Preprints.org

Article

Not peer-reviewed version

Leveraging Reinforcement Learning for an Efficient Automation of Windows Registry Analysis During Cyber Incident Response

Mohamed Chahine Ghanem * , Elhadj Benkhelifa , Dominik Wojtczak , Mohamed Amine Ferrag , Norbert Tihanyi , Erivelton Geraldo Nepomuceno

Posted Date: 7 January 2025

doi: 10.20944/preprints202501.0416.v1

Keywords: digital forensics; Windows forensics; Window registry; timeline analysis; registry forensics; cyber incident; automation; Rule-Based Expert Systems; Reinforcement Learning; MDP; incident response

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Leveraging Reinforcement Learning for an Efficient Automation of Windows Registry Analysis During Cyber Incident Response

Mohamed Chahine Ghanem [1,2,*], Elhadj Benkhelifa [3], Dominik Wojtczak [1], Mohamed Amine Ferrag [4], Norbert Tihanyi [5] and Erivelton Geraldo Nepomuceno [6]

[1] Cybersecurity Institute, University of Liverpool, Liverpool, UK
[2] Cyber Security Research Centre, London Metropolitan University, London, UK
[3] Smart Systems, AI and Cybersecurity Research Centre, Staffordshire University, Stafford, UK
[4] Department of Computer Science, Guelma University, Algeria
[5] Technology Innovation Institute (TII), Abu Dhabi, UAE
[6] Hamilton Institute, Maynooth University, Kildare, Ireland
* Correspondence: mcghanem@liverpool.ac.uk

Abstract: Windows represents the most common platform found in seized computers due to its widespread presence. This disparity has become worse due to the introduction of Microsoft's Windows. Post Cyber Incident analysis of Microsoft Windows machines has become increasingly challenging due to the ever-evolving nature of digital threats. Traditional digital forensics methods often struggle to keep pace with modern cybercrime activities' volume, sophistication, and complexity, which either target or originate from Windows machines. In response to these challenges, this research introduces WinRegRL, a framework that combines Reinforcement Learning (RL) and Rule-Based Artificial Intelligence (RB-AI) to enhance the efficiency, effectiveness and accuracy of digital investigations in the context of Windows Operating Systems. WinRegRL fully captures key information, elaborates the MDP environment, solves the RL problem and extracts expertise for later use. Implementation and testing of WinRegRL validated the research hypothesis by enabling optimised analysis and correlation of Registry forensics. Results prove that the proposed RL model outperforms all previous approaches including bling automation and human expert performance in terms of time, the number of artefacts explored, and the accuracy of results. Another advantage of the proposed framework is the ease of repetition, especially in this context, where more than one machine of the same configuration is under investigation, a context often faced in real DFIR practice.

Keywords: digital forensics; Windows forensics; Window registry; timeline analysis; registry forensics; cyber incident; automation; Rule-Based Expert Systems; Reinforcement Learning; MDP; incident response

1. Introduction

Microsoft Windows dominates the global operating system (OS) market, with over 75% of desktop users worldwide relying on it for both personal and professional purposes [1]. This widespread adoption, however, makes Windows a prime target for malicious actors seeking to exploit vulnerabilities for data theft, system disruption, and other nefarious activities [2]. The increasing complexity, volume, and evolution of cyber threats have rendered traditional investigative methods insufficient for addressing large-scale cyber incidents [3,4].

The Windows Registry, a hierarchical database storing critical system configurations and user activity data, is pivotal in digital forensic investigations. It contains interrelated keys and values that record recently accessed files, user actions, and system settings [4]. However, the sheer volume of data within the Registry and the sophistication of modern cyberattacks pose significant challenges for the digital forensics and incident response (DFIR) community [6]. The introduction of Windows 10 and 11,

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

designed for seamless operation across computers, smartphones, tablets, and embedded systems, has further expanded their adaptability and reach, complicating forensic efforts [5].

In cyber or cyber-enabled offenses, the efficiency of DFIR processes is critical for loss recovery and mitigation [3]. Effective DFIR can identify key elements such as the point of entry, techniques used for unauthorised access, perpetrator identities, and attack motives [4]. Digital forensics, as an investigative technique, is essential for identifying and analysing evidence related to cybersecurity incidents or crimes [5]. The Windows Registry serves as a living repository of configuration settings, user activities, and system events, providing a foundation for reconstructing digital event sequences, detecting malicious behaviour, and delivering crucial evidence in investigations [9]. The cyber incident response involves both live monitoring and post-incident forensic analysis. Live monitoring tracks and records system data in real-time, while post-incident analysis investigates events after they occur. Both methods are used to map Registry paths containing USB identifiers, such as make, model, and GUIDs (Global Unique Identifiers), which distinguish individual USB devices. These identifiers are often located in both allocated and unallocated Registry spaces [7].

This research aims to enhance the efficiency and effectiveness of forensic investigations involving the Windows Registry. Modern Windows versions generate substantial logging data, which can hinder analytical processes [8]. To address this challenge, we propose a novel Reinforcement Learning (RL)-driven framework that combines RL techniques with a Rule-Based Artificial Intelligence (RB-AI) system. The framework incorporates a Markov Decision Process (MDP) model to represent the Windows Registry and Events Timeline environment, enabling state transitions, actions, and rewards for Registry analysis and event correlation.

1.1. Challenges and Motivation

Current Registry forensic analysis, whether manual or automated, faces several challenges:

- Volume of Data: The Registry contains vast amounts of data reflecting system activities and user interactions.
- Lack of Automation: Many traditional tools require manual, repetitive tasks, consuming significant time and resources.
- Dynamic Nature: Registry entries frequently change during system operation or software installations.
- Limited Contextual Information: Registry entries often lack explicit context, necessitating expert interpretation.
- Data Fragmentation: Registry data may be fragmented across hives and keys.
- Evolution of Windows Versions: Different Windows versions introduce variations in Registry structures.
- Limited Advanced Analysis: Traditional tools often lack features like anomaly detection or pattern recognition.

Addressing these challenges requires advanced tools, investigative techniques, and AI integration to enhance the efficiency and accuracy of Registry analysis. Our motivation for integrating reinforcement learning (RL) is to optimise the automation and enhance the coverage of Windows Registry analysis in cyber incident response. This is driven by the growing complexity and volume of cyber threats targeting critical systems. The Windows registry stores various information, including system information, configuration settings, and application data. Considering the data found in the registry, it is an invaluable source of evidence in digital investigations. Windows registry is key for tracking changes and uncovering traces of malicious acts in the registry. However, the amount of data contained in the registry poses eminent challenges to digital forensic analysis. this is particularly evident when real-time analysis is necessary. Analysing the entire registry manually can be time-consuming and resource-intensive, making it challenging for forensic analysts to efficiently extract relevant information.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

1.2. Research Questions

This study aims to contribute to the advancement of Cyber incident response in the context of Windows Registry analysis and forensics by addressing the following research questions:

RQ1: How does the combination of RL with RB-AI, under the framework of WinRegRL, help improve the effectiveness, accuracy, and scalability of digital forensic investigations on Windows Operating Systems?

RQ2: What are the primary metrics of performance used to validate the effectiveness of the framework WinRegRL compared to traditional digital forensics methods, and how does it address the problems brought about by modern cybercrime in attacking Windows Registry artefacts?

RQ3: To what extent does the WinRegRL framework allow for consistent, optimised analysis on many machines sharing identical configurations, and what implications are there for practical applications in Digital Forensics and Incident Response (DFIR)?

1.3. Novelty and Contribution

The research novelty lies in the leveraging of RL to enhance the automation of cyber incident investigation in Windows machines in terms of accuracy, timing, and coverage. In the beginning, this work identified the gap by reviewing existing methodologies, tools, and techniques for analysing registry artefacts and identifying current challenges and limitations. Subsequently, this work's contributions to the body of knowledge and DFIR practice are:

MDP Model and WinRegRL: A comprehensive framework relying on a novel MDP model that provides a comprehensive and structured representation of Windows environment covering Registry, Volatile Memory and File System data and facilitating an efficient automation of the investigation process.

Expertise Extraction and Reply: Through the Optimal Policies capturing and generalisation, We minimise investigation time by enhancing the MDP solving time through direct feeding of optimal policies to the out-of-the-box MDP solver, which reduces the time needed for the RL agent to discover optimal decision policies for each state.

Performance Superiority: Observing the empirical results, our results demonstrate that the proposed RL-based approach outperforms existing industry approaches in terms of Consumed Time, Relevance and Coverage accounting for both automated systems and human experts.

1.4. Article Organization

The remainder of this paper is organized as follows: Section 2 provides a technical overview of Windows registry and volatile memory forensic analysis as well as the current state of the art of tools, and framework investigation techniques. Section 3 explains registry structure and its relevance in forensics. Additionally, the section introduces the reinforcement learning key elements as well as RL approaches and how they were leveraged in this research. The section also covers the proposed RL-powered framework's overall architecture and modules. Section 4 covers WinRegRL framework implementation, testing, obtained results, and an in-depth discussion. Section 5 provides reflections and conclusions regarding answering the research questions and future directions.

2. Literature Review

2.1. Windows Registry

The Windows Registry is a cornerstone of Microsoft Windows OS, serving as a hierarchical database that stores configuration settings, systems information, and user preferences. From a forensic perspective, the Registry is a goldmine of information, offering insights into system activities, user actions, and potential security breaches. Table 1 summarizes the high-level architecture and content of different Registry hives. These hives store critical data, including user profiles, installed software, and network settings, and are traditionally analysed using tools such as Registry Editor or specialized forensic software like FTK Imager. Key forensic artefacts within the Registry include **UserAssist**

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

(recently executed programs), *MRU* (most recently used files), *ShellBags* (folder navigation), and *External Devices* (connected USB drives) [16]. These artefacts provide valuable information for live security monitoring and post-incident investigations.

Table 1. Windows Registry Hives and Supporting files [22].

Registry hive	Supporting files
HKEY_LOCAL_MACHINE_SAM	SAM, SAM.log, SAM.sav
HKEY_LOCAL_MACHINE_Security	Security, Security.log, Security.sav
HKEY_LOCAL_MACHINE_Software	Software, Software.log, Software.sav
HKEY_LOCAL_MACHINE_System	System, System.alt, System.log, System.sav
HKEY_CURRENT_CONFIG	System, System.alt, System.log, System.sav, Ntuser.dat, Ntuser.dat.log
HKEY_USERS_DEFAULT	Default, Default.log, Default.sav

The registry contains "hives," each containing supporting files, as illustrated in the table below. These hives store critical information, including user profiles, installed software, network settings, and more. Traditionally, such data has been extracted and analysed using native tools such as Registry Editor or Specialized forensics tools like FTK Imager. During analysis artefacts including **UserAssist** which reveals recently executed programs, \textit{MRU (most recently used) }which indicates which file is recently accessed, \textit{ShellBags} which helps with folder navigation, and \textit{External devices}, which gives information about USB drives that have been connected to the system are examined [16].

The registry is loaded into memory when the system is booted and written to disk when the system is shut down. Due to its nature of storing low-level configuration settings, the registry contains hugely valuable artefacts for forensic analysis [15]. File system artefacts are caused by the Windows filesystem, which is used by NT (new technology) to store and retrieve HDDs (hard drive disks) and SSDs (solid state drives). Event Log is contained in the Windows Event Log and it stores audit logs from the OS and applications. Memory artefacts are artefacts that can be collected from endpoints while the system is running from the live system. These four formulate complete Windows endpoint forensic analysis. The Windows registry is a hierarchical database that stores information for configuring the system for applications, users, and hardware devices. The main registry files are software, system, security, and SAM registry [9].

