electronics

MDPI

Article

I Know What You Played Last Summer: Evaluating the Feasibility of Privacy Attacks in Massively Multiplayer Online Role-Playing Games

Parisa Rahimi [1,*], George Spary [1], Amit Kumar Singh [2], Seyedali Pourmoafi [1], Xiaohang Wang [3] and Alexios Mylonas [1]

[1] School of Physics, Engineering and Computer Science, College Lane Campus, University of Hertfordshire, Hatfield AL10 9AB, UK; g.spary@herts.ac.uk (G.S.); s.pourmoafi@herts.ac.uk (S.P.); a.mylonas@herts.ac.uk (A.M.)
[2] School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK; a.k.singh@essex.ac.uk
[3] College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China; xiaohangwang@zju.edu.cn
* Correspondence: p.rahimi@herts.ac.uk

Abstract

Massively Multiplayer Online Role-Playing Games (MMORPGs) increasingly rely on player-developed third-party tools to extend functionality and personalise gameplay, creating a complex software ecosystem that introduces both usability benefits and security risks. This study investigates whether such tools can be exploited as an attack vector for cybercrime by designing and implementing a proof-of-concept add-on within a widely deployed commercial MMORPG using its native scripting and application programming interface. The developed tool supports automated player discovery, chat capture, target inspection, and local data persistence, enabling a systematic evaluation of how cyber-assisted and cyber-dependent crimes could be facilitated within the game client. Empirical testing demonstrates that while the platform's protected execution model and interface restrictions prevent direct credential theft and remote code execution, the add-on architecture allows extensive behavioural data collection and social-engineering-relevant monitoring, making several forms of cyber-enabled crime technically feasible. These findings show that MMORPG add-on frameworks represent a non-trivial socio-technical attack vector in next-generation online platforms, where security depends not only on code isolation, but also on how user-generated extensions interact with human behaviour. The results highlight the need for architecture-aware security controls and governance mechanisms to mitigate emerging threats in large-scale, extensible virtual environments.

Keywords: cyber security; cybercrime; third-party tools; MMORPGs; software ecosystems; game add-ons; socio-technical security

Academic Editor: Juan-Carlos Cano

Received: 16 February 2026
Revised: 10 April 2026
Accepted: 27 April 2026
Published: 29 April 2026

1. Introduction

In recent years, artificial intelligence systems and large-scale digital platforms have become increasingly embedded in everyday life, shaping communication, information exchange, and social interaction on a global scale. Alongside these developments, massively multiplayer online role-playing games (MMORPGs) have evolved into complex socio-technical environments that combine interactive systems, user-generated content, and extensible software architectures [1,2]. A key feature of many modern MMORPGs is the

integration of officially supported add-on mechanisms and application programming interfaces (APIs), which allow users to customise gameplay and extend functionality within controlled and sandboxed environments [3].

These add-on systems are typically designed with strong security constraints. They operate within restricted execution environments, do not allow direct access to memory or network layers, and are governed by platform policies intended to prevent exploitation, reverse engineering, or unauthorised data access [3,4]. As a result, they are widely perceived as secure-by-design mechanisms that balance flexibility with control. However, this perception does not fully consider the implications of data exposure through legitimate and policy-compliant interfaces.

The central problem addressed in this paper is that officially supported add-on systems may expose user-visible metadata which, although individually limited and benign, can be aggregated over time to reveal meaningful patterns of user behaviour. As demonstrated in the associated project work, such aggregation can enable the inference of contextual and behavioural information without requiring any form of system compromise, code injection, or violation of platform restrictions. This raises important questions regarding privacy, data exposure, and the broader responsibilities of platform providers when designing extensible systems [5].

To investigate this problem, this study is guided by three research questions. First, what categories of user-related data are exposed through official MMORPG add-on APIs and event systems? Second, to what extent can this data be collected and aggregated longitudinally within the constraints of the platform? Third, what forms of behavioural or contextual inference can be derived from such aggregated data, and what are the associated privacy and security implications?

Existing research in cybersecurity and software systems has extensively examined vulnerabilities, exploits, and malicious behaviours, including client modification, unauthorised access, and cheating mechanisms in online environments [6,7]. Similarly, prior studies have investigated privacy risks in digital platforms, particularly in relation to large-scale data collection, tracking, and profiling [8,9]. However, comparatively limited attention has been given to the risks that emerge from legitimate, policy-compliant use of officially exposed APIs in interactive systems such as MMORPGs. In particular, the potential for longitudinal aggregation of seemingly harmless data to enable behavioural inference remains underexplored.

This gap is especially significant in environments where user interaction is continuous, data is generated in real time, and APIs expose structured event-driven information. While sandboxing mechanisms effectively restrict direct exploitation, they do not necessarily prevent indirect inference through aggregation and correlation of available data [4,10]. The present study addresses this gap by systematically examining the extent, limitations, and implications of data exposure within an MMORPG add-on ecosystem, building on empirical observations and implementation details presented in the main project report.

The contribution of this paper is therefore twofold. First, it provides an empirical analysis of what data can be accessed through officially supported add-on systems and how this data can be structured and analysed over time. Second, it offers a critical discussion of the privacy implications of such exposure, highlighting how legitimate system features may introduce unintended risks when considered from a longitudinal and analytical perspective.

Existing research on abuse in MMORPG environments has primarily focused on behaviours that violate platform rules, such as botting, cheating, and unauthorised client modification. While these studies provide important insight into technical and gameplay exploitation, they largely examine scenarios in which system safeguards are bypassed. In contrast, this study examines behaviour that operates entirely within the officially

supported add-on framework, without circumventing sandbox restrictions or modifying system components. Similarly, prior work on plugin and extension security has predominantly focused on permission models, code-level vulnerabilities, and privilege escalation mechanisms [11]. While these approaches address technical compromise, they do not fully capture how policy-compliant access to user-visible data can enable behavioural inference and profiling.

The contribution of this work is therefore not in demonstrating new forms of technical exploitation, but in showing how cyber-enabled misuse can emerge through the structured aggregation of accessible data within a constrained environment. This positions MMORPG add-on ecosystems as socio-technical systems in which security is shaped not only by code isolation, but also by the nature, persistence, and combinability of exposed information.

Recent research on privacy and security in consumer systems has explored the use of advanced cryptographic techniques, such as hyperchaos-based image encryption, to protect sensitive data from unauthorised access [11]. These approaches focus on securing data through transformation and controlled access mechanisms. In contrast, this study highlights a complementary challenge: the risks associated with data that is already accessible within system boundaries and can be aggregated over time. This distinction reinforces the need to consider both technical protection and information exposure when evaluating the security of extensible platforms.

The remainder of this paper is structured as follows. Section 2 reviews related work on add-on ecosystems, extension security, and data exposure in interactive systems. Section 3 presents the methodology, including the design and constraints of the implemented add-on and the data collection process. Section 4 reports the experimental results and observed patterns. Section 5 discusses the implications of these findings in relation to privacy, security, and system design. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Background

Since the term Massively Multiplayer Online Role-Playing Game was introduced in 1997, MMORPGs have evolved into some of the most complex interactive digital systems in existence [1]. Their origins lie in tabletop role-playing games such as *Dungeons and Dragons* and fantasy literature, but they have since diversified into numerous genres, including science fiction, cyberpunk, and horror-themed worlds [1]. Despite this diversity, all MMORPGs share a common structure: persistent virtual environments in which large numbers of players interact, cooperate, and compete through networked software.

Modern MMORPGs support a wide range of activities, including player-versus-environment (PvE) combat, player-versus-player (PvP) competition, crafting, trading, and social role-play. Many of these activities require sustained coordination between players, particularly in group-based encounters such as dungeons and raids, where failure by a single participant can compromise the success of the entire group. This creates strong incentives for players to seek tools that improve situational awareness, coordination, and efficiency.

Although developers attempt to provide comprehensive in-game functionality, player needs are often not fully met by the base game. As a result, players turn to external tools and extensions to customise and enhance their experience [2]. These tools can range from simple interface modifications to highly complex automation systems. While this extensibility increases player engagement and innovation, it also introduces new risks, as the same mechanisms that enable beneficial customisation can be exploited for harmful purposes, including cybercrime

2.1. Cybercrime in Digital Environments

Cybercrime refers to criminal activity in which computers are either the primary target or an essential component of the offence [3]. It is commonly classified into cyber-dependent crime, cyber-enabled crime, and cyber-assisted crime [3].

Cyber-dependent crime includes offences that cannot exist without computing systems, such as hacking, distributed denial-of-service attacks, and malware deployment, including trojans and ransomware [4,5]. These crimes typically involve exploiting software vulnerabilities or persuading users to execute malicious code. Cyber-enabled crime refers to traditional crimes that are significantly amplified by digital technologies. This category includes phishing, fraud, harassment, and cyber-stalking, where online platforms allow offenders to reach more victims, collect more information, and act with greater persistence and anonymity [4,5]. In MMORPGs, cyber-enabled crime is particularly relevant because players maintain persistent identities, communicate continuously, and often invest significant time and emotional value in their characters [6].

