacklist entry was documented up-front, so the actual coding phase consisted of translating an existing precise specification rather than re-reading PowerShell for clarification. Additionally, the explicit FindingSeverity enum replaced an entire category of fragile, expression-based conditional formatting. The downstream effect on the export code, the dashboard and the incident model was much larger than the size of the change suggests.

10.2 What was hard

The hardest problem of all was detecting and fine-tuning the output. The detection initially produced a great deal of noise, flagging normal events as malicious and generating a large number of false positives. Tuning this was difficult because every BEC case is different, so there is no one-size-fits-all detection: a rule tight enough to catch a real compromise in one tenant would either miss it or drown the analyst in false alarms in another. Reducing that noise without losing genuine signal was a continuous, iterative effort throughout the project.

Beyond the tuning, two further technical problems stood out, both about consistency across schemas rather than about any single detection rule.

The first was the SessionId priority chain. Microsoft places the session identifier in three different locations depending on the workload, and the rules that depend on it (most notably the AiTM time-correlator) silently lose accuracy when even one location is missing.

The second was the message-subject recovery. The naïve approach of displaying the subject only on records that carry it left the analyst with rows full of empty placeholders on exactly the records that matter most (mail-item access). The cross-record join on InternetMessageId is conceptually simple but only became possible after the entire CSV had been parsed into memory, which in turn ruled out a streaming pipeline.

10.3 What I learned

The most generalizable lesson is that idiomatic PowerShell is not idiomatic C#. Patterns that read perfectly in shell, such as files as inter-phase IPC, conditional-formatting expressions as severity encoding and magic-number AppIds, actively get in the way of a maintainable C# port. Recognising those patterns and deciding chapter-by-chapter what to keep and what to redesign was probably the most valuable thinking I did during the project.

A second lesson is about scope discipline. The temptation when the rule engine started working was to keep adding features: real-time mode, a web dashboard, a SIEM exporter. Each would have been interesting; none was in scope. Saying no to those ideas and finishing the workbook export and the incident layer instead is what made the tool deliverable.

On the threat side, I learned a great deal about the techniques attackers use to bypass phishing filters. One example is Base64-encoding the phishing link: the link is embedded in a

form that a filter does not recognise as a URL, yet when Outlook decodes it (treating it as an image) it renders the link and makes it clickable to the victim. I also learned a great deal about email headers and how much forensic value they carry, and about the many different UAL event types and what each one reveals about an attacker's activity inside a mailbox.

10.4 Future work

Three improvements are considered realistic next steps. First, per-tenant threshold overrides: the CSV-driven config layer already exists, and extending it to host per-rule numeric thresholds would let the team tune detection without touching code.

Second, a parallel rule evaluator: the rule contract is already stateless, which means the only thing standing between the current sequential evaluator and a parallel implementation is measurement confirming that on the typical workload the speed-up is worth the added complexity.

Third, a regression-test suite: a small library of redacted UAL excerpts, each known to fire or not fire a specific rule, would catch threshold drift before it ships.

Finally, a machine-learning layer is planned as a final verification stage. Sitting after the rule engine, it would take the events the deterministic rules have flagged and assess whether a BEC actually occurred, acting as a second opinion that reduces the residual false positives the rules alone cannot eliminate.

Conclusion

This report has presented PhishTrace, a C# / .NET 10 forensic toolkit designed to detect Business Email Compromise in Microsoft 365 environments. The toolkit was developed during an internship at i-Force, a Belgian independent forensic audit firm, to address the limitations of the existing PowerShell-based workflow used by the DFIR team.

PhishTrace consists of two complementary tools. UalBecDetector analyses the tenant-wide Unified Audit Log and successfully ported the 73-rule detection catalogue from UAL-Analyzer.ps1 into a structured, maintainable codebase. The key architectural improvements, an explicit severity model, a stateless rule contract, incident correlation, configurable reference data, and a severity-routed Excel workbook, address the practical shortcomings identified in the original script. Finally, PstBecDetector complements this tenant-level view with message-level analysis, scanning PST and OST mailbox files for phishing URLs, file-share lures, fake SharePoint notifications and domain reputation signals using a memory-bounded streaming pipeline.

Validation of UalBecDetector on a real case of 3,394 UAL records produced 454 findings with a severity distribution consistent with analyst expectations. The Critical and High tiers together accounted for 112 findings and 20 incidents, a volume suitable for manual triage. After incident correlation, the operator-facing unit count dropped more than twentyfold, confirming that the correlation layer achieves its design intent.

The most valuable lesson from the project is that porting a tool between languages is not a line-by-line translation exercise. The real work lies in recognizing language-specific idioms that do not transfer well and deciding for each one whether to preserve the behavior or redesign the approach. The specification-first workflow adopted at the start of the project proved essential in making those decisions systematically rather than using an ad hoc approach.

PhishTrace is now part of the i-Force DFIR toolkit and both tools are available for use on future BEC investigations.

Reference List

[1] Federal Bureau of Investigation, Internet Crime Report 2023, Internet Crime Complaint Center (IC3), 2024.

[2] Microsoft Threat Intelligence, “From cookie theft to BEC: Attackers use AiTM phishing sites as entry point,” Microsoft Security Blog, 2022.

[3] MITRE, T1539: Steal Web Session Cookie, MITRE ATT&CK Enterprise Matrix. [Online]. Available: https://attack.mitre.org/techniques/T1539/ [Accessed: 6-Jun-2026].

[4] Microsoft, “Detect and remediate the illicit consent grant attack,” Microsoft Learn, Defender for Office 365, 2024. [Online]. Available: https://learn.microsoft.com/en-us/defender-office-365/detect-and-remediate-illicit-consent-grants [Accessed: 6-Jun-2026].

[5] Microsoft, “Phishing investigation, Inbox rules,” Microsoft Learn, Security operations, 2024. [Online]. Available: https://learn.microsoft.com/en-us/security/operations/incident-response-playbook-phishing [Accessed: 6-Jun-2026].

[6] Microsoft, “Search the audit log in the compliance portal,” Microsoft Learn, Microsoft Purview, 2024. [Online]. Available: https://learn.microsoft.com/en-us/purview/audit-log-search [Accessed: 6-Jun-2026].

[7] MITRE, ATT&CK Enterprise Matrix, version 14, 2023. [Online]. Available: https://attack.mitre.org/versions/v14/matrices/enterprise/ [Accessed: 6-Jun-2026].

[8] M. Willing, UAL-Analyzer.ps1, Lethal-Forensics, 2025. [Online]. Available: https://lethal-forensics.com/

[9] Invictus Incident Response, Microsoft-Extractor-Suite, v4.0.0, 2024. [Online]. Available: https://github.com/invictus-ir/Microsoft-Extractor-Suite

[10] ipinfo.io, Lite API reference, 2024. [Online]. Available: https://ipinfo.io/developers/lite-api [Accessed: 6-Jun-2026].

[11] EPPlus Software, EPPlus 7 documentation, 2024. [Online]. Available: https://epplussoftware.com/docs/7.0/ [Accessed: 6-Jun-2026].

[12] A. Cidon, L. Gavish, I. Bleier, N. Korshun, M. Schweighauser and A. Tsitkin, “High precision detection of business email compromise,” in Proc. 28th USENIX Security Symposium (USENIX Security 19), Santa Clara, CA, USA, 2019, pp. 1291–1307.

[13] N. S. Al-Musib, F. M. Al-Serhani, M. Humayun and N. Z. Jhanjhi, “Business email compromise (BEC) attacks,” Materials Today: Proceedings, 2021, doi: 10.1016/j.matpr.2021.03.647.