2.2. Window Cyber Incident Response and Analysis

Windows registry forensics—part of both live monitoring and post-incident analysis methodologies—deals with the real-time tracking of system activities, where data is captured either when a USB device is inserted or removed from the system, and after events have occurred [4]. The registry paths housing information about USB-related data, inclusive of make, model, and Globally Unique Identifiers (GUIDs) that identify individual USB devices, are identified by means of both methodologies [5] and [6]. This information is invaluable in reconstructing USB usage and for analyzing potential security implications [14]. Traditional approaches to analyzing the registry usually involve manually searching through registry keys, subkeys, timestamps, and values. Such methods are feasible in controlled or small-scale environments but become ineffective and prone to error in high-usage or fluid environments [29]. Especially in the case of large-scale attacks, manual registry analysis in incident response can be particularly laborious, where subtle IOCs or complex attack sequences must be identified. The sheer volume and complexity of the registry, which records changes generated by software installations, user activities, and system operations, only exacerbate the challenges of manual analysis [30].

Automated tools summarised in Table 3 have emerged to address these inefficiencies, but many depend on static, rule-based approaches. These systems, hence, have inherent limitations in their adaptability, often failing to keep up with emerging patterns of attack or new vectors of threat that may be hiding within registry alterations. This, therefore, constrains the generation of actionable insight

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

in dynamic and complex threat environments through dependence on static approaches. With the increasing sophistication of today's systems and the changing dynamics of cyber threats, there has never been a greater need for more adaptive and intelligent forensic systems.

Newly developed frameworks seek to overcome these shortcomings by implementing automation strategies that identify patterns within registry data. These systems are capable of conducting real-time analyses of registry modifications, thereby minimizing dependence on human oversight and enhancing the identification of unauthorized access, software installations, and alterations to system configurations. This transition towards intelligent and adaptive methodologies is essential for tackling the complexities associated with contemporary registry forensics, as well as facilitating scalable and resilient investigative processes.

2.3. Windows Registry Hives and Volatile Data Acquisition

Registry hives play a vital role in maintaining information about user profiles, installed applications, system configurations, network settings, and connected peripherals. During volatile data acquisition, tools like RegRipper extract valuable forensic artefacts from registry hive files, facilitating the analysis of system activity, user behaviour, and potential signs of compromise. This makes registry hives a crucial source of evidence in digital forensics and incident response, where their detailed insights aid in reconstructing events and identifying security breaches. Table 2 summarise Windows registry forensic analysis scope and coverage.

Table 2. WinRegRL forensic analysis Coverage and Scope

Windows Artefacts	Scope & Description
Operating Systems fingerprinting	Windows 7, Windows 8/8.1, Windows 10, Windows 11, and Server 2008/2012/2016/2019/2022
File Systems	NTFS, FAT, exFAT
Registry Forensics	Shell Items, Shortcut Files (LNK)-File Opening, ShellBags-Folder Opening
Jump Lists	File Opening and Program Execution
Users' Activities	Browser, Webmail, MS Office Documents, System Resource Usage DB Search Index, Recycle Bin, Files Metadata, Myriad Execution, Electron/WebView2
Cloud Files and Metadata	OneDrive, Dropbox, Google Drive, and Box
Timeline and Journaling	Microsoft Unified Audit Logging, Event Log Analysis, Deleted Registry Key, File Recovery ESE Database, .log Files, Data Recovery, String Searching, File Carving
Peripheral Profiling and Analysis	Removable Device, Suspect Executed Program, Remote Logging (Console, RDP, or Network)
Anti-Forensics	Auto-Deleted Browse (Private Browsing), File Wiping, Time Manipulation, and Application Removal

2.4. Manual Events Timeline Analysis

We have defined here event reconstruction in digital forensics using finite-state machines [27]. However, in simpler terms, this refers to the process of transforming the state of digital objects into the events that caused those states. This can range from simply confirming that an event occurred to highlighting the exact time it occurred. Detailed event reconstruction involves examining timestamps gathered from digital forensic artefacts. These timestamps can be found in various sources including file system metadata (e.g., file access modification, creation times) and more complex file formats Windows Registry, event logs, and SQLite databases.

Plaso(log2timeline): represents the cutting edge in timestamp extraction, featuring parsers and plugins. However, a major challenge lies in the sheer volume of timestamps generated during system analysis. This occurs even with minimal user activity, which could result in millions of data points. To tackle this challenge,e efforts have been made to automate the analyses of these timestamps to construct meaningful activity history. A method proposed by Hargreaves and Patterson involves manually the pattern of low-level timestamps that correspond to higher-level events. For example, a user opening a file on Windows OS generates a series of low-level artefacts, including changes in the registry, jump lists, and link files. This pattern-matching approach can be manually coded, however, it is often labor-intensive to identify, code and test these changes. In addition, there is a risk of errors due to subtle variations in the behaviour of different OSs variations. These errors could result in incorrect conclusions [34].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

Registry Data capturing These steps ensured a systematic and comprehensive approach to analysing changes in the Windows registry caused by the introduction of USB drives. Capturing Windows registry data in forensic investigations involves extracting key registry hives, such as NTUser.dat, SYSTEM, SOFTWARE, and SAM, which store critical system and user information. For this task, we utilised RegRipper and Regshot to parse and compare registry snapshots efficiently. The process begins with creating a forensic image of the target system to ensure data integrity and prevent accidental modifications. Once the registry hives are extracted, they are analysed for evidence of user activity, software installations, and external device connections. This information is fed into the timeline of events, user actions, and potential malicious activity.

An initial snapshot of the Registry hives from the target system was taken using Regshot before introducing any test USB drives. A specific directory containing the relevant hive files was specified. After introducing the test USB drives, a second snapshot of the Registry hives was captured. The two snapshots were compared, and differences were documented in HTML reports for each USB drive scenario. The generated HTML reports were meticulously searched for artefacts identified during the review process [49]. Additionally, RegRipper plugins targeting the NTUser.dat, System hive, and Software hive were executed on the corresponding registry hives extracted post-thumb drive introduction. The parsing results were systematically documented in detailed reports. The extracted data was then organized into Excel spreadsheets and consolidated.

2.5. Related Works

In computer forensics, registry forensics is vital due to the large available data and its importance. Efficiently, extracting this data is significant for forensic investigation [11]. However, the limitation of tools has caused the extraction to be undertaken manually. The nature of the textit{.REG} file contributes to the difficulty of the examination process. Data is stored by registry files under a value labelled *Key* [59]. The registry contains five most hierarchical folders called .hive, which begin with HKEY. Out of the five hives, only two are active HKU (HKEY_USER) and HKLM (HKEY_LOCAL_MACHINE). The rest serve as shortcut subfolders of the two. All registry keys have associated values labelled "Last_Write" time. This is information on when the last modification of the Registry key was made, and the information is stored as FILE_TIME structure. The Last_Write time update is made whenever the Registry key is created, accessed, modified and deleted [7]. Figuring this out in an automated approach helps the forensic analyst save a significant amount of time compared to doing it manually. In addition to this, the traditional manual approach has difficulties with determining what part of the key is changed even after manually finding out that the key is altered [35].

Specific work in post-incident malware investigations has incorporated RL to efficiently process memory dumps, network data, and registry hives. By defining registry artefacts as part of the state space in an MDP model, RL-based systems can classify and prioritise malicious registry entries, improving both speed and accuracy. This approach integrates Q-learning algorithms to train agents on actions such as identifying anomalies, collecting forensic data, and correlating registry changes with attack vectors [9]. The intervention of technology has rapidly improved the quality of forensic science, especially in the field of digital forensics. However, the need for speedy and intelligent investigation still exists to maintain an elevated position on digital crime [33]. ProDiscover, which is a digital forensic tool used to generate, and analyse data. It can be used to analyse a broad range of data and generate reports autonomously. It can also be used for searching specific data based on the node-enabled.

In [32], researchers proposed ARXEL an interacting GUI which is an automated tool for Windows registry analysis that has the capability of extracting data from E01 image. AXREL uses interacting NTFSDUMP which then outputs results in a table mode, containing key registry information that can be used for manual investigation with keyword search that is related to specific events. However, this tool only filters System Info, Program Execution, Autoruns, and Installed Software. ARXEL is time efficient and productive, However, it must be noted that it does not analyse all necessary data. It also needs to be manually verified for accuracy and relevance reasons.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

MultiAgent digital investigation toolkit (MADIK) proposed by [36] is a tool used to assist cyber incident response in forensic examination. It deploys an intelligent software agent (ISA), a system that uses AI to achieve defined goals in an autonomous manner. The tool performs different analyses related to a given case in a distributed manner [9]. MADIK contains six intelligent agents including HashSetAgent, TimelineAgent, FileSignatureAgent, FilePathAgent, KeywordAgent, and WindowsRegistryAgent. The WindowsRegisteryAgent examines information related to time zones, installed software, removable media information, and others [36]. Artificial intelligence-enabled forensic analysis has become outstanding in its contribution to detecting and preventing digital crimes [9]. In AI, machine learning has four main categories: supervised learning, semi-supervised, unsupervised learning and reinforcement learning. Reinforcement learning works by rewarding good behaviours and punishing behaviours that are not good enough. Various Machine Learning approaches including KNN, DT, SVM, PCA, K-Means, SVD, NB, ANN, LR, and RF have been deployed for forensic analysis. Windows registry is a huge data source for forensic analysis, and advantage must be taken on AI techniques to analyse Windows registry [39].

Kroll Artefact Parser and Extractoris (KAPE) is a multifunctional digital forensic tool developed for efficient data collection and parsing of forensically relevant artefacts [46]. It uses a target-based approach to data acquisition. Targets are predefined instructions for KAPE, identified by their *tkape* extension, directing it to locate specific files and folders on a computer. Upon the identification of items that align with the predetermined criteria, KAPE methodically duplicates the pertinent files and folders for subsequent analysis. The tool comprises an array of default targets, including the SANS_Triage compound collection, which enhances the efficiency of triage and evidence collection. Furthermore, various programs are run against the files, and the output from the programs is then saved in directories named after a category, such as Evidence_Of_Execution, Browser_History or Account_Usage. Table 3 summarise the most relevant research works proposing impactful tools or frameworks.

Table 3. Summary of Related Works.