Cyber-assisted crime involves the use of digital systems to support conventional criminal activity, such as using online messaging platforms to plan or coordinate illegal actions [3]. In practice, this is less relevant to MMORPGs, as external platforms such as WhatsApp provide encrypted and unmoderated communication channels that are more attractive for such purposes [7]. Despite the scale and social complexity of virtual worlds, research into cybercrime in MMORPGs remains limited. A recent literature review identified fewer than a dozen academic studies addressing cybercrime in virtual environments over more than three decades [8]. This gap suggests that important aspects of how crime manifests in online games, particularly through platform-provided extension mechanisms, remain poorly understood.

2.2. Cheating and Botting as Related Risk Vectors

Cheating is widespread in online games and can be defined as a player obtaining an unfair advantage by modifying or circumventing the intended rules of the game [9]. In MMORPGs, cheating often takes the form of automation, exploitation of bugs, or engagement in real-money trading (RMT), where in-game items or services are exchanged for real-world currency [10]. RMT frequently relies on botting, which involves the use of scripts or programs to automate gameplay tasks such as resource gathering or combat [6]. Botting is prohibited by most MMORPG terms of service, including the WoW End User License Agreement [12], yet it remains prevalent due to the financial incentives involved.

Research into cheating and botting is extensive, particularly in the area of detection and mitigation [13]. Importantly, cheating and cybercrime are often intertwined. Botting, account theft, and fraud are routinely used to support RMT operations, creating pathways through which cybercrime can emerge within game environments.

2.3. World of Warcraft as a Case Study

World of Warcraft (WoW) is one of the largest and most enduring MMORPGs, with a long-standing active player base and a mature ecosystem of third-party add-ons. Its history includes numerous emergent behaviours and incidents, such as the "Corrupted Blood" event, which demonstrated how in-game systems can produce complex and unintended social and technical effects [14]. WoW also illustrates how third-party tools can shape player behaviour. The GearScore add-on, for example, became an informal standard for evaluating player competence during the *Wrath of the Lich King* expansion, influencing group formation and social dynamics in ways not intended by the developers [15].

Crucially, WoW formally permits add-ons through its End User License Agreement, provided they do not constitute cheating [13]. Surveys indicate that more than 75% of WoW

players use add-ons to customise their interface and gameplay experience [16], making WoW an ideal platform for studying how widespread TPT usage interacts with security and privacy risks.

2.4. Third-Party Tools and Add-On Architectures

Third-party tools in MMORPGs are typically implemented as embedded extensions that run within the game client using developer-provided scripting environments and APIs. In WoW, add-ons are written primarily in Lua and interact with the game through the Blizzard API, which exposes events, player data, and user-interface elements while restricting access to sensitive system functions [17].

Lua is particularly well suited to this purpose because it allows deep integration with the game interface without requiring invasive techniques such as dynamic-link library injection, which are more likely to trigger security mechanisms or violate platform policies [18,19]. Visual elements are typically defined using XML, allowing add-ons to create buttons, frames, and hidden interface components that can interact with players in subtle ways. While this architecture supports innovation and usability, it also creates opportunities for abuse. A malicious add-on could hide elements from the user, silently collect behavioural data, or manipulate interactions in ways that are difficult for players to detect [20].

2.5. Cyber-Stalking and Harassment in Virtual Worlds

Cyber-stalking refers to the repeated use of digital technologies to monitor, follow, or harass an individual, often with the intent to intimidate, distress, or control them [21]. Although it is most commonly associated with social media platforms, MMORPGs provide similar conditions through persistent identities, real-time communication, and shared virtual spaces. In a game environment, cyber-stalking may involve tracking a character's movements, monitoring their interactions, or repeatedly appearing in their vicinity to harass or intimidate them. Such behaviour is often accompanied by cyber-harassment or cyber-bullying, which can have significant psychological impacts on victims [22]. While substantial research exists on these phenomena in social media, comparatively little attention has been given to their manifestation in online games.

2.6. Fraud and Social Engineering in MMORPGs

Fraud involves deceiving a victim in order to obtain money, assets, or personal information [23,24]. In digital environments, this is often achieved through social-engineering techniques such as phishing, impersonation, or the provision of false information.

In MMORPGs, fraud typically manifests as account theft, item scams, or impersonation attacks. However, social-engineering threats in online games are becoming increasingly sophisticated, reflecting broader trends in cybercrime [25]. Because players frequently trust others within their virtual communities, and because valuable digital assets are at stake, MMORPGs provide fertile ground for deceptive practices.

2.7. Data Exposure and Inference Risks in Extensible Platforms

A useful comparative perspective can be drawn between MMORPG add-on ecosystems and other extensible software environments, particularly browser extension and plugin architectures. Although these systems differ in purpose, interface design, and operational context, they share an important structural characteristic: third-party code is permitted to operate within a constrained and officially supported environment, while being granted selective access to user-facing data, events, or interaction contexts.

In browser and plugin security research, a recurring concern is that technical restrictions may successfully prevent direct compromise of the host system while still permitting

forms of observation, aggregation, or inference that create privacy risk. This perspective is relevant to the present study. The World of Warcraft add-on model, as examined here, demonstrates a similar pattern in which strong sandboxing and API constraints prevent cyber-dependent attacks such as arbitrary code execution or credential theft, yet still permit the systematic collection of behavioural metadata through legitimate mechanisms.

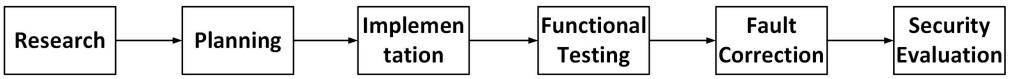
The relevance of this comparison lies not in claiming that MMORPG add-ons and browser extensions are functionally identical, but in highlighting a broader architectural issue common to extensible platforms. In both cases, the security model may be effective in preventing overtly malicious technical actions, while remaining less effective against privacy-relevant inferences derived from cumulative access to exposed data. This strengthens the argument that the risks identified in this study should be understood not only as a property of one game environment, but as part of a wider class of socio-technical concerns associated with extensible systems.

3. Methodology

This study employed a mixed-method, design-science methodology to examine whether third-party tools (TPTs) can be used to facilitate cyber-enabled activities within a mainstream MMORPG. The research combined empirical add-on development with systematic experimental evaluation conducted using controlled accounts and the platform's officially supported add-on interfaces. All experiments were performed within the constraints of the platform's sandboxed architecture and without bypassing technical safeguards, modifying client software, or interacting with uninvolved players. The evaluation therefore reflects the practical capabilities and limitations of third-party tools operating under the same conditions available to ordinary users.

3.1. Development Process and Research Workflow

An agile sprint methodology was adopted to support iterative feature development, validation, and refinement. Each sprint was one week in duration and focused on a single capability relevant to cybercrime enablement. The core of each sprint followed the process illustrated in Scheme 1.



Scheme 1. Software development workflow from research to security evaluation.

During the research phase, platform documentation, community forums, and developer resources (e.g., WoW API references and Wowpedia) were analysed to determine which data sources, events, and commands were accessible within the add-on sandbox [19,20]. This ensured that each proposed feature was both technically feasible and aligned with real-world TPT development practices. The planning phase defined how the selected API calls, UI elements, and data structures would be combined to achieve the desired functionality. Implementation was then performed in Lua using WoW's native API, followed by incremental testing to ensure stability and correctness before full validation.

This iterative approach allowed platform limitations, security restrictions, and edge-case behaviours to be discovered and incorporated into subsequent design decisions.

3.2. System Design and Threat-Driven Feature Selection

To provide a clear foundation for the experimental design, the threat model in this study is defined based on the capabilities demonstrated within the implemented add-on and the constraints of the platform.

The primary adversarial scenario considered is that of a standard player utilising an add-on developed using officially supported APIs. In this setting, the actor does not modify the game client, intercept network traffic, or perform any unauthorised actions. Instead, the add-on operates entirely within the sandboxed environment provided by the platform and accesses only user-visible data exposed through event-driven mechanisms. This reflects the behaviour implemented in the prototype, where data such as player presence, identifiers, and interaction events are collected and stored during normal gameplay.

Within this context, data collection can occur over multiple gameplay sessions, allowing information to be aggregated over time. While the study evaluates this behaviour using controlled accounts, the same mechanism can be conceptually extended to scenarios where data is collected more persistently or across multiple instances. However, such extensions are not experimentally evaluated in this work and are considered only to contextualise the potential implications of the approach.

The effort required to implement this method is limited, as it relies on documented APIs and standard add-on functionality. No specialised privileges or external tools are required beyond those available to regular users. The scalability of the approach is constrained by factors such as session duration, frequency of observable events, and the number of accounts used for data collection, all of which are intentionally limited in this study to maintain controlled conditions.

In terms of detectability, the approach operates entirely within policy-compliant behaviour. Since the add-on does not perform intrusive actions or access restricted resources, its operation is consistent with typical add-on usage. As a result, detection is not directly addressed within the scope of this study, although large-scale or prolonged data aggregation may introduce observable patterns at the platform level.

This threat model reflects the practical conditions under which the implemented approach operates and ensures that the analysis remains grounded in realistic and policy-compliant system behaviour. The final TPT implemented four functional modules:

- **Player discovery** via automated/who queries, capturing character names, levels, and timestamps.
- **Chat logging** for selected in-game channels (e.g., SAY and PARTY), enabling the capture of player communications.
- **Target inspection**, allowing metadata associated with selected player characters (e.g., character name, in-game race as defined by the game mechanics, and platform-exposed character identifiers) to be recorded.
- **Note-taking and data persistence**, enabling the investigator to attach timestamped annotations to collected records.

Each module stored its output in structured tables that could be printed, cleared, or exported, enabling real-time inspection and offline analysis during controlled experimentation. All recorded data was limited to platform-exposed, fictional in-game character attributes and technical identifiers obtained from controlled test accounts, and no personal data relating to identifiable individuals was collected or retained. These capabilities were selected to evaluate how permitted add-on functionality could be repurposed to support cyber-enabled activities such as stalking, profiling, harassment, and social-engineering preparation within the game environment.

3.3. Platform, Tools, and Programming Environment

World of Warcraft was selected as the experimental platform due to its large user base, mature add-on ecosystem, and permissive End User License Agreement, which explicitly allows third-party add-ons provided they use the official API.

The TPT was implemented using Lua as the scripting language, leveraging the World of Warcraft API for event handling, controlled data access, and user interface construction. Add-on configuration and lifecycle management were handled through standard TOC files, while all collected datasets were stored locally using the platform's SavedVariables mechanism, which provides persistent client-side storage within the constraints of the official add-on framework.

Lua was chosen because it is the only officially supported language for WoW add-ons and operates entirely within Blizzard's sandbox, avoiding invasive techniques such as DLL injection that would invalidate the security model under study [18].

3.4. Add-On Implementation and Functional Design

The proposed third-party tool was implemented as a World of Warcraft add-on written in Lua and packaged using the standard add-on configuration format. The add-on metadata is specified in the .toc file, which defines the client interface version, add-on title, versioning, and the persistent storage variables used by the implementation. The add-on loads a single primary Lua script (*Cyber_Tool.lua*) and declares saved variables to support persistence of selected state across gameplay sessions.

The implementation follows an event-driven design aligned with the World of Warcraft add-on execution model. The add-on registers handlers for gameplay events that are relevant to the studied capabilities. Specifically, an event frame is registered to detect target changes (*PLAYER_TARGET_CHANGED*) to support collection of target-related metadata when the user interacts with nearby entities. A separate event handler is registered for chat message events (*CHAT_MSG_SAY* and *CHAT_MSG_PARTY*) to evaluate the feasibility of capturing limited communication metadata available through the official API. In addition, an initialisation frame is registered for the environment load event (*PLAYER_ENTERING_WORLD*) to provide user guidance on available commands at runtime. To support structured data aggregation, the add-on maintains in-memory tables for each capability and uses counters to append records sequentially. The implementation includes dedicated tables for storing: (i) previous "who" search queries, (ii) target-derived metadata captured upon a target-change event when logging is enabled, (iii) chat message records captured from selected channels when a target filter is configured, and (iv) user-generated notes. These structures allow the tool to evaluate the difference between transient observation and persistent aggregation, which is central to assessing cyber-enabled misuse potential.

3.4.1. Feature Set and User Interaction Model

The add-on exposes functionality through two interaction modes: slash commands and a graphical button-based interface. Slash commands provide direct invocation of core functions for searching, logging, viewing stored records, and clearing stored data. The graphical interface provides equivalent capabilities through on-screen buttons and contextual labels, allowing the evaluation to consider usability and the extent to which an average user can operate the tool without specialised technical knowledge.

The add-on also includes pop-up dialogue windows to support data entry and structured extraction of stored records. For example, input dialogs are used to capture a search string for in-game "who" queries and to define the current target used for message filtering. Extraction dialogs display concatenated stored records within an edit box to enable copying of stored content. This design choice enables a practical evaluation of persistence and exportability within the constraints of the sandboxed add-on environment, without relying on external automation or modification of the game client.

3.4.2. Data Persistence and Storage Controls

Persistence is implemented using the platform's saved variables mechanism, which writes declared variables to a local file on the client system. This provides a constrained but effective mechanism for retaining data across sessions. The add-on includes explicit functions to clear each storage table, allowing controlled experimentation and repeatable trials by resetting state between runs. In the evaluation, this storage and reset mechanisms were used to measure what information remains accessible over time and how reliably the add-on can re-surface previously collected records after the client is restarted.

3.4.3. Implementation Constraints and Scope of Data Access

All data collection in this study is strictly limited to information exposed through the official add-on API and event system provided by the platform. The add-on does not modify client binaries, inject code into the game process, intercept network traffic, or perform any form of external communication. Observed limitations are enforced by the platform's sandbox design, including API-level protections, event visibility constraints, and channel-based access controls. Consequently, the implementation is not intended to demonstrate technical exploitation or system compromise, but rather to examine what categories of user-facing metadata are exposed by design and how their aggregation may give rise to residual privacy and cyber-enabled risks, even when direct technical attacks are effectively mitigated.

3.5. Add-On Implementation Details

While Section 3.4 describes the functional behaviour of the prototype, further implementation detail is provided below to support reproducibility and clarify how these behaviours are realised within the platform constraints.

To improve transparency and reproducibility, the implementation of the prototype add-on is described in terms of its structure, event-driven behaviour, and data storage mechanisms.

The add-on is implemented as a standard World of Warcraft add-on consisting of a Table of Contents (TOC) file and a Lua script. The TOC file defines metadata and declares persistent saved variables (*targetWho*, *currentTarget*), which are used to retain information across gameplay sessions. The Lua script contains all functional logic and operates entirely within the officially supported add-on API.

The prototype follows an event-driven design. Specific gameplay events are registered using frame-based handlers, including *PLAYER_TARGET_CHANGED*, *CHAT_MSG_SAY*, *CHAT_MSG_PARTY*, and *PLAYER_ENTERING_WORLD*. These events trigger the collection of information that is already accessible through the game interface. For example, when a player target changes, the add-on retrieves the target's GUID and associated attributes using API functions such as *UnitGUID* and *GetPlayerInfoByGUID*, enabling structured recording of player information.

User interaction is provided through both slash commands and graphical interface elements. The add-on supports commands for performing/who searches, storing and reusing search criteria, toggling target-based data collection, setting message capture targets, and managing stored data. These functions are also exposed through on-screen buttons to ensure accessibility for non-technical users.

Data collected through these interactions is stored in a set of structured tables. The *whoStorage* table records search queries used for player discovery. The *onTargetStorage* table stores information obtained through target inspection, including GUID-linked attributes and timestamps. The *messageTable* stores captured chat messages along with associated

metadata such as author, channel, and identifiers. The *notesStorage* table allows the user to create and retain contextual annotations linked to observed behaviour.

Importantly, the prototype does not introduce new data sources or bypass platform restrictions. All collected information is derived from user-visible data exposed through the official API. The functionality of the add-on therefore arises from the aggregation, persistence, and organisation of accessible data, rather than from technical exploitation of the platform.

3.6. Experimental Testing Strategy

Two forms of testing were conducted:

3.6.1. Sprint-Level Testing

Each feature was validated during development through debugging, UI verification, and live interaction with the game client to confirm basic functionality before formal evaluation.

3.6.2. Unit-Style Functional Testing

Formal test cases were defined for each functional module in order to evaluate its behaviour under controlled and repeatable conditions. For each test, specific input parameters were selected, such as character names, in-game chat activity, and */who* search queries, and the corresponding expected system responses were determined in advance. The add-on was then exercised using these inputs, and the actual outputs produced by the client were recorded and compared against the expected outcomes to identify both correct operation and platform-imposed limitations.

All testing was conducted using two controlled World of Warcraft accounts operating on the same server. A free-trial account was used to establish baseline behaviour under the most restrictive platform conditions, while a paid subscription account was introduced to verify how functionality changed when access to additional social and search features was enabled. This dual-account setup allowed subscription-related constraints to be isolated from architectural or implementation-related effects.

3.7. Validation and Experimental Controls

To ensure consistency and comparability across all experiments, a fixed set of parameters was maintained throughout the testing process. All tests were conducted on the same server, using the same character identities and communication channels, with every interaction recorded using time-stamped logs generated by the add-on. This controlled configuration ensured that variations in the observed behaviour could be attributed to the functionality under evaluation rather than to environmental differences.