Reference	Tool - Framework	Technique and Approach	Output & Description
[8]	TREDE & VM-POP	(1) Practical methodology to develop synthetic reference datasets in security and digital forensics. (2) Dataset automated generation feasibility and effectiveness through practical deployment and values assessment.	Generating a synthetic corpus into two different classes: user-generated and system-generated reference data.
[18]	ReLOAD	(1) A digital forensic tool used to generate and analyse data. (2) It utilises hardware resources and highlights deleted files	has the ability to analyse a broad range of data.
[25]	AXREL	(1) Interactive GUI for Windows registry analysis (2) Extracts data from Eo1 image using NTFSDUMP, and outputs results in a table format.	Filters system info, Program execution, and Installed Software. It is time efficient, although it requires manual verification.
[36]	MADIK	(1) MultiAgent digital investigation toolkit that utilises intelligent software agents (ISA) to analyse OSs forensics extracted Data. (2) Contains six agents: HashSetAgent, TimelineAgent, FileSignatureAgent, FilePathAgent, KeywordAgent, and WindowsRegistryAgent.	Processes and Analyses Time Zone info, installed software, removable media info, and OS events.
[44]	RaDaR	(1) multi-perspective data collection and labelling of malware activity. (2) Contains 7 million network packets, 11.3 million OS system call traces, and 3.3 million events related to 10,434 malware samples.	Open real-world datasets for runtime behavioural analysis of Windows malware. Dataset offers a comparison of different solutions and fosters multiple verticals in malware research.
[46]	Kroll KAPE	Kroll artefact Parser And Extractor automates the extraction of key forensic artefacts and supports rapid analysis.	Identification module to identify and collect specific forensic artefacts from systems. Parsing module targets specify what to collect (e.g., registry keys, logs), and the processing module to analyse and collects relevant artefacts.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

3. Research Methodology

This section provides an outline of the research methodology followed in this work. These steps are summarized below in a manner which enables grasping the Windows Registry (WinRegRL) Forensics domains and components and understanding the interaction between the different entities and the human expert. The research methodology used in this research has five steps:

- Current Methods Review for Windows Registry Analysis Automation and study the mechanisms, limitations, and challenges of existing methods in light of the requirements set by Windows Registry forensics for efficiency and accuracy.
- Investigating Techniques and Approaches used in Windows Registry Forensics. we particularly focused on machine learning approaches and rule-based reasoning systems that can extend or replace human intervention in the sequential decision-making processes involved in Windows Registry forensics. Find the best ways through which expert systems can be integrated into the forensics process in order to gain better investigative results.
- WinRegEL Framework Developmentas we progress with the design and implemention of diffirenet modules making the WinRegRL framework, we first focus on the WinRegRL-Core which introduce a new MDP model and Reinforcement Learning to optimize and enhance Windows Registry investigations. Also, add a module to capture, process, generalize, and feed human expertise into the RL environment using a Rule-Based Expert System (RBES) via Pyke Python Knowledge Engine.
- Testing and Validation of WinRegRL on the latest research and industry standards datasets made to mimic real-world DFIR scenarios and covering different size evidences especially in large-scale Windows Registry forensics. The performance of the framework should be evaluated against traditional methods and human experts in terms of efficiency, accuracy, time reduction, and artefact exploration. Refine the framework iteratively based on feedback testing for robust and adaptive performance in diverse investigative contexts.
- Finalisation of the WinRegRL Framework through unifying reinforcement learning and rule-based reasoning to support efficient Windows Registry and volatile data investigations. the final testing will demonstrate how it reduces reliance on human expertise while increasing efficiency, accuracy, and repeatability significantly, especially in cases with multiple machines of similar configurations.

3.1. Windows Registry and Volatile Data Markov Decision Process

A Markov Decision Process (MDP) is commonly used to mathematically formalize RL problems. The key components of an MDP include the state space, action space, transition probabilities, and reward function. This formalism provides a structured approach to implement RL algorithms, ensuring that agents can learn robust policies for complex sequential decision-making tasks across diverse application domains[53]. Markov decision process (MDP) is widely adopted to model sequential decision-making process problems to automate and optimise these processes [56]. The components of an MDP environment are summarised in equation 1:

A **Markov Decision Process**, MDP, is a 5-tuple (S, A, P, R, γ) where:

$$\begin{aligned} &\text{finite set of of states:} \\ &s \in S \\ &\text{finite set of actions:} \\ &a \in A \\ &\text{state transition probabilities:} \\ &p(s'|s,a) = Pr\{S_{t+1} = s'|S_t = s, A_t = a\} \\ &\text{expected reward for state-action-next\_state:} \\ &r(s',s,a) = \mathbb{E}[R_{t+1}|S_{t+1} = s', S_t = s, A_t = a] \end{aligned} \tag{1}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

Value Function: Value function describes *how good* is to be in a specific state s under a certain policy π. For MDP:

$$V_\pi(s) = \mathbb{E}[G_t|S_t = s] \tag{2}$$

To achieve the optimal value function, WinRegRL Agent will calculate the expected reward return when starting from s and following π but account for the cumulative discounted reward.

$$V_*(s) = \max_\pi V_\pi(s) \tag{3}$$

WinRegRL Agent at each step t receives a representation of the environment's *state*, $S_t \in S$ and it selects an action $A_t \in A(s)$. Then, as a consequence of its action the agent receives a *reward*, $R_{t+1} \in R \in \mathbb{R}$.

Policy: WinRegRL *policy* is a mapping from a state to an action

$$\pi_t(s|a) \tag{4}$$

That is the probability of select an action $A_t = a$ if $S_t = s$.

Reward: The total *reward* is expressed as:

$$G_t = \sum_{k=0}^{H} \gamma^k r_{t+k+1} \tag{5}$$

Where γ is the *discount factor* and H is the *horizon*, that can be infinite.

3.2. Overall Windows Registry and Timeline MDP Modelling

Markov decision processes (MDPs) provide a natural representation for DFIR investigator sequential decision making including browsing, interacting and performing actions on registry files and waiting for the outcome to validate the finding or exclude a scenario. As with any sequential decision process, automation could be enhanced considerably using RL, it involves an agent that interacts synchronously with the its environment or system; the sequence of system states are modelled as a stochastic process. The RL agent's goal is to maximize reward by choosing appropriate actions. These actions and the history of the environment states determine the probability distribution over possible next states. This MDP framework provides a structured approach for modelling and optimizing forensic investigations, enabling analysts to make statistically informed decisions in the presence of complex Windows Registry and volatile memory analysis.

Our proposed WinRegRL MDP illustrated in Figures 1 and 2 representation is unique and distinct by the complete capturing of the Registry Forensics and Timeline features of allowing the RL agent to act even when it fails to identify the exact environment state. The environment for RL in this context would be a simulation of the Windows operating system registry activities combined with real-world data logs. This setup should allow the agent to interact with a variety of registry states and receive feedback. The MDP model elaboration is operated following SANS including File Systems, Registry Forensic, Jump Lists, Timeline, Peripheral Profiling and Journaling as summarised in Table 2.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

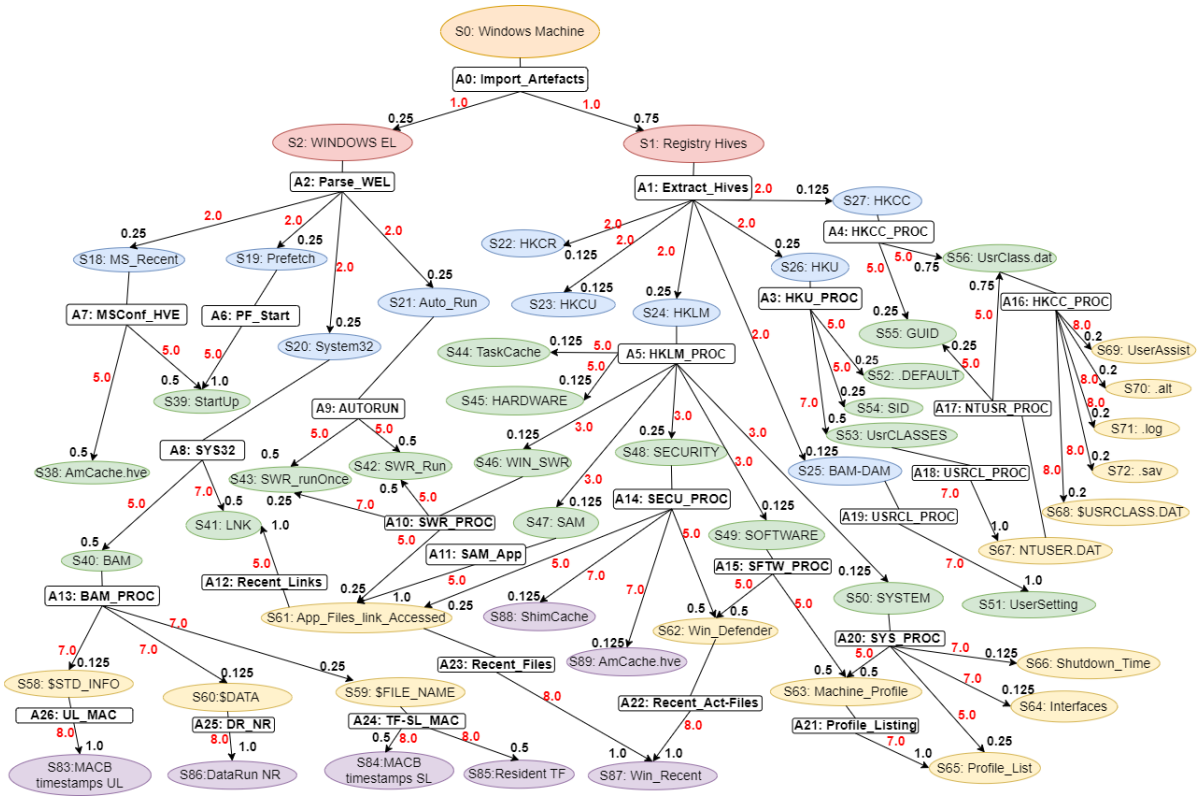

Figure 1. Detailed Windows 10 Registry Hives Only MDP States, Actions, Transition Probabilities and Rewards.

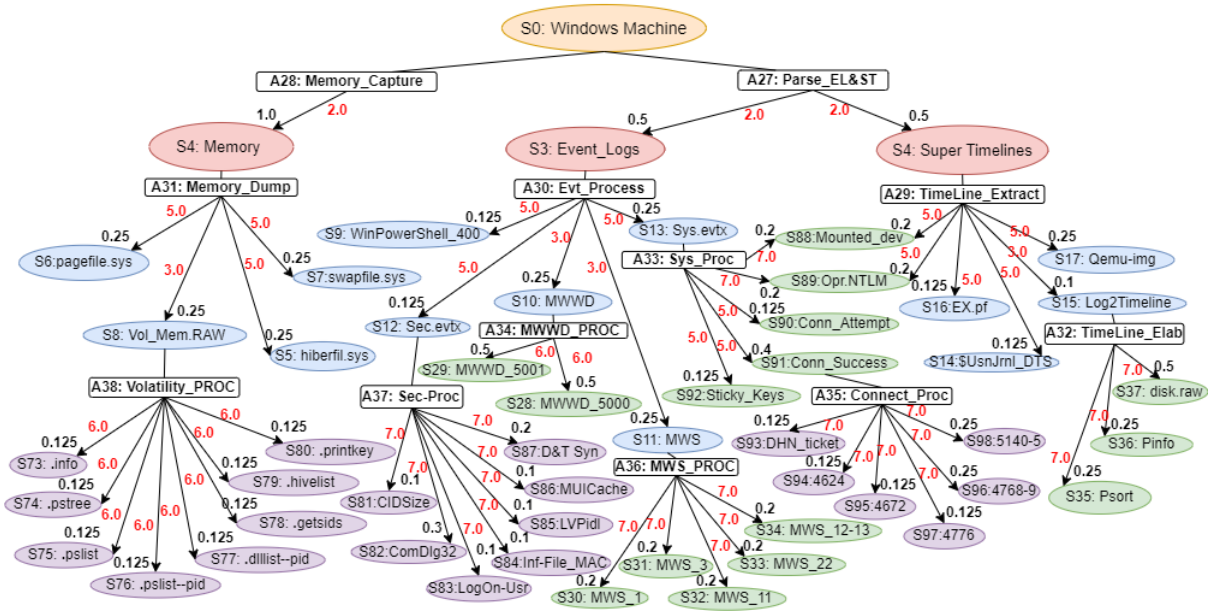

Figure 2. Detailed Windows 10 RAM Memory, Timeline and Events MDP States, Actions, Transition Probabilities and Rewards.