The controlled use of two World of Warcraft accounts on a single server was chosen to support internal validity and repeatability rather than broad external generalisation. This configuration allowed the study to observe the behaviour of the implemented add-on under stable and known conditions, while also distinguishing between architectural constraints and account-policy restrictions through the comparison of a free-trial account and a paid account. In particular, this design made it possible to identify which limitations were imposed by the platform's sandbox and which arose from subscription-level access controls. For a proof-of-concept feasibility study, this level of control was necessary to ensure that the observed outcomes could be attributed to the platform and prototype behaviour rather than to uncontrolled environmental variation.

To ensure that the evaluation was conducted in a systematic and controlled manner, each capability was tested through repeated trials under consistent conditions using the same pair of accounts and server environment. For each feature, multiple executions were

performed to confirm that the observed behaviour was reproducible and not the result of isolated or transient conditions. Observation windows were maintained for sufficient duration to capture event-driven outputs, including chat messages, player targeting updates, and/who query responses.

The timing of/*who* queries was controlled to reflect realistic interaction patterns while avoiding platform rate-limiting behaviour. Queries were issued at regular intervals rather than continuously, ensuring that responses were obtained under normal operating conditions of the game client.

Testing was conducted across a range of typical gameplay scenarios, including general open-world exploration, proximity-based player interaction, and participation in common communication channels such as local (/*say*) and group (/*party*) chat. These contexts were selected to reflect standard player behaviour rather than edge-case conditions.

To support consistent interpretation of results, each evaluated capability was classified according to the following criteria. A capability was considered fully feasible if it could be reliably executed using only officially supported API functionality, with consistent and repeatable results across trials. It was considered partially feasible if the behaviour could be achieved but was constrained by platform limitations, such as restricted automation, incomplete data access, or inconsistent triggering conditions. A capability was classified as blocked if it could not be implemented due to enforced restrictions within the add-on environment, such as protected execution, lack of API access, or client-side safeguards.

The outputs produced by each feature were then compared against the predefined expected outcomes to determine whether the corresponding cybercrime-relevant capability was fully operational, partially constrained, or entirely blocked by the platform. Where discrepancies were observed, further analysis was performed to establish whether these resulted from implementation faults within the add-on or from security restrictions enforced by the World of Warcraft add-on architecture.

3.8. Practical Constraints and Platform Safeguards

Several architectural safeguards limited the scope of feasible attacks. WoW enforces protected functions that prevent add-ons from sending arbitrary messages, accessing external links, performing file I/O, or operating before the player enters the game world. This prevents keylogging, credential theft, malware delivery, and remote command execution. Free-trial account restrictions further limited visibility in/*who* searches and chat reception, although these constraints were shown to disappear for paid accounts, confirming that the observed limitations were policy-based rather than architectural.

These constraints demonstrate that while cyber-dependent crimes are effectively blocked, the platform still allows extensive behavioural surveillance and data collection that can support cyber-enabled crimes.

3.9. Ethical Considerations

This study was conducted as a controlled technical investigation of the World of Warcraft add-on environment using researcher-managed accounts. No human participants were used in the project. All experimental testing was performed using two controlled World of Warcraft accounts operating on the same server under fixed and repeatable conditions.

All data collection was limited to information exposed through the official add-on API and event system provided by the platform. The implemented add-on did not modify client binaries, inject code into the game process, intercept network traffic, or perform any form of external communication. The study was therefore designed to examine the exposure of user-facing metadata within the documented sandbox environment, rather than to demonstrate technical exploitation or system compromise.

Because the work did not involve human participants or participant-generated research procedures, formal ethical review was not required in the conventional sense associated with human-subject studies. The ethical responsibility of the work instead lies in how the technical findings are framed and communicated. For this reason, the manuscript presents the results at an analytical level and emphasises their value for understanding privacy exposure, socio-technical risk, and platform design considerations in extensible digital systems.

4. Results

This section presents the experiential evaluation of the developed third-party tool and examines which capabilities remain feasible within the official add-on sandbox environment. The following subsections analyse three key aspects: data collection and profiling capabilities, real-time tracking and monitoring behaviour, and the persistence of stored information across gameplay sessions. These results are important because they allow us to determine which forms of cyber-enabled or cyber-assisted misuse remain practically achievable without violating the platform's architectural protections. By systematically evaluating each capability, the study identifies the functional boundaries of the sandbox model and highlights the specific mechanisms through which user information can still be exposed or aggregated.

To support a consistent interpretation of the results, data exposure in this study is defined as the extent to which player-related metadata becomes accessible and usable through the add-on within the official API environment. In this context, exposure refers not only to whether information can be observed during gameplay, but also to whether it can be recorded, linked across events, and retained over time. This includes functionality demonstrated in the study, such as player discovery through search queries, chat capture and logging, GUID-based tracking, and structured note storage. While each of these features provides access to limited information in isolation, their combined use enables the aggregation of data across gameplay sessions, increasing its persistence and practical utility. Under this definition, the results presented in Section 4.3, including the examples illustrated in Figure 6 and subsequent analyses, should be understood as demonstrating how accessible data can transition from transient observation to structured and persistent representation.

4.1. Deployment and Interface Functionality

To ensure that the evaluation extends beyond a pure implementation effort, the behaviour of the developed tool was compared against the default gameplay environment in which no add-on functionality was enabled. This comparative approach allows the study to determine whether the observed capabilities represent a genuine increase in accessible information rather than behaviour that is already available through standard gameplay features. In addition, where applicable, observations were compared across different account configurations in order to distinguish between limitations imposed by platform architecture and those enforced through policy or subscription restrictions (Figure 1).

This takes the form of either commands that are able to be performed in the console, or via the drop-down icons that appear on screen. Assuming that not everyone will have the technical knowledge to use the available commands correctly, and knowing a TPT can be used by anyone, the creation of this function allows all users to understand what they might be able to do at a glance. Each of these functions has been built on the concept of 'Frames', which are used in-game as a way to interact with the screen. Each frame can be purposed to do various things, such as: show text on the screen, handle an event that occurs in-game, act as a button, and so on. Without these 'Frames', the code would be far

more limited in what could be done without extensive XML to account for the visual and practical elements (Figure 2).

Figure 1. The console in-game with the successful onload command. Taken in World of Warcraft (Lua 5.1).

```
--Creating frame for events and Registers an event for when a player changes their target
local Target_EventFrame = CreateFrame("Frame");
Target_EventFrame:RegisterEvent("PLAYER_TARGET_CHANGED");
```

Figure 2. An example of a Frame being created using code. Taken in Sublime Text 4.

User interaction with the TPT was provided through both slash commands and a graphical interface based on WoW's frame system. Frames were used to create buttons, text displays, event handlers, and interactive pop-ups, allowing all features to be accessed without technical knowledge of command syntax. The interface presented the available functions in a vertically organised and clearly labelled drop-down menu, shown in Figure 3, enabling rapid activation and monitoring of all data-collection modules.

Figure 3. The drop-down menu opened up, showing all features in the current build of the TPT. Taken in World of Warcraft.

4.2. Player Discovery via/Who Queries

The first operational feature exploited WoW's built-in/*who* command, which returns information about online characters matching specified criteria such as name, location, or level. TPT automated these queries and stored the results in a persistent table along with timestamps, as shown in Figures 4 and 5. Users could print the contents of this table, clear it, or export it for offline analysis.

Figure 4. The in-game Who List menu, Taken form World of Warcraft.

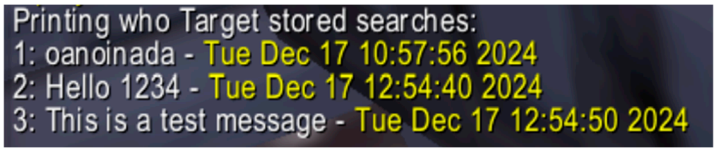


Figure 5. Information example from Who Feature.

Empirical testing demonstrated that repeated use of the/*who* search functionality can be used to infer the presence and movement of specific player characters across zones and over time. By issuing successive queries with controlled parameters and associating results with timestamps, it was possible to reconstruct coarse-grained activity patterns for individual characters. While this behaviour does not involve technical exploitation or circumvention of platform safeguards, it highlights how officially exposed metadata may be aggregated over time, raising privacy concerns and enabling forms of cyber-enabled misuse such as stalking or targeted harassment.

The results presented in Table 1 show that/*who* search visibility is influenced by account policy rather than architectural constraints. Characters associated with free-trial accounts did not appear in/who search results, even when their names and locations were explicitly specified, indicating that such accounts are intentionally excluded from query visibility. In contrast, characters associated with paid subscription accounts were fully discoverable, with names, locations, and other exposed attributes correctly displayed in the/who search panel. This confirms that the limitation observed for free-trial accounts is policy-based and that the Who Feature remains effective for the majority of active players.