- **State Space:** In Windows Registry and volatile memory forensics, the state space refers to the status of a system at some time points. Such states include the structure and values of Registry keys, the presence of volatile artefacts, and other forensic evidence. The state space captures a finite and plausible set of such abstractions, allowing the modeling of the investigative process as an MDP. For example, states could be "Registry key exists with specific value" or "Volatile memory contains a given process artefact."
- **Actions Space:** Actions denote forensic steps that move the system from one state to another. Examples of these are Registry investigation, memory dump analysis, or timeline artefact correla-

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

tion. Every action can probabilistically progress to one of many possible states. The latter reflect inherent uncertainties of forensic methods: for example, querying a given Registry key might show some value with some probability and guide the investigator with future actions.

- **Transition Function:** The transition function models the probability of moving from one state another after performing an action. For example, examining a process in memory might lead to identifying associated Registry keys or other volatile artefacts with a given likelihood. The transition probabilities capture the dynamics of the forensic investigation, reflecting how evidence unfolds as actions are performed.
- **Reward Function:** The reward function assigns a numerical value to state-action pairs, representing the immediate benefit obtained by taking a specific action in a given state. In the forensics context, rewards can quantify the importance of discovering critical evidence or reducing uncertainty in the investigation process. For example, finding a timestamp in the Registry that corresponds to one of the known events can be assigned a large reward for solving the case.

This MDP framework provides a structured approach for modelling and optimizing forensic investigations, enabling analysts to make statistically informed decisions in the presence of complex Windows Registry and volatile memory analysis. The MDP model elaboration is operated following SANS including File Systems, Registry Forensic, Jump Lists, Timeline, Peripheral Profiling and Journaling as summarised in Table 2.

The problem of registry analysis can be defined as an optimisation problem focused on the agent, where it is to determine the best sequence of actions so as to maximize long-term rewards, thus obtaining the best possible investigation strategies under each system state. Reinforcement Learning (RL) provides a broad framework for addressing this challenge through flexible decision-making in complex sequential situations. Over the past decade, RL has revolutionized planning, problem-solving, and cognitive science, evolving from a repository of computational techniques into a versatile framework for modelling decision-making and learning processes in diverse fields [52]. Its adaptability has, therefore, stretched beyond the traditional domains of gaming and robotics to address real-world challenges ranging from cybersecurity to digital forensics.

in the proposed MDP model, an investigative process can be modelled as a continuous interaction between an agent—forensic tool or algorithm—and its environment—Windows Registry structure. The environment is formally defined by the set of states representing the registry's hierarchical keys, values, and metadata; the agent, on the other hand, selects actions in a number of states at scanning, analysing, or extracting data from it. Every action transitions the environment to a new state and provides feedback in the form of rewards, which guide the agent's learning process.

The sequential decision-making inherent in registry forensics involves navigating a series of states—from an initial registry snapshot to actionable insights. Feedback in this context includes positive rewards for identifying anomalies, detecting malicious artefacts, or uncovering system misconfigurations and negative rewards for redundant or ineffective actions. The RL agent will learn how to optimize its analysis by interacting iteratively with the environment and refining its policy; hence, it will raise the efficiency of finding important forensic artefacts systematically. That would automate and speed up registry analysis, additionally making it adaptive to changing attack patterns—what makes RL a very valuable tool for modern Windows Registry forensics.

RL enables an autonomous system to learn from its experiences using rewards and punishments derived from its actions. By exploring its environment through trial and error, the RL agent refines its decision-making process [50]. As training RL agents requires a large number of samples from an environment to reach human-level performance [45], we have opted for WinRegRL for an efficient approach by adopting model-based RL instead of training an agent's policy network using actual environment samples. We only use dataset samples to train a separate model being supervised by a certified Windows Registry Forensics having acquired GCFE or GCFA. This phase enabled us to predict the environment's behaviour, and then use this transition dynamics model to generate samples for learning the agent's policy [15].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

3.3. Reinforcement Learning

In this system, an RL agent interacts with its surround in a sequence of steps in time (t): 1) the agent receives a representation of the environment (state); 2)) selects and executes an action; 3 receives the reinforcement signal; 4) updates the learning matrix; 5) observe the new state of the environment.

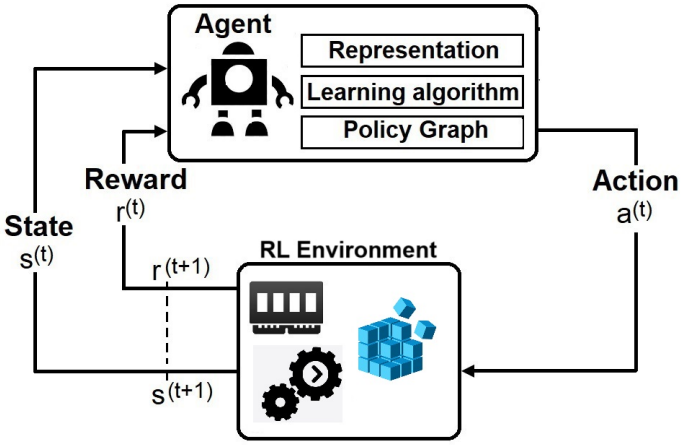

Figure 3. The Reinforcement Learning conceptual framework in the context of Windows Registry and Volatile Data Forensic analysis.

The RL Agent at each step t receives a representation of the environment's *state*, $S_t \in S$ and it selects an action $A_t \in A(s)$. Then, as a consequence of its action the agent receives a *reward*, $R_{t+1} \in R \in \mathbb{R}$. RL is about learning a policy (Π) that maximises the reinforcement values. A policy defines the agent's behaviour, mapping states into actions. The ϵ-greedy method is an example of the action selection policy adopted in RL. We are seeking an Exact solution to our MDP problems and initially considered the following methods:

- Value Iteration
- Policy Iteration
- Linear Programming

3.3.1. Value Iteration and Action-Value (Q) Function

We denoted the expected reward for state and action pairs in equation 6.

$$q_\pi(s,a) = \mathbb{E}_\pi\Big[G_t | S_t = s, A_t = a\Big] \tag{6}$$

3.3.2. Optimal Decision Policy

WinRegRL optimal value-action decision policy is formulated by the following function:

$$q_*(s,a) = \max_\pi q^\pi(s,a) \tag{7}$$

We can then redefine V^*, equation 3, using $q^*(s,a)$, equation 7:

$$V_*(s) = \max_{a \in A(s)} q_{\pi*}(s,a) \tag{8}$$

This equation conveys that the value of a state under the optimal policy **must be equal** to the expected return from the best action taken from that state. Rather than waiting for V(s)V(s) to converge, we can perform policy improvement and a truncated policy evaluation step in a single operation [28]

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

Algorithm 1 WinRegRL Value Iteration

Require: State space S, action space A, transition probability $p(s', r \mid s, a)$, discount factor $\gamma \in [0, 1)$, small threshold $\theta > 0$

Ensure: Optimal value function V_* and deterministic policy π_*

Initialization:
$V(s) \leftarrow 0,\ \forall s \in S$ ▷ Initialize value function

repeat
  $\Delta \leftarrow 0$
  for all $s \in S$ **do**
    $v \leftarrow V(s)$ ▷ Store current value of s
    $V(s) \leftarrow \max_{a \in A} \sum_{s', r} p(s', r \mid s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ ▷ Update the maximum change
  end for
until $\Delta < \theta$ ▷ Check for convergence

Extract Policy:
for all $s \in S$ **do**
  $\pi(s) \leftarrow \arg\max_{a \in A} \sum_{s', r} p(s', r \mid s, a)[r + \gamma V(s')]$
end for
Output: Deterministic policy $\pi \approx \pi_*$ and value function $V \approx V_*$

Monte Carlo (MC) is a *model-free* method, meaning it does not require complete knowledge of the environment. It relies on **averaging sample returns** for each state-action pair [28]. The following algorithm outlines the fundamental implementation, as shown in Figure 4.

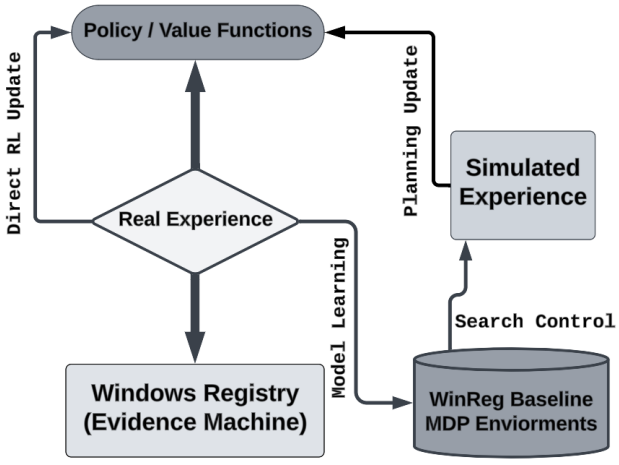

Figure 4. WinRegRL Reinforcement Learning Learning Approach.

3.3.3. Bellman Equation

An important recursive property emerges for both Value (2) and Q (6) functions if we expand them [51].

3.3.4. Value Function

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma V_\pi(s') \Big] \tag{9}$$

Similarly, we can do the same for the Q function:

$$q_\pi(s, a) = \sum_{s', r} p(s', r|s, a) \Big[r + \gamma V_\pi(s') \Big] \tag{10}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

3.3.5. Policy Iteration

We can now find the optimal policy

Algorithm 2 WinRegRL Policy Iteration

Require: State space S, action space A, transition probability $p(s', r \mid s, a)$, discount factor $\gamma \in [0, 1)$, small threshold $\theta > 0$
Ensure: Optimal value function V_* and policy π_*

Initialization:
$V(s) \leftarrow 0,\ \forall s \in S$ ▷ Initialize value function
$\pi(s) \in A,\ \forall s \in S$ ▷ Initialize arbitrary policy

repeat
  Policy Evaluation:
  $\Delta \leftarrow 0$
  for all $s \in S$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_{a \in A} \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \theta$ ▷ Repeat until value function converges

Policy Improvement:
policy_stable $\leftarrow$ true
for all $s \in S$ **do**
  old_action $\leftarrow \pi(s)$
  $\pi(s) \leftarrow \arg\max_{a \in A} \sum_{s', r} p(s', r \mid s, a)[r + \gamma V(s')]$
  if old_action $\neq \pi(s)$ **then**
    policy_stable $\leftarrow$ false
  end if
end for

if policy_stable **then**
  return $V \approx V_*$ and $\pi \approx \pi_*$
else
  go to Policy Evaluation
end if

3.4. WinRegRL Framework Design and Implementation

The proposed WinRegRL framework illustrated in Figure 5 integrates data from the Windows Registry and volatile memory into a decision-making system that models these data as MDP Environments, it also relies on Rule-Based Expert System (RBES) and other Machine Learning algorithms during the phase of MDP environments elaboration. It enables forensic examiners to validate decisions, identify patterns in fraudulent or malicious activities, and generate actionable reports.