Table 1. Testing done for the/who search feature functions.

Test (#)	Test Conditions	Additional Parameters	Expected Outcome	Actual Outcome
1	Using the Search Who Feature to locate a Free-Trial Account Character	Target character: Testpersonac, Location: Stormwind City	Character name and details will be shown in Who menu	Character did not appear in the search who list
2	Using the Search Who Feature on a location	Location: Exile's Reach	Character names and details will be shown in the Who menu	Characters that are in that specific zone appear on the search

Table 1. *Cont.*

Test (#)	Test Conditions	Additional Parameters	Expected Outcome	Actual Outcome
3	Using the Search Who Feature on a random string	String: HelloThere123456	No character details will show up, Displays 0 results	Result was shown as: [0 Players Found]
4	Confirming the Prior Search Conditions using Last Who commands	Prior Conditions: Test #3	The prior conditions should be stored and displayed as a prompt	The prior search request was displayed as a prompt
5	Using the Search Who on a previous search	Previous Search: Test #3	The same result should be shown as prior test using previous search conditions	The same search was conducted via the command, providing the same result
6	Print Who Search Table to console	N/A	All contents of Who search table are printed from prior tests	Information was printed to the console successfully
7	Clear who search table and repeat print	N/A	No information relating to prior tests is printed	No information was printed, outside of the key for the table
8	Using Search Who Feature on a Non-Free Trial Subscription Character	Character Name: Testpersonaae	Character whereabouts are printed in the who search pannel	Character location and details were printed correctly.

Further testing confirmed that zone-based/*who* searches reliably returned lists of characters present within a specified location, while searches using random or invalid strings consistently produced zero results, as expected. The system also preserved prior search parameters, allowing previous queries to be recalled and re-executed without modification. When combined with timestamped execution, this persistence supports repeated observation under identical conditions across gameplay sessions.

The add-on additionally allowed/*who* search results to be printed to the console for inspection and subsequently cleared, demonstrating predictable and user-controlled handling of collected metadata. Together, these findings show that, although direct technical compromise is mitigated by the platform's sandboxed design, residual privacy risks may arise through the aggregation and longitudinal analysis of user-facing metadata exposed by official APIs.

4.3. Chat Capture and Logging

The second feature recorded messages from the publicly accessible *SAY* and *PARTY* channels. Users could either specify a target character or log all messages in these channels, with each entry stored alongside sender identifiers and timestamps, as illustrated in Figure 6. The captured data could be printed, cleared, or exported using the same mechanisms as the/*who* logs.

Although free-trial accounts could not access *WHISPER* or *RAID* channels, the available channels still provided sufficient access to local conversations and group coordination. The results show that significant volumes of social communication can be harvested within the add-on sandbox, providing a foundation for social-engineering attacks, harassment, and information gathering.

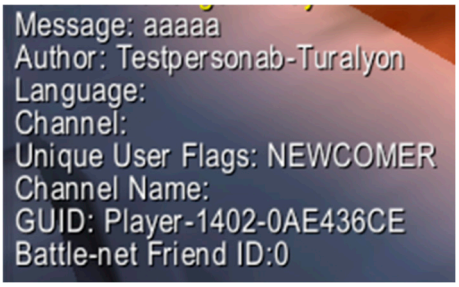


Figure 6. Information example from Message Feature.

4.4. Target Inspection and Metadata Collection

The third feature enabled the collection of metadata from selected nearby players when the tracking toggle was active. Selecting a character caused the TPT to record their visible attributes, including name, race, gender, and globally unique identifier (GUID), as shown in Figure 7. These records were stored persistently and could be printed or exported.

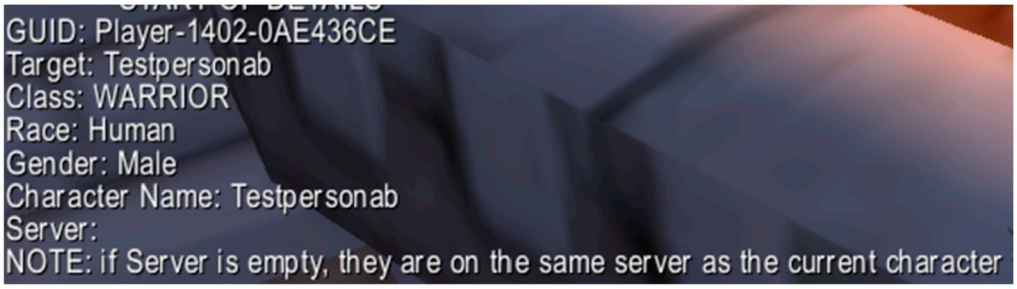


Figure 7. The console response when targeting a character with the command on.

The exposure of GUIDs is particularly significant because it enables persistent identification of characters even if their names change or they move between locations. This capability allows long-term correlation of player activity and strengthens the feasibility of sustained surveillance and profiling.

4.5. Comparative Evaluation

The comparative analysis indicates that, while the platform's sandbox architecture effectively prevents direct technical compromise or privilege escalation, the presence of the third-party tool measurably increases the volume and persistence of accessible player-related metadata when compared with the default environment. In particular, automated logging and structured storage mechanisms enable information to be aggregated over time rather than relying on transient, manual observation. These findings suggest that the primary safeguards operate at the level of system integrity, whereas residual exposure remains at the level of behavioural and identity-related information, which may be leveraged in cyber-enabled misuse scenarios such as profiling or targeted harassment. As shown in Figure 6, the add-on-enabled environment exposes a noticeably higher number of accessible player-related attributes compared with the default gameplay configuration, primarily due to automated logging and structured data aggregation.

Figure 6 presents a comparative overview of the volume of player-related data attributes accessible under the default gameplay environment and when the third-party add-on is enabled. The results illustrate that automated collection mechanisms increase the number of observable metadata, even though core architectural protections continue to prevent direct system-level compromise.

4.6. Note-Taking and Annotation

The fourth feature allowed users to create free-text notes that were displayed on screen and stored with timestamps. Although only the most recent note was visible in the interface, all entries were preserved and could be exported, as shown in Figure 8. This allowed human observations to be integrated with automatically collected data, enabling the construction of detailed behavioural timelines for target players.

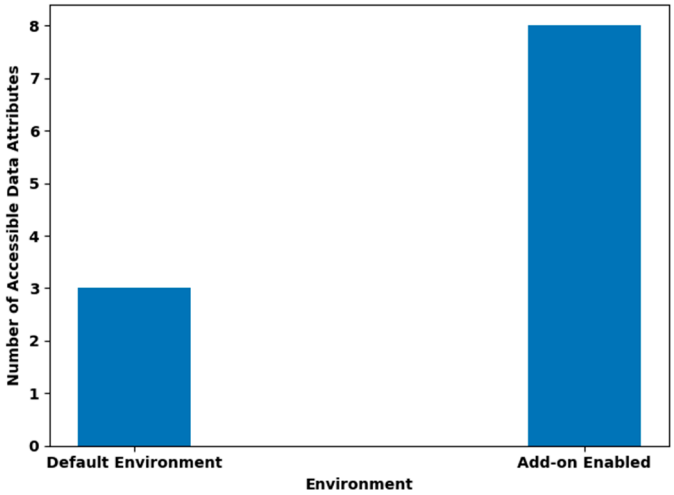


Figure 8. Comparison of data exposure: default environment vs. add-on enabled.

4.7. Data Persistence and Exfiltration

The extraction function represents the final feature implemented in the TPT. It enables users to retrieve information that has been stored across any of the tables generated by earlier functions, allowing this data to be exported from the game for external analysis or other user-defined purposes. Each table entry is separated by a delimiter to ensure that individual records can be reliably distinguished during extraction.

Although the game environment does not natively support direct data export, the stored information can still be accessed through two alternative mechanisms. The first involves manually searching the game's WTF directory to locate both the primary data file and its corresponding backup for each account that has been logged in on the local machine. These files contain variables defined as persistent in the add-on's TOC file. This process is technically feasible but highly time-consuming, as it requires detailed knowledge of the game's file structure and storage conventions in order to locate and interpret the relevant data (Figure 9).

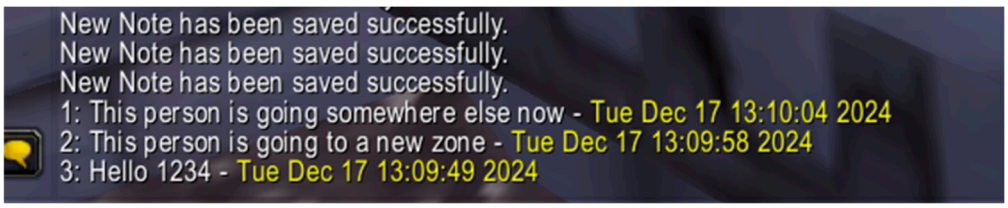


Figure 9. An example of printing the notes made to create a timeline of events.

To simplify exfiltration, the TPT implemented an alternative mechanism that condensed stored data into a single editable text box displayed in-game, shown in Figure 10. This allowed users to copy and paste the information directly into external applications, enabling practical data extraction despite the lack of direct file or network access within the sandbox. The alternative method is to create pop-ups that can take the information and

condense it into a single field, which can be then displayed in an 'editbox' (Figure 11). This is so that users are able to copy and paste the information, as otherwise it would not be collectable in-game.

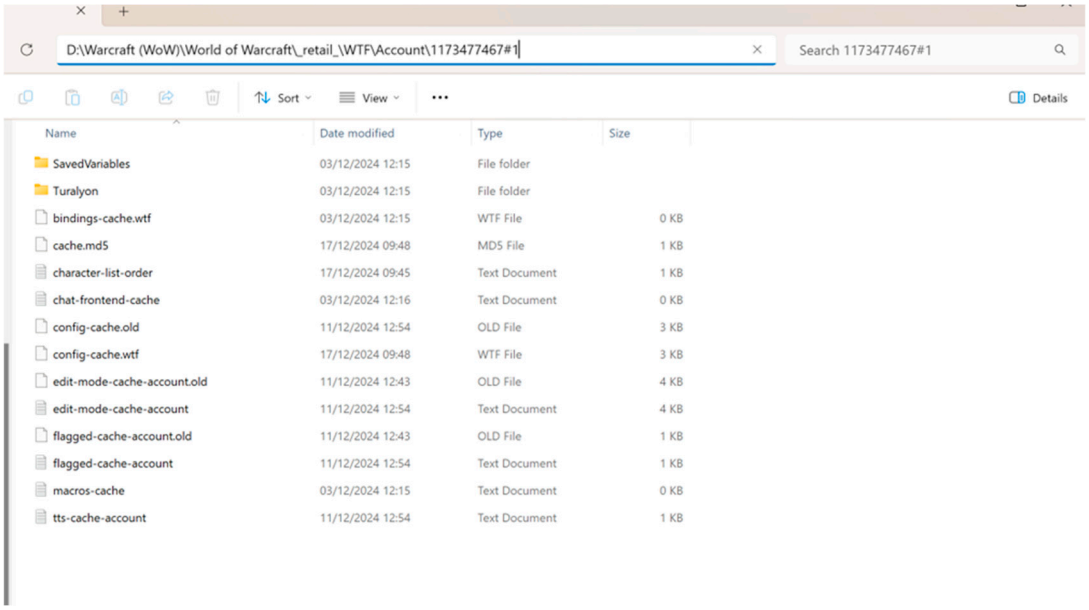


Figure 10. The location where you would find the information using the WTF folder route. Taken using Windows File Explorer.

```
--variables and options for an edit box to extract info for Who Storage
StaticPopupDialogs["EXTRACTINFO2"] = {
    text = "Use CTR+A, then CTR+C to extract the information",
    hasEditBox = true,
    button2 = "Okay",
    OnShow = function (self, data, data2)
        l = 1
        extractMsg = ""
        for k,v in pairs(whoStorage) do
            extractMsg = extractMsg .. "; " .. whoStorage[l]
            l = l + 1
        end
        self.editBox:SetText(extractMsg)
    end,
    timeout = 0,
    whileDead = true,
    hideOnEscape = true,
    preferredIndex = 3,
}
```

Figure 11. Code created for one of the pop-ups used to extract information. Taken using Sublime Text.

4.8. Platform Safeguards and Failure Modes

Testing revealed several API-level protections that constrained malicious functionality. Attempts to send automated chat messages outside permitted contexts failed, as shown in Table 2, indicating that message transmission is a protected function restricted to specific in-game situations. External hyperlinks were also blocked, preventing direct redirection to malicious websites.

Table 2. Additional testing conducted outside the confidences of the created features.

Test (#)	Test Conditions	Additional Parameters	Expected Outcome	Actual Outcome	Notes
1	Attempting to break Who Search feature	Search Criteria: "−1", "aonadondan", "Hz"	Who Search fails	Shows as 0 results, appeared to still work as intended	
2	Attempting to break Messages feature	Target: "−1", "aonadondan", "Hz"	Messages are not stored	Messages not stored as expected, similar to random string test	
3	Attempting to break Notes feature	Notes: "−1", "aonadondan", "Hz"	No note is shown	Note is shown, feature not broken	
4	Send a chat message using custom command	Message "Hello 12345"	Message is sent to SAY Channel	Message was not sent	Appears to be a protected feature, limited in usage to specific instance rules
5	Hyperlink in-game is clicked	No link to external sites, simply says "Hello" to console	Once clicked, an action occurs	Action was performed successfully	
6	a Link to an external site is clicked	N/A	Link should send to an external website	Link failed to work	Seems the API and WoW itself prevent external links

In addition, add-ons do not load until the game world is entered, which prevents any form of pre-login keylogging or credential capture. These safeguards effectively block cyber-dependent crimes such as malware delivery, credential theft, and remote command execution within the add-on framework.

4.9. Cybercrime Feasibility

The experimental results show that while cyber-dependent and most cyber-assisted crimes are prevented by WoW's architecture, cyber-enabled crimes remain feasible. The implemented TPT enabled surveillance, chat harvesting, identity tracking, and behavioural profiling, all of which can violate users' security irrespective of the use of sandboxes, e.g., via profiling, cyberstalking, harassment, fraud preparation, and social-engineering attacks. These findings demonstrate that MMORPG add-on ecosystems constitute a significant socio-technical attack vector, despite the presence of strong technical sandboxing.

4.10. Quantitative Summary of Results

The quantitative observations demonstrate that several data collection and tracking functions operate with a high success rate under normal gameplay conditions, while other capabilities are consistently restricted by API protections or visibility constraints. Expressing these outcomes in numerical terms highlights that the sandbox model significantly reduces the feasibility of cyber-dependent attacks but does not fully eliminate opportunities for persistent information gathering. This distinction is important because it clarifies that the remaining risk is not related to system compromise, but rather to the accumulation and

correlation of user activity data over repeated interactions. The quantitative evaluation summarised in Table 3 presents the number of trials conducted for each capability, the proportion of successful executions, and the practical constraints that limited full functionality within the sandboxed add-on environment.

Table 3. Quantitative Evaluation of Add-on Capabilities.

Capability Evaluated	Number of Trials	Successful Executions	Success Rate (%)	Observed Limitation	Security Implication
Player identification using visible metadata (name, class, level)	10	9	90%	Restricted when player not within visibility range or phased zone	Enables basic profiling of nearby users
Collection of unique identifiers (GUID-based detection where available)	10	7	70%	Some identifiers inaccessible due to API protection	Partial re-identification across sessions remains possible
Automated logging of nearby player activity	10	8	80%	Dependent on event triggers and proximity conditions	Supports aggregation of behavioural data over time
Real-time detection of player presence via system events	10	7	70%	Limited by distance and visibility mechanics	Enables situational awareness that could support monitoring behaviours
Persistence of stored records using local saved variables	5 sessions	5	100%	Storage limited to local client environment	Allows long-term accumulation of player-related metadata
Retrieval of previously stored entries after restart	5 sessions	4	80%	Data loss possible if variables reset or overwritten	Demonstrates feasibility of longitudinal tracking patterns
Manual observation without add-on (baseline comparison)	10	3	30%	Requires continuous human attention and memory	Shows automation significantly increases data capture capability

The results indicate that, although architectural safeguards prevent direct system compromise, automated data collection and persistence mechanisms substantially increase the reliability and volume of accessible player-related information when compared with manual observation alone. To ensure the reliability of the experimental observations, each test scenario was executed multiple times under controlled and consistent conditions. The purpose of these repetitions was to verify that the observed behaviours, including data exposure, event generation, and aggregation mechanisms, were reproducible and not dependent on isolated or incidental interactions within the system. Across all experiments, repeated executions produced consistent outcomes, indicating that the identified data collection and behavioural inference mechanisms operate deterministically within the constraints of the platform's API and event-driven architecture. The success rates reported in Table 3 therefore reflect stable and repeatable behaviour observed across multiple runs, rather than single-instance results. Given the proof-of-concept nature of this study, the

experimental design prioritises verification of feasibility and reproducibility over large-scale statistical evaluation. As such, formal analysis of variance or distributional characteristics of the results is not included. A more extensive statistical treatment, including systematic measurement of variability across a larger number of repetitions and diverse scenarios, is identified as part of future work.

5. Discussion

The results of this study provide clear empirical evidence that third-party tools (TPTs) embedded within MMORPG platforms can support meaningful forms of cybercrime, although the feasibility varies substantially across crime categories. By implementing and testing a fully functional add-on within World of Warcraft, this work moves beyond speculative or anecdotal claims and demonstrates, in a controlled and repeatable manner, which attack classes are enabled by the platform's architecture and which are constrained by its security model.