Figure 5 provides a detailed diagram illustrating WinRegRL functioning covering Windows Registry Hives and Volatile Data Parsing, MDP environment elaboration, RL and RBES functioning as well as the human expert (GCFE) validation. The Diagram only consider Windows Registry (WinReg) analysis and volatile memory (WinMem-TL) data for elaborating MDP envisionments. This framework highlights an iterative decision-making process that combines expert knowledge with Reinforcement Learning and the use of automated forensic tools to enhance the analysis and detection of fraudulent or anomalous activities in a Windows environment.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

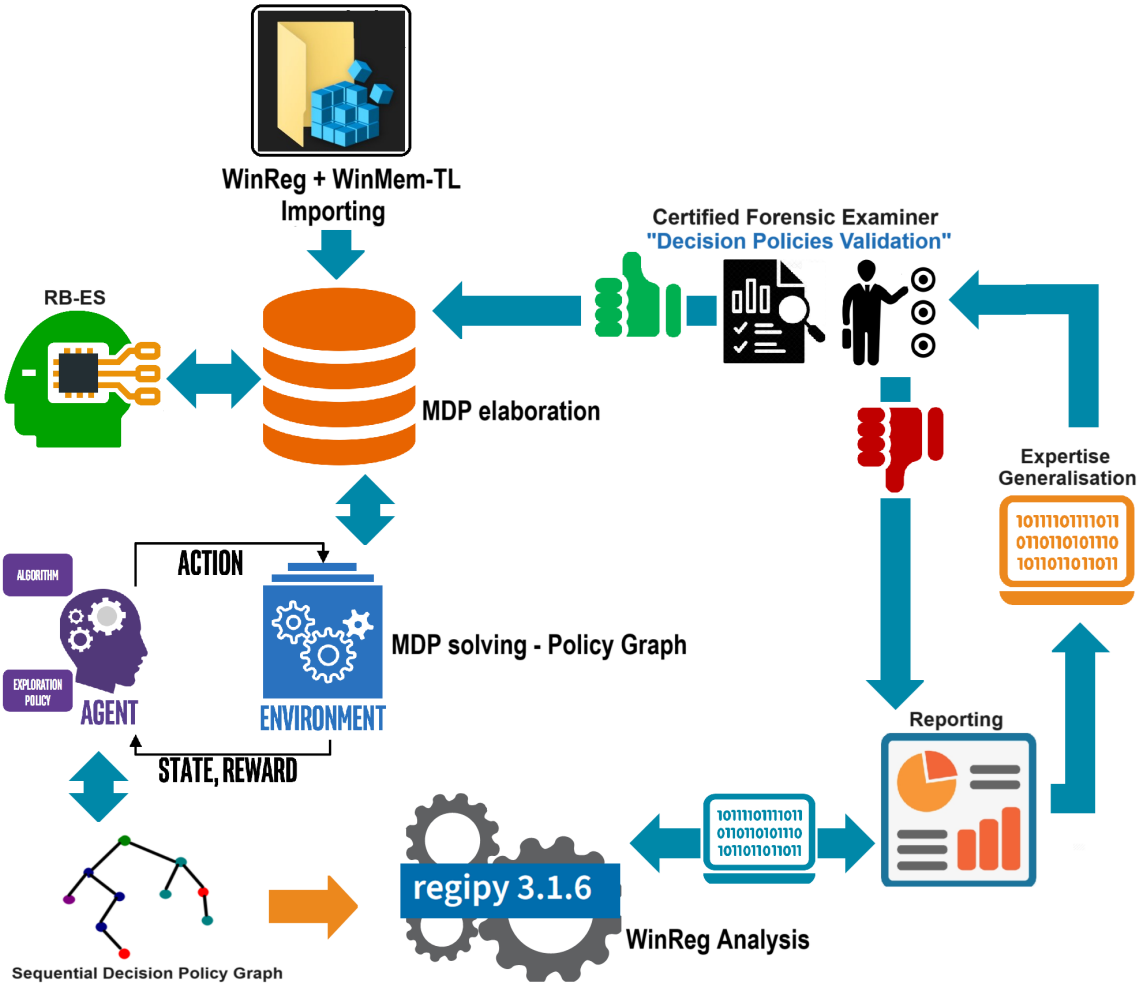

Figure 5. WinRegRL Framework detailed functioning diagram

We Describe here, WinRegRL framework modular breakdown:

WinReg + WinMem-TL Importing The process begins by importing Windows Registry data and memory timelines (WinMem-TL).

MDP Elaboration The imported data is transformed into a Markov Decision Process (MDP) environment following the structure defined in Equation 1.

MDP Solver The out-of-the-box MATLAB MDP toolbox contains various algorithms to solve the discrete-time Markov Decision Processes: backward induction, value iteration, policy iteration, and linear programming [19]. In the case of the analysis of the Windows Registry, an MDP agent—a purple head icon with the label AGENT—interacts with the registry environment to receive state observations and reward signals to improve its decision policies [21]. The RL agent will therefore systematically search through registry keys and values, identifying patterns and anomalies in a feedback-driven cycle of analysis. This interaction constructs a Sequential Decision Policy Graph to visualize possible actions and decision paths specific to registry analysis. An off-the-shelf MDP solver is then applied to find optimal policies, dynamically focusing on registry artefacts based on their reward importance. This methodology exposes, with great skill, hidden malicious behaviours or misconfigurations in the Windows Registry, yielding insights that are often overlooked by traditional static analysis or manual investigation techniques. The algorithm functioning is detailed in Subsection 3.7

3.5. Expertise Capturing and Generalisation

A Rule-Based Expert System (RBES) with an online integrator that interacts with the RL MDP-solver, likely applying expert knowledge or predefined rules to assist in elaborating decision policies based on the imported data [47].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

3.5.1. Expertise Generalisation

If the generated policies are inadequate, they undergo further Expertise Generalisation, where more complex or nuanced decision-making logic is applied to better handle fraud detection or anomaly identification.

3.5.2. Reporting and Analysis

The validated results are compiled into a report for decision-making or further investigation. The process feeds back into WinReg Analysis using Regipy 3.1.6, a tool for forensic analysis of Windows Registry hives, ensuring continual refinement and improvement of the decision-making model.

3.6. Human Expert Validation module

The generated decision policies are validated by a Certified Forensic Examiner holding at least GIAC GCFE or GCFA to ensure that the automatically generated decision policies are correct and appropriate. If the policies are valid, they receive a "thumbs up"; if not, further generalisation or adjustments are needed (denoted by the "thumbs down" icon).

3.7. WinRegRL-Core

WinRegRL-Core as illustrated in Algorithm 3 is the Core Reinforcement Learning algorithm where RL Agent is trained on a model MDP—the latter created based on some given data about the Windows Registry. The algorithm initialises the policy, value function, and generator with random values first, then updates each iteration of the policy of the agent based on the computed state-action values by using either policy iteration or value iteration. Policy iteration directly updates the policy by choosing those actions with maximum Q-values, while value iteration, is an iterative refinement of a value function V(s). Q-learning is used so the RL Agent would explore the environment and update a Q-table while following the epsilon-greedy approach for balancing exploration and exploitation. WinRegRL-Core incorporates increasingly improved iterations of the critic, or value function, and the generator, or policy, for the refinement of the policy towards the selection of appropriate actions for the maximisation of cumulative rewards. Convergence is verified based on whether the difference between the predicted and target policy is less than a specified threshold, which demonstrates that $\diamond$ is the optimal policy. The last trained generator $\mathcal{D}$ and the critic $\mathcal{E}$ represent the optimized policy and value function, Respectively, used for decision-making within the MDP environment.

For non-stationary problems, the Monte Carlo estimate for, e.g, V is:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big] \tag{11}$$

Where α is the learning rate, how much we want to forget about past experiences.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

Algorithm 3 WinRegRL-Core

Require: MDP (S, A, P, R, γ), where S is the set of states derived from Windows Registry data, A is the set of actions, P is the transition probability matrix, R is the reward function, and γ is the discount factor; Local iterations I, learning rate for Critic α_D, learning rate for Generator α_E, gradient penalty λ

Ensure: Trained Critic $\mathcal{D}$ and Generator $\mathcal{E}$ for the optimal policy π^*

Initialize Generator $\mathcal{E}$ with random weights
Initialize Critic (value function) $V(s) \leftarrow 0$ for each $s \in S$
Initialize policy $\pi(s)$ arbitrarily for each $s \in S$
for iteration $t = 1$ to I **do**
    while MDP not converged **do**
        for each state $s \in S$ **do**
            for each action $a \in A$ **do**
                Compute Bellman update for expected reward $Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s')$
            end for
            Update policy with max Q-value$\pi(s) \leftarrow \arg\max_a Q(s,a)$
            Update value $V(s) \leftarrow \max_a Q(s,a)$
        end for
        if using Policy Iteration **then**
            Check for stability of π; if stable, **break**
        else if using Value Iteration **then**
            Check convergence in $V(s)$; **break** if changes are below threshold
        end if
    end while
    if Q-Learning is needed **then**
        Initialize Q-table $Q(s,a) \leftarrow 0$ for all $(s,a) \in S \times A$
        for each episode until convergence **do**
            Observe initial state s
            while episode not done **do**
                Choose action a using ϵ-greedy policy based on $Q(s,a)$
                Execute action a, observe reward r and next state s'
                Update $Q(s,a) \leftarrow Q(s,a) + \alpha_D\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big)$
                $s \leftarrow s'$
            end while
        end for
    end if
    Train Critic $\mathcal{D}$ to update value function
    Train Generator $\mathcal{E}$ to optimize policy using updated critic values
    Confirm convergence of the target policy ≤ 0.1, **break**
end for
return Trained Generator $\mathcal{E}$ and Critic $\mathcal{D}$ for optimized policy π^*

3.8. PGGenExpert-Core

PGGenExpert is the proposed Expertise Extraction, Assessment and Generalisation Algorithm 4, it converts WinRegRL Policy Graph to Generalized Policies independently of the covered case for future use, processes and corrects a state-action policy graph derived from reinforcement learning on Windows Registry data. It starts by taking the original dataset, a complementary volatile dataset, and an expert-corrected policy graph as inputs. The algorithm first creates a copy of the action space and iterates over each feature to ensure actions remain within the valid range, correcting any out-of-bounds actions. Next, it addresses binary values by identifying incorrect actions that differ from expected binary states and substitutes them with expert-provided corrections. For each state-action pair, it then adjusts the action values so that only the highest-probability action remains non-zero, effectively refining the policy to select the most relevant actions. Finally, the algorithm returns the corrected and generalized policy graph, providing an updated model of action transitions that aligns with expert knowledge and domain constraints.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

Algorithm 4 GenExpert: Policy Graph Expertise Extraction, Assessment and Generalisation Algorithm

Require: Original Windows Registry data $W \in \mathbb{R}^{n \times d}$ with n instances and d features, Complementary volatile data Y with shape matching W, and State-to-Action Policy Graph Z. Expert-corrected general policy graph C.

Ensure: Updated transition probabilities in the MDP model based on corrected actions.

procedure CORRECT_ACTION_STATE_PG MAPING(W, Y, Z, B, C)
  Copy of the action space: $S' \leftarrow S$
  for $i \in R$ **do** ▷ Handle missing and out-of-range actions
    $v_{\min,i} \leftarrow \min(W_{:,i})$
    $v_{\max,i} \leftarrow \max(W_{:,i})$
    $S'_{:,i} \leftarrow \max(\min(S_{:,i}, v_{\max,i}), v_{\min,i})$ ▷ Bound actions within valid range
  end for
  for $i \in B$ **do** ▷ Correct binary values
    $Y'_{\text{incorrect},i} \leftarrow (Y_{:,i} \neq W_{:,i}) \wedge (Y_{:,i} \neq 1)$ ▷ Identify invalid actions
    $Y_{\text{corrected},i} \leftarrow Y_{:,i} \times (\neg Y'_{\text{incorrect},i}) + C_{:,i} \times Y'_{\text{incorrect},i}$ ▷ Replace with expert-corrected values
    $Y'_{:,i} \leftarrow \boldsymbol{e}_{h_i}$ ▷ Set all but highest value in state-action vector to 0
  end for
  $h_i \leftarrow \text{argmax}(Y_{:,i})$ ▷ Extract generalized policy graph based on highest value action
  return Y' ▷ Return updated state-action policy graph
end procedure

4. Testing, Results and Discussion

4.1. Testing Datasets

To test WinRegRL, we opted for different DFIR datasets. Table 4 summarises the WinRegRL Testing Datasets which was a mixed selection of four (04) datasets that cover multiple Windows architectures and versions. In general, three datasets are widely adopted in DFIR research:

CFReDS: The NIST Computer Forensic Reference DataSet (CFReDS) Portal is a comprehensive resource for accessing documented digital forensic image datasets and supporting tasks such as tool testing, practitioner training, and research. Each dataset includes descriptions of significant artefacts and can be searched by year, author, or attributes. The portal features collections developed by NIST for projects like Computer Forensic Tool Testing (CFTT) and Federated Testing, alongside contributions from other sources. These datasets enable users to evaluate forensic tools, develop expertise, train professionals, and conduct custom studies, offering a valuable foundation for advancing digital forensics practices [12].

TREDE and VMPOP: The dataset includes four Windows Registry files from different PCs, providing critical insights for system security and digital forensics. The Windows Registry, storing essential system and application configurations, helps identify vulnerabilities and forensic artefacts. As the PCs were inactive during extraction, no active threats were present. Three files are raw datasets directly extracted using Windows command-line tools, maintaining their original integrity. The fourth file is a reformatted key-value pair version of one raw dataset, preserving its semantic accuracy [8]. This structure supports in-depth analysis, uncovering traces of unauthorized access, malware persistence, and other forensic evidence.

DWosFA: This dataset contains Windows forensic artefacts, including the NTFS file system and event log records. This dataset was created from disk images of devices used in CTF competitions, mainly focusing on the forensics of the Windows operating system and analysis of user activity. Security incident timelines were extracted using the Plaso tool, allowing creation a timeline of events in order from these disk images. The timeline attributes were then converted into binary values for easier data analysis and machine learning. The final dataset is divided into 13 separate files, saved in CSV format for the purpose of organized analysis [10].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

Table 4. WinRegRL Testing Forensics Datasets.

Ref.	Datasets	Description
[54]	Magnet-CTF 2022	Dataset developed by Magnet Forensics including several forensic image as a public service part of Magnet CTF-2022. Datasets made of Registry and Memory of different Windows Machines (10, 8, 7 and XP).
[55]	IGU-CTF 2024	Dataset was taken from IGUCTF-24 which include CVE-2023-38831 Lockbit 3.0 Ransomware. Dataset is made of Registry and Memory of infected Windows machine capturing Powershell2exe and Persistence tactics.
[57]	MemLabs-CTF 2019	Dataset developed by MemLabs with CTF Registry and Memory Forensics. All the labs are of Windows 7. Datasets made of Registry and Memory of Six (06) different Windows 7 Machines.
[58]	NPS-Domexusers 2009	Digital-Corpora, Two disk images including full system Registry of Widows XP executables redacted so that they cannot be executed.

4.2. Reinforcement Learning Parameters and Variables

The WinRegRL reinforcement learning model uses important parameters to optimise decision-making. The discount factor of **0.99**, seeks long-term rewards by properly weighting immediate and future gains. Decision epochs of **200** was selected to determine the maximum number of steps per episode. The value function tolerance is set to **0.001**, determines the precision of convergence, and epsilon of 0.00001, controls the threshold of value improvement. The learning generator rate was set between [0–99%] which determines the change of policies; the reward and penalty function [-10, +100]) adjusts the feedback mechanism. Expertise thresholds is set between [0–99%] and interpolated samples of 0.01 to enhance performance. Generalization loss is set between [0, +100] to evaluate how well expertises are generalised. A buffer size of 100,000 allows a memory replay, which is controlled by variant batch sizes [1024, 512, 256]. Table 5 summarises the variables and parameters we selected for running WinRegRL Experiments.

Table 5. Reinforcement Learning Experiments Variables and Parameters

Parameter	Value	Description
γ	[0 – 1]	Discount Factor value is determined by testing - γ adopted is 0.99)
H	200	Decision Epochs; Maximum number of steps allowed per episode
$\mathcal{T}$	0.001	Value Function Tolerance where the algorithm will if two successive iterations are smaller
ϵ	0.00001	Vectors are only added if the value improvement exceeds Epsilon
α	[0 - 99%]	Learning Generator rate
λ	$[-10, +100]$	Reward and Penalty Function
$\mathcal{E}$	[0 - 99%]	Expertise Extractor Thresholds
$\tilde{x}$	0.01	Interpolated sample
$\mathcal{L}_{\text{gen}}$	$[0, +100]$	Expertise Generalization Loss
Buffer Size	100,000	The size of a replay buffer (Learning from older samples) including Memory usage
Batch Size	[1024, 512, 256]	The number of samples trained in neural work
Gradient Steps	3	Parameter based on the change in weights in relation to a change in error Number of steps to update a policy
Exploration Fraction	0.1	How much of the training time does the algorithm spend exploring
Exploration Final Eps	0.02	Updating after an episode will take longer to converge but offers more exploration

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

4.3. WinRegRL Results

Figure 6 illustrates the evaluation of the WinRegRL Framework's performance in different scenarios of Markov Decision Processes (MDPs), including the Full Activities REG Investigation MDP, the Data Exfiltration REG Investigation MDP, and the Malware Artefacts REG Investigation MDP. It is contrasted with the baseline time taken by a GCFE Human Expert Forensics Examiner (green dashed) used only as a reference point. The horizontal axis represents the Windows registry size in MB, while the vertical axis represents the time in seconds it took to run the respective MDP tasks. The results show that Full Activities REG Investigation MDP always takes the longest time, followed by Data Exfiltration REG Investigation MDP, then Malware Artefacts REG Investigation MDP—showing how different the task complexities are. Notably, the GCFE Human Expert shows a linear and considerably lower time requirement in smaller datasets but points out scalability limitations when dealing with larger registries.

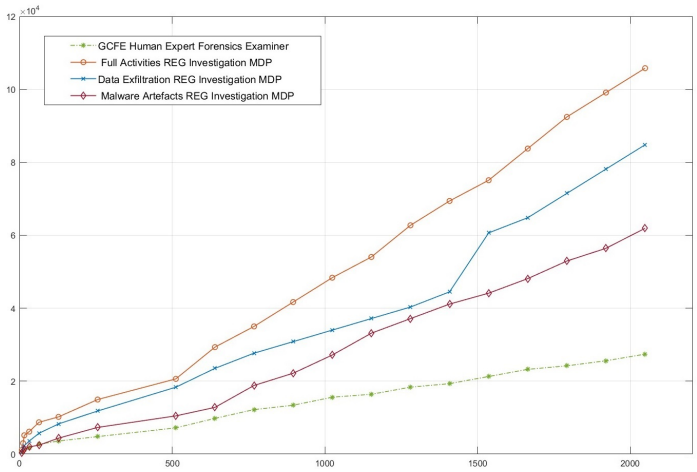

Figure 6. Results Solving MDP problem in different envisionments.

On the other hand, Figure 7 shows a comparative analysis of the WinRegRL Framework's performance regarding various Windows registry sizes (on the x-axis) and the time required to complete forensic analyses (on the y-axis, in seconds). Four approaches are compared: Full Automation FTK Reg Viewer with Regipy, GCFE Human Expert Forensics Examiner, MDP (RL) with Pre-Processing, and MDP (RL) with Pre-Processing and RBES. Results show that the fully automated approach vastly outperforms all others, with a sharp rise in efficiency as the registry size scales. The MDP (RL) with Pre-Processing follows, showing moderate scalability, while the inclusion of RBES further optimizes performance but remains below full automation. By comparison, the GCFE Human Expert Forensics Examiner method is the least efficient, with the investigation time increasing linearly and falling behind both automated and reinforcement learning-based methods by a large margin. This comparison empowers the scalable nature and effectiveness of automated and AI-driven approaches in handling large-scale registry investigations.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

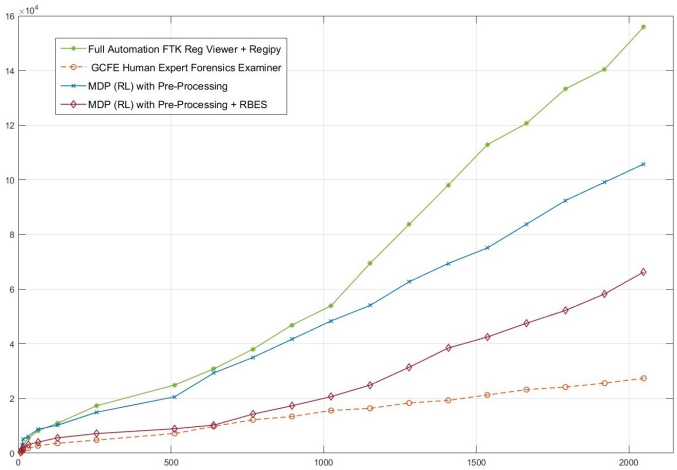

Figure 7. WinRegRL Automation Results.

Figure 8 shows the efficacy of the WinRegRL Framework in three cases of investigation, including malware artefacts, Full Windows REgistry with Timeline (WinReg+TL) Investigation, data exfiltration WinReg+TL Investigation, and Full Activities WinReg+TL Investigation. The x-axis represents the size of the Windows registry, while the y-axis shows the number of artefacts found during each investigation. Results show that the Full Activities WinReg+TL Investigation always finds the most artefacts, and this number goes up quickly as the registry size gets bigger. The Data Exfiltration WinReg+TL Investigation comes next, Showing a fair rate of finding artefacts with a steady increase. On the other hand, the Malware Artefacts WinReg+TL Investigation finds the least artefacts with a slower growth rate and a flat level seen at larger registry sizes. These results indicate that the level of complexity and the quantity of artefacts vary in divergent investigation situations. They emphasize the fact that full-activity investigations span more ground than focused malware and data theft studies.

Figure 9 illustrates the performance of the WinRegRL framework tested under varying Windows registry sizes (x-axis) against execution time in seconds (y-axis) across three scenarios: full activities, data exfiltration, and malware artefacts, in comparison to a human expert (GCFE). The results demonstrate that while the GCFE maintains consistent performance, WinRegRL outperforms in all scenarios, especially in handling larger registry sizes, with significantly lower time costs and error-free operation. The full activities scenario exhibits the highest time efficiency gain, followed by data exfiltration and malware artefacts. The findings highlight WinRegRL's capability to deliver scalable, cost-effective, and error-free registry analysis compared to human expertise.