The findings of this study are derived from controlled experiments conducted within the World of Warcraft add-on environment and therefore reflect the capabilities and constraints of this specific platform. The results show strong support for the feasibility of cyber-enabled activity within this context. The implemented TPT was able to perform player tracking, chat logging, identity correlation via GUIDs, and persistent behavioural recording, all of which represent mechanisms that can support stalking, harassment, profiling, and social-engineering scenarios. These observations align with prior research on cyber-enabled activity in digital communities, which highlights how persistent identity, communication, and visibility can amplify harmful behaviours [5,6]. In MMORPG environments, these effects are further shaped by the continuity of player identity and the immersive nature of interaction.

In contrast, cyber-dependent activity was consistently prevented by platform-level controls. Attempts to implement pre-login keylogging, credential harvesting, malware delivery, or external command execution were blocked by the protected execution model, delayed add-on loading, and restrictions on external communication. These results are consistent with prior work describing MMORPG add-on frameworks as strongly sandboxed environments designed to prevent direct technical compromise [9,10]. Similarly, cyber-assisted activity was constrained by limitations on automated messaging and the inability to integrate external communication channels within the client.

Taken together, these results indicate an asymmetry between technical enforcement and information exposure within the evaluated environment. The platform architecture effectively prevents direct system compromise but allows structured access to user-visible metadata through legitimate mechanisms. This behaviour is empirically demonstrated within the World of Warcraft add-on framework and should be interpreted within that specific context.

At a broader level, these findings suggest that similar patterns may arise in other extensible systems where third-party components operate within sandboxed environments while retaining access to user-facing data. However, this should be understood as a conceptual implication rather than a generalised claim. Differences in API design, permission models, and platform governance may significantly affect the extent and nature of such behaviour in other systems. The contribution of this study therefore lies in providing a detailed case-specific analysis that highlights how socio-technical risks can emerge in a policy-compliant environment, rather than establishing universal behaviour across all extensible platforms.

This study focuses on the design and evaluation of a prototype add-on developed to systematically analyse data exposure within the constraints of officially supported APIs.

While the implementation is presented as a standalone system, it is important to situate it within the broader ecosystem of existing add-ons and data collection approaches. Many commonly used add-ons in MMORPG environments rely on similar API calls and event-driven data structures to provide functionality such as performance tracking, interface customisation, and gameplay assistance. These tools demonstrate that access to structured user-visible data is an established and legitimate feature of the platform. However, such add-ons are typically designed with functional or usability objectives in mind, rather than with the explicit goal of analysing privacy implications or enabling longitudinal behavioural inference. In contrast, the prototype presented in this study adopts a different perspective by structuring and aggregating accessible data specifically to evaluate its potential for inference. The contribution therefore lies in how the data is collected, organised, and interpreted, rather than in the introduction of new data access capabilities.

Alternative data collection approaches, such as external logging tools, network traffic analysis, or client modification techniques, may provide access to a broader range of information. However, these methods often require elevated privileges, may violate platform policies, or fall outside the boundaries of legitimate system use. The approach taken in this study is intentionally constrained to officially supported interfaces, ensuring that the analysis reflects realistic and policy-compliant conditions. By positioning the prototype within this broader context, the study highlights that the identified privacy risks are not dependent on novel or intrusive techniques but can arise from mechanisms that are already widely available and accepted within the platform ecosystem.

The findings of this study provide insight into the privacy implications of add-on systems; however, several limitations should be acknowledged to properly contextualise the results. The experimental evaluation was conducted within a single platform, namely World of Warcraft. This platform was selected due to its mature and well-documented add-on ecosystem, clearly defined API structure, and strict sandboxing model, all of which enable controlled and reproducible analysis. At the same time, it is recognised that differences in API design, data exposure policies, and architectural constraints across other MMORPGs or similar interactive systems may influence both the type and extent of observable data. The specific observations presented in this study should therefore not be interpreted as directly transferable to all platforms.

The contribution of this work lies in identifying a broader conceptual mechanism rather than presenting a platform-specific outcome. The study demonstrates that privacy-relevant inferences can emerge through the longitudinal aggregation of data that is individually accessible and policy-compliant. This principle is not inherently tied to a single system but is instead characteristic of environments that combine structured API access, event-driven data generation, and persistent user interaction. In this respect, the empirical evaluation is platform-bound, while the underlying insight has wider relevance.

To more clearly connect the observed data collection capabilities with potential misuse scenarios, the results can be interpreted through a structured threat mapping. The prototype demonstrates the ability to collect and persist several categories of user-visible metadata, including player identifiers (e.g., GUID-linked information), chat interactions, proximity-based discovery via/who, and user-generated annotations. Individually, these data elements are accessible through normal gameplay; however, their aggregation enables additional forms of inference. From an adversarial perspective, these capabilities require only the use of a standard add-on operating within permitted functionality, without the need for elevated privileges or technical exploitation. This corresponds to a low-complexity attacker model, such as a malicious player or small coordinated group using standard game interfaces. The collected data can support several plausible misuse pathways. For example, persistent GUID-linked records combined with chat capture may enable tracking

of player presence and behaviour over time, supporting targeted harassment or unwanted monitoring. Similarly, the aggregation of communication patterns and contextual annotations may facilitate profiling or social-engineering preparation by allowing an attacker to build a structured view of player behaviour, relationships, and activity patterns.

It is important to emphasise that the study does not demonstrate the occurrence of these harms directly. Rather, it identifies the enabling conditions under which such behaviours may arise, based on the availability, persistence, and correlation of data within the add-on environment. The findings therefore highlight a potential gap between technical enforcement mechanisms and the broader socio-technical implications of data exposure.

The experimental design is based on a limited number of controlled accounts. The use of two accounts was a deliberate methodological choice, allowing the study to operate within a controlled setting where interactions, event generation, and data collection processes could be systematically observed and verified. This approach supports internal validity, reproducibility, and clear attribution of observed behaviours. However, it also limits the scale of the study. The small sample size restricts the possibility of statistical validation and limits the ability to generalise the findings across diverse user populations. The results should therefore be interpreted as a demonstration of feasibility rather than statistically generalisable evidence. The study shows that even under constrained and minimal conditions, it is possible to collect and aggregate data in a way that enables behavioural inference, indicating that the identified risk can arise without reliance on large-scale datasets.

In addition to these constraints, the experimental evaluation was conducted under controlled conditions without the involvement of real players. This design enables precise observation of system behaviour and ensures that data collection and aggregation processes can be analysed in a consistent and reproducible manner. However, it does not fully capture the complexity of real gameplay environments, where player behaviour is shaped by dynamic social interactions such as group activities, communication, coordination, and emergent gameplay patterns. These factors may influence both the volume and structure of observable data and may affect the nature and strength of behavioural inference.

The experimental evaluation presented in this study is based on a limited set of functional tests designed to demonstrate the feasibility of data collection and behavioural inference within the constraints of officially supported APIs. While these experiments provide clear evidence of the mechanism under investigation, they do not constitute a large-scale or statistically driven evaluation. As a result, the study does not provide a quantitative assessment of the prevalence, variability, or statistical significance of the observed patterns. The findings should therefore be interpreted as a proof-of-concept demonstration rather than as a comprehensive empirical analysis. However, the controlled and reproducible nature of the experimental setup ensures that the observed behaviour can be consistently replicated under the same conditions, supporting the reliability of the methodological approach.

While the current study demonstrates the feasibility of data collection and behavioural inference, it does not include a quantitative evaluation based on formally defined metrics or statistical testing. The experimental design focuses on validating the existence and operation of the underlying mechanism rather than measuring its performance or comparative effectiveness.

The evaluation was also conducted on a single server and under a narrow set of gameplay conditions. While this was sufficient to establish the feasibility of the observed behaviours within a real and policy-compliant add-on environment, it does not demonstrate how these behaviours may vary across different servers, character types, gameplay settings, or broader account configurations. As such, the present study should be inter-

preted as establishing technical feasibility under controlled conditions rather than external validity across the full diversity of MMORPG use. Future work should therefore extend validation across multiple servers, character states, and gameplay contexts to determine the consistency and variability of the identified mechanisms.

The current evaluation does not include a formal statistical analysis of variance across repeated trials, as the study is primarily focused on demonstrating feasibility under controlled conditions.

6. Conclusions

This study examined whether third-party tools in MMORPGs could support cyber-enabled misuse rather than direct technical compromise. Using a functional add-on developed for World of Warcraft, the results show that while the platform's sandboxed architecture effectively prevents cyber-dependent attacks such as credential theft and malware delivery, it nevertheless permits the collection and aggregation of user-facing metadata. This enables cyber-enabled activities, including stalking, harassment, profiling, and social-engineering preparation. These findings demonstrate that third-party tools pose a meaningful socio-technical risk not by breaking platform security, but by observing and exploiting human behaviour within officially exposed interfaces, highlighting the need for continued research into privacy-aware design for extensible digital platforms.