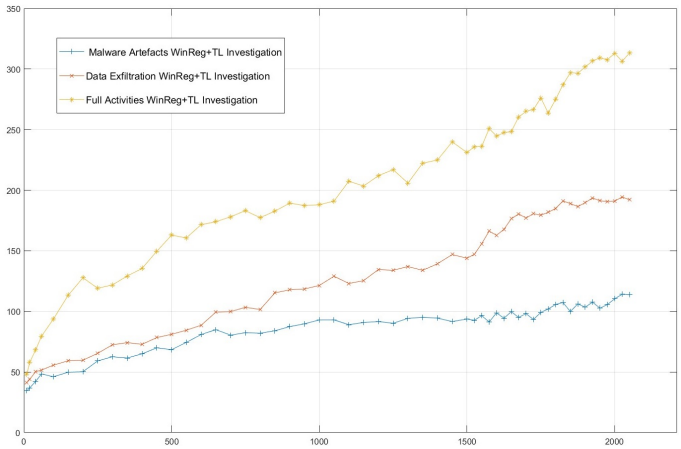

Figure 8. Overall number of relevant artefacts processed in a variable size Registry Hives with a Maximum value of 2048 Mb.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

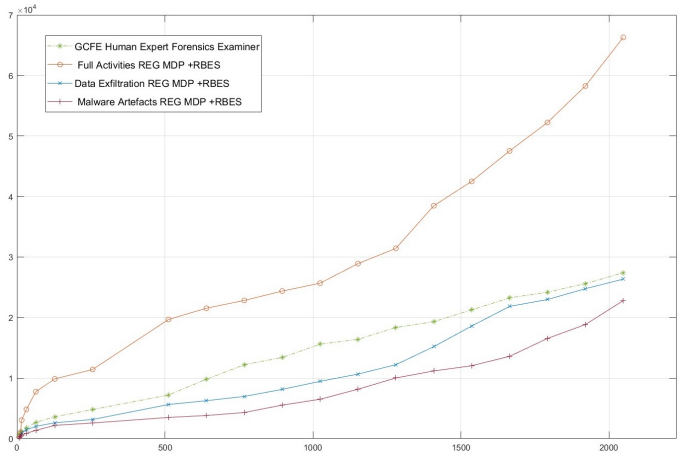

Figure 9. Results .

4.4. WinRegRL Performances Validation

Finally, we proceeded to the validation of the obtained results by running the same experiments on different forensic datasets. we summarised WinRegRL accuracy and performance benchmarking compared with industry tools when tested using the aforementioned four datasets summarised in Table 6 which validated the fact that WinRegRL outperforms both the automated full FTK Registry Viewer and Kroll KAPE in terms of the number of relevant artefacts discovered as well as the attribution of these later to the right evidence categories.

Table 6. WinRegRL accuracy and performance benchmarking compared with industry tools in 4 scenarios

	Artefacts Relevance Accuracy			Artefacts Attribution Performance		
Tool or Framework	**WinRegRL**	FTK + Register Viewer	KAPE	**WinRegRL**	FTK + Register Viewer	KAPE
Magnet-CTF 2022 [54]	**98.7%**	76.4%	81.4%	**100.0%**	63.9%	41.6%
IGU-CTF 2024 [55]	**98.5%**	77.2%	80.9%	**100.0%**	53.5%	46.8%
MemLabs-CTF 2019 [57]	**99.5%**	82.1%	88.7%	**100.0%**	66.7%	50.1%
NPS-Domexusers 2009 [58]	**100.0%**	90.7%	93.2%	**100.0%**	71.2%	52.5%

In terms of explainability and accuracy, we reflected WinRegRL output optimal PGs and mapped them to an expert GCFE-certified expert decision-making in terms of optimal action for each state. Figures 10 and 11 illustrates one example from a Windows 10 full investigation both covering Registry analysis and Volatile Data timeline analysis Respectively.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

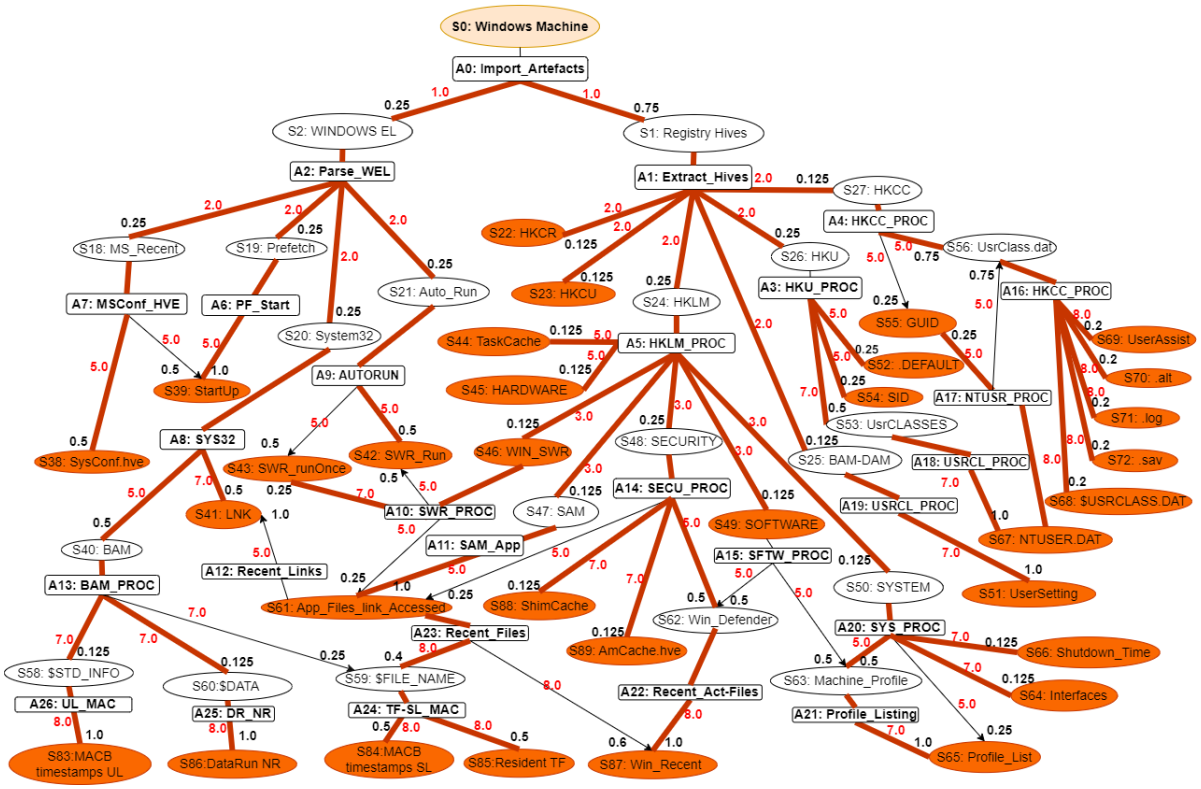

Figure 10. Summary of Validated WinRegRL PGs for Full Windows 10 Registry Hives Investigation.

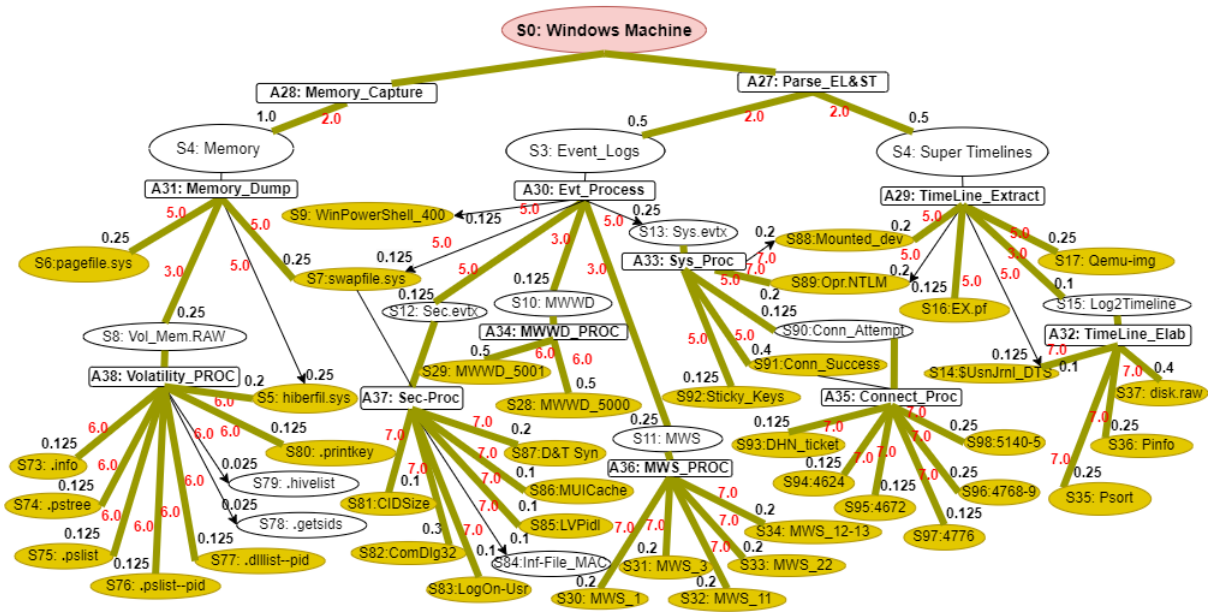

Figure 11. Summary of Validated WinRegRL PGs for Full Windows 10 Volatile Memory Timeline Investigation.

4.5. Discussion

The WinRegRL framework represents a new revolution in the analysis of the Windows Registry, offering a solution that is fast, secure, and efficient for investigations of cyber incidents. Thanks to the integration of reinforcement learning into this analytical framework, WinRegRL is capable of automating key forensic and security operations while addressing the dynamic and complex nature of the Windows Registry for anomaly, malware, and misconfiguration detection—commonly exploited by cybercriminals. In this ever-changing landscape of cybersecurity, RL enables WinRegRL to continually adapt and tune its decision-making capabilities. Such dynamic adaptability improves the ability of the framework to detect cyber threats, anomalies, and suspicious registry behaviours. Cyber

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

incidents, such as malware infections, often leave traces in the registry. WinRegRL accelerates and simplifies the identification of such threats, ensuring that responses during investigations are both fast and accurate. Traditional registry analysis is very time-consuming, error-prone, and hence risky. WinRegRL automates this process by leveraging RL to identify irregularities and suspicious activities with precision. The more the RL models see data, the better they become at recognizing subtle patterns that may go unnoticed by static rule-based systems or manual methods. This ensures a scalable and error-free approach to handling registry analysis. WinRegRL significantly enhances the detection of malicious artefacts in the Windows Registry and volatile memory. Reinforcement learning-driven automation enables fast analysis of registry data, drastically reducing the time required for detection and response. Moreover, the adaptive nature of the framework allows it to stay proactive with newly emerging threats, providing very important insights into innovative attack vectors that may evade traditional approaches. By underlining the aspects of speed, safety, and accuracy, WinRegRL places itself as a fundamental tool for modern digital forensics and incident response.

5. Conclusion and Future Works

This paper introduces a novel RL-based approach to enhance the current automation and data management in Windows registry forensics and timeline analysis in volatile data investigation. We have shown that it is possible to streamline the analysis process, reducing both the time and effort required to identify relevant forensic artefacts as well as increase accuracy and performance which will be translated into a better and more trusted output. Our proposed MDP model dynamically covers all registry hives, volatile data and key values based on their relevance to the investigation, and has proven effective in a variety of simulated attack scenarios. The model's ability to adapt to different investigation contexts and learn from previous experiences represents a significant advancement in the field of cyber forensics. This research addresses gaps in existing methodologies through the introduction of a novel MDP model representing the Windows environment, enabling efficient automation of the investigation process. It also tackles the waste of knowledge in the practice as RL Optimal policies are captured and generalised thus reducing investigation time by minimising MDP solving time. However, while the results are promising, several challenges remain. The effectiveness of the RL model heavily relies on the quality and the modelling which vary considerably in real-world scenarios and can be sparse or unbalanced notably in new Windows OSs. Ultimately, the application of RL in Windows Registry analysis holds significant promise for improving the speed and accuracy of cyber incident investigations, potentially leading to more timely and effective responses to cyber threats. As the cyber threat landscape continues to evolve, the development of intelligent, automated tools such as the one proposed in this study will be crucial in staying ahead of adversaries and ensuring the integrity of digital investigations.

While RL offers robust solutions for cyber incident forensics analysis, challenges remain in implementing scalable models for Windows Registry analysis. Existing studies highlight the complexity of crafting accurate state-action representations and addressing reward sparsity. Future work should refine the model to handle diverse and evolving threat landscapes more robustly [48]. This includes incorporating Large Language Models (LLMs) coupled with Retrieval-Augmented Generation (RAGs) techniques in defining the MDP environments from dynamic cyber incident response data, enhancing the model's generalisability, and ensuring its applicability across various versions of the Windows operating system.

References

1. StatCounter. 2024. Desktop Windows Version Market Share Worldwide for the period Oct 2023 - Oct 2024. Statcounter GlobalStats Report. https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide.
2. Microsoft Inc. 2024. Microsoft Digital Defense Report 2024 The foundations and new frontiers of cybersecurity. https://www.microsoft.com/en-us/security/security-insider/intelligence-reports/microsoft-digital-defense-report-2024.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

3. Casino, F., Dasaklis, T.K., Spathoulas, G.P., Anagnostopoulos, M., Ghosal, A., Borocz, I., Solanas, A., Conti, M. and Patsakis, C., 2022. Research trends, challenges, and emerging topics in digital forensics: A review of reviews. IEEE Access, 10, pp.25464-25493. https://doi.org/10.1109/ACCESS.2022.3154059
4. Yosifovich, P., Solomon, D.A., Russinovich, M.E. and Ionescu, A., 2017. Windows Internals: System architecture, processes, threads, memory management, and more, Part 1. Microsoft Press.https://dl.acm.org/doi/abs/10.5555/3158930
5. Singh, A., Venter, H.S. and Ikuesan, A.R., 2020. Windows registry harnesser for incident response and digital forensic analysis. Australian Journal of Forensic Sciences, 52(3), pp.337-353. https://doi.org/10.1080/00450618.2018.1551421.
6. Lee, J.H. and Kwon, H.Y., 2022. Large-scale digital forensic investigation for Windows registry on Apache Spark. Plos one, 17(12), p.e0267411. https://doi.org/10.1371/journal.pone.0267411.
7. Ghanem, M.C., Mulvihill, P., Ouazzane, K., Djemai, R. and Dunsin, D., 2023. D2WFP: a novel protocol for forensically identifying, extracting, and analysing deep and dark web browsing activities. Journal of Cybersecurity and Privacy, 3(4), pp.808-829. https://doi.org/10.3390/jcp3040036.
8. Park, J., 2018. TREDE and VMPOP: Cultivating multi-purpose datasets for digital forensics–A Windows registry corpus as an example. Digital Investigation, 26, pp.3-18. https://doi.org/10.1016/j.diin.2018.04.025.
9. Dunsin, D., Ghanem, M.C., Ouazzane, K., Vassilev, V., 2024. A comprehensive analysis of the role of artificial intelligence and machine learning in modern digital forensics and incident response. Forensic Science International. Digital Investigation 48, 301675. https://doi.org/10.1016/j.fsidi.2023.301675
10. Marková, Eva; Sokol, Pavol; Krišáková, Sophia Petra; Kováčová, Kristína. 2024. Dataset of Windows operating system forensics artefacts, Mendeley Data, V1.https://doi.org/10.17632/8dfh724hvc.1
11. Farzaan, M.A., Ghanem, M.C. and El-Hajjar, A., 2024. AI-Enabled System for Efficient and Effective Cyber Incident Detection and Response in Cloud Environments. https://arxiv.org/abs/2404.05602
12. Park, J., Lyle, J.R. and Guttman, B., 2016. Introduction to CFTT and CFReDS projects at NIST. https://cfreds.nist.gov/https://doi.org/10.1080/00450618.2018.1551421
13. Dunsin, D., Ghanem, M.C., Ouazzane, K. and Vassilev, V., 2024. Reinforcement Learning for an Efficient and Effective Malware Investigation during Cyber Incident Response. High-Confidence Computing Journal. https://doi.org/10.48550/arXiv.2408.01999.
14. Lee, Jun-Ha, and Hyuk-Yoon Kwon. 2020.Redundancy analysis and elimination on access patterns of the Windows applications based on I/O log data. IEEE Access 8 (2020): 40640-40655.https://doi.org/10.1109/ACCESS.2020.2964260
15. X. Wang, Z.Yang, G. Chen, Y. Liu. (2023). A Reinforcement Learning Method of Solving Markov Decision Processes: An Adaptive Exploration Model Based on Temporal Difference Error. Electronics. 12. 4176. https://doi.org/10.3390/electronics12194176.
16. T. Zhang, C. Xu, J. Shen, X. Kuang and L. A. Grieco, How to Disturb Network Reconnaissance: A Moving Target Defense Approach Based on Deep Reinforcement Learning,in IEEE Transactions on Information Forensics and Security, vol. 18, pp. 5735-5748, 2023, https://doi.org/10.1109/TIFS.2023.3314219.
17. A. Bashar, S. Muhammad, N. Mohammad and M. Khan, "Modeling and Analysis of MDP-based Security Risk Assessment System for Smart Grids", *2020 Fourth International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, India, 2020, pp. 25-30, https://doi.org/10.1109/10.1109/ICISC47916.2020.9171072.
18. X. Mo, S. Tan, W. Tang, B. Li and J. Huang, ReLOAD: Using Reinforcement Learning to Optimize Asymmetric Distortion for Additive Steganography. IEEE Transactions on Information Forensics and Security, vol. 18, pp. 1524-1538, 2023, https://doi.org/10.1109/TIFS.2023.3244094
19. Borkar, V. and Jain, R., 2014. Risk-constrained Markov decision processes. IEEE Transactions on Automatic Control, 59(9), pp.2574-2579.https://doi.org/10.1109/TAC.2014.2309262
20. Dunsin, D., Ghanem, M.C., Ouazzane, K. and Vassilev, V., 2024. A comprehensive analysis of the role of artificial intelligence and machine learning in modern digital forensics and incident response. Forensic Science International: Digital Investigation, 48, p.301675. https://doi.org/10.1016/j.fsidi.2023.301675.
21. Wang, C., Ju, P., Lei, S., Wang, Z., Wu, F. and Hou, Y., 2019. Markov decision process-based resilience enhancement for distribution systems: An approximate dynamic programming approach. IEEE Transactions on Smart Grid, 11(3), pp.2498-2510.https://doi.org/10.1109/TSG.2019.2956740
22. Kwon, Hyuk-Yoon (2022). Windows Registry Data Sets, Mendeley Data, V1, https://doi.org/10.17632/ghkms3cgbg.1

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

23. MChadès, I., Cros, M.J., Garcia, F. and Sabbadin, R., 2009. Markov decision processes (MDP) toolbox. http://www.inra.fr/mia/T/MDPtoolbox.
24. Hore, S., Shah, A. and Bastian, N.D., 2023. Deep VULMAN: A deep reinforcement learning-enabled cyber vulnerability management framework. Expert Systems with Applications, 221, p.119734. https://doi.org/10.1016/j.eswa.2023.119734.
25. Ganesan, R., Jajodia, S., Shah, A. and Cam, H., 2016. Dynamic scheduling of cybersecurity analysts for minimising risk using reinforcement learning. ACM Transactions on Intelligent Systems and Technology (TIST), 8(1), pp.1-21. https://doi.org/10.1145/2882969.
26. Huang, C., Zhang, Z., Mao, B. and Yao, X., 2022. An overview of artificial intelligence ethics. IEEE Transactions on Artificial Intelligence, 4(4), pp.799-819. https://doi.org/10.1109/TAI.2022.3194503.
27. Puterman, M.L., 1990. Markov decision processes. Handbooks in operations research and management science, 2, pp.331-434. https://doi.org/10.1016/S0927-0507(05)80172-0.
28. Littman, M.L., Dean, T.L. and Kaelbling, L.P., 1995, August. On the complexity of solving Markov decision problems. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (pp. 394-402). https://dl.acm.org/doi/abs/10.5555/2074158.2074203.
29. Lapso, J.A., Peterson, G.L. and Okolica, J.S., 2017. Whitelisting system state in Windows forensic memory visualizations. Digital Investigation, 20, pp.2-15https://doi.org/10.1016/j.diin.2016.12.002
30. Choi, J., Park, J. and Lee, S., 2021. Forensic exploration on Windows File History. Forensic Science International: Digital Investigation, 36, p.301134. https://doi.org/10.1016/j.fsidi.2021.301134
31. Berlin, K., Slater, D. and Saxe, J., 2015, October. Malicious behaviour detection using Windows audit logs. In Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security (pp. 35-44). https://doi.org/10.1145/2808769.2808773.
32. Visoottiviseth, V., Noonkhan, A., Phonpanit, R., Wanichayagosol, P. and Jitpukdebodin, S., 2023, September. AXREL: Automated Extracting Registry and Event Logs for Windows Forensics. In 2023 27th International Computer Science and Engineering Conference (ICSEC) (pp. 74-78). IEEE. https://doi.org/10.1109/ICSEC59635.2023.10329743.
33. Bhatt, S. and Garg, G., 2021, July. Comparative analysis of acquisition methods in digital forensics. In 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT) (pp. 129-134). IEEE. https://doi.org/10.1109/CCICT53244.2021.00035.
34. Studiawan, H., Ahmad, T., Santoso, B.J. and Pratomo, B.A., 2022, October. Forensic timeline analysis of iOS devices. In 2022 International Conference on Engineering and Emerging Technologies (ICEET) (pp. 1-5). IEEE. https://doi.org/10.1109/ICEET56468.2022.10007150.
35. Ionita, M.G. and Patriciu, V.V., 2015, May. Cyber incident response aided by neural networks and visual analytics. In 2015 20th International Conference on Control Systems and Computer Science (pp. 229-233). IEEE. https://doi.org/10.1109/CSCS.2015.41.
36. Hoelz, B.W., Ralha, C.G., Geeverghese, R. and Junior, H.C., 2008. Madik: A collaborative multi-agent toolkit to computer forensics. OTM Confederated International Workshops and Posters, Monterrey, Mexico, November 9-14, 2008. Proceedings (pp. 20-21). https://doi.org/10.1007/978-3-540-88875-8_10.
37. Ghanem, M.C., Chen, T.M., Ferrag, M.A. and Kettouche, M.E., 2023. ESASCF: expertise extraction, generalization and reply framework for optimized automation of network security compliance. IEEE Access. https://doi.or/10.1109/ACCESS.2023.3332834
38. Horsman, G. and Lyle, J.R., 2021. Dataset construction challenges for digital forensics. Forensic Science International: Digital Investigation, 38, p.301264. https://doi.org/10.1016/j.fsidi.2021.301264.
39. Dunsin, D., Ghanem, M.C., Ouazzane, K. and Vassilev, V., 2024. Reinforcement learning for an efficient and effective malware investigation during cyber Incident response. High-Confidence Computing. https://arxiv.org/abs/2408.01999
40. Du, J., Raza, S.H., Ahmad, M., Alam, I., Dar, S.H. and Habib, M.A., 2022. Digital Forensics as Advanced Ransomware Pre-Attack Detection Algorithm for Endpoint Data Protection. Security and Communication Networks, 2022(1), p.1424638. https://doi.org/10.1155/2022/1424638.
41. Moore C., Detecting ransomware with honeypot techniques, Proceedings