Future research should examine other MMORPGs and modification ecosystems to assess how API design, governance, and enforcement influence socio-technical risk. In particular, platforms that prohibit third-party tools and rely on unofficial extensions warrant closer scrutiny, as they may lack comparable safeguards. Further work on privacy-preserving API design, behavioural rate-limiting, and add-on monitoring could help balance extensibility with misuse prevention. More broadly, these findings underscore the need to treat virtual worlds as next-generation computing platforms where usability, extensibility, and security must be co-designed.

Future work will extend this approach by:

1. Covering multiple platforms and larger user populations, as well as incorporating more naturalistic gameplay scenarios involving real players. This includes evaluating different MMORPG environments and extension frameworks to assess the consistency of data exposure patterns and conducting longer-term studies with a broader set of users to enable quantitative analysis. Such extensions will support a more comprehensive understanding of the prevalence, variability, and impact of the identified phenomena.
2. Incorporating clearly defined quantitative metrics, such as the number of detected players, the volume of captured interaction events, and the consistency of player identification across sessions. These metrics will enable a more structured assessment of the data collection and inference capabilities of the proposed approach. In addition, subsequent studies will include comparative evaluation under controlled baseline conditions, such as manual observation or non-instrumented gameplay, allowing direct comparison with the add-on-based method. This will support the application of statistical analysis techniques to evaluate differences between conditions. Depending on the distribution and characteristics of the collected data, appropriate statistical tests, including parametric and non-parametric methods, will be employed to assess the significance of observed differences.

   Extending the study in this manner will enable a more comprehensive and statistically grounded evaluation of the proposed approach, supporting a clearer understanding of its effectiveness and practical implications.

Future work will also extend the evaluation by incorporating larger datasets, a greater number of users, and longer observation periods. This will enable the application of

statistical analysis techniques to assess the scale and variability of the phenomenon, as well as to evaluate reproducibility across different environments and usage scenarios. Such extensions will provide a more comprehensive understanding of the broader implications of the identified data exposure mechanisms.

Author Contributions: Conceptualization, G.S. and P.R.; methodology, G.S. and P.R.; software, G.S.; validation, G.S., P.R., and S.P.; formal analysis, G.S. and X.W.; investigation, P.R.; resources, G.S.; data curation, G.S.; writing—original draft preparation, P.R.; writing—review and editing, A.K.S., S.P., X.W., and A.M.; visualization, G.S. and P.R.; supervision, P.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding. Any support provided in the form of institutional resources or supervisory guidance did not influence the study design; the collection, analysis, or interpretation of data; the writing of the manuscript; or the decision to submit the work for publication.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: This study did not involve human participants. The investigation was conducted using researcher-managed accounts within the World of Warcraft add-on environment under controlled and repeatable conditions. No participant data, personal data, or identifiable human subject information was collected or processed. Consequently, informed consent was not required.

Data Availability Statement: https://github.com/Parisa823/MMORPG_Cybercrime.git (accessed on 14 April 2026).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
DLL	Dynamic-Link Library
EULA	End User License Agreement
MMORPG	Massively Multiplayer Online Role-Playing Game
PvE	Player versus Environment
PvP	Player versus Player
RMT	Real-Money Trading
TPT	Third-Party Tool
UI	User Interface
WoW	World of Warcraft

References

1. Jøn, A. The development of MMORPG culture and the guild. *Acad. Forum* **2010**, *25*, 97–112.
2. Zhang, Q.; Zhu, Y.; Onita, C.G.; Banks, M.S.; Zhang, X. Towards a conceptual model for understanding the relationships among MMORPGs, gamers, and add-ons. *Issues Inf. Syst.* **2020**, *21*, 186–195.
3. Phillips, K.; Davidson, J.C.; Farr, R.R.; Burkhardt, C.; Caneppele, S.; Aiken, M.P. Conceptualizing cybercrime: Definitions, typologies and taxonomies. *Forensic Sci.* **2022**, *2*, 379–398. [CrossRef]
4. Akdemir, N.; Lawless, C. Exploring the human factor in cyber-enabled and cyber-dependent crime victimisation: A lifestyle routine activities approach. *Internet Res.* **2020**, *30*, 1665–1687. [CrossRef]
5. Chalak, A. Cyber-Crime in Online Videogames and Its Reporting by School-Aged Children. Master's Thesis, Macquarie University, Sydney, Australia, 2020.
6. Ibrahim, A. Guarding the future of gaming: The imperative of cybersecurity. In *2024 2nd International Conference on Cyber Resilience (ICCR)*; IEEE: New York, NY, USA, 2024; pp. 1–9.
7. Endeley, R. End-to-end encryption in messaging services and national security: Case of WhatsApp Messenger. *J. Inf. Secur.* **2018**, *9*, 123–138. [CrossRef]

8. Matteo, C. Cybercrimes and virtual worlds: A systematic literature review. *J. Inf. Secur. Cybercrimes Res.* **2022**, *5*, 124–134. [CrossRef]
9. Borges, R.C.; Malheiros, M.D.G.; Billa, C.Z.; Pias, M.R.; Bicho, A.D.L. An open-source framework using WebRTC for online multiplayer gaming. In *Proceedings of the 22nd Brazilian Symposium on Games and Digital Entertainment*; Association for Computing Machinery: New York, NY, USA, 2023; pp. 143–150.
10. Quick, L. Playing to profit: Selling currency in video games. *Deviant Behav.* **2024**, *46*, 1697–1714. [CrossRef]
11. Lin, Y.; Liao, Y.; Zeng, W.; Wei, Y.; Chen, D.; Yuan, X.; Li, Y.; Erkan, U.; Toktas, A.; Zhang, C.; et al. 3D Non-degenerate Hyperchaos: Design, Analysis, and Application in Image Encryption. *IEEE Trans. Consum. Electron.* **2026**, 1. [CrossRef]
12. Mulakaluri, T.; Puppala, R.; Yasmina, C.; Madhavi, P.B.; Balaji, K.S.V.G. Reviewing cheating and detection techniques in massively multiplayer online role-playing games: A systematic analysis. In *2024 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)*; IEEE: New York, NY, USA, 2024.
13. Viescinski, A.; Heinrich, T.; Fulber-Garcia, V.; Maziero, C. How risky is it? A closer look at game anti-cheat software. In *2025 IEEE Symposium on Computers and Communications (ISCC)*; IEEE: New York, NY, USA, 2025; pp. 1–6.
14. Paul, C. Optimising play: How theorycraft changes gameplay and design. *Game Stud.* **2011**, *11*, 100.
15. Hannes, P. Popularity of User Interface Modifications in World of Warcraft. Bachelor's Thesis, Haaga-Helia University of Applied Sciences, Helsinki, Finland, 2022.
16. Hampton, D. The Technical World of Warcraft. Master's Thesis, University of Central Florida, Orlando, FL, USA, 2021.
17. Grande, V. ColLabWoW: Development of a Decision Support Add-on for World of Warcraft. Bachelor Thesis, University of Skövde, Skövde, Sweden, 2013.
18. Starink, J. Analysis and Automated Detection of Host-Based Code Injection Techniques in Malware. *J. Comput. Virol. Hacking Tech.* **2021**, *17*, 1–12.
19. Gavrilov, V.I.; Oskin, A.R. *Lua Multiparadigm Programming Language: Application and Opportunities*; Polotsk State University: Novopolotsk, Belarus, 2019; pp. 164–165.
20. Fabian, M.; Malik, R.; Mohajerani, S. Lupremica: Lua scripting for Supremica. *IFAC-PapersOnLine* **2023**, *56*, 6099–6104. [CrossRef]
21. Bogdanov, M. Conversing in Massive Multiplayer Online Games: A Discourse Analysis of Chat Interactions in World of Warcraft and League of Legends. Doctoral Dissertation, Vytautas Magnus University, Kaunas, Lithuania, 2022.
22. Saleem, S.; Khan, N.F.; Zafar, S.; Raza, N. Cyberbullying and cyber harassment: A tertiary study. *Comput. Hum. Behav.* **2022**, *128*, 107118.
23. Wilson, C.; Sheridan, L.; Garratt-Reed, D. What is cyberstalking? A review of measurements. *J. Interpers. Violence* **2022**, *37*, NP9763–NP9783. [CrossRef] [PubMed]
24. Grandhi, S.; Galimotu, N. Understanding social engineering threats in massively multiplayer online role-playing games. *Glob. Adv. Res. J. Bus. Manag. Stud.* **2020**, *9*, 66–71.
25. Crown Prosecution Service. *Fraud Act 2006*; CPS: London, UK, 2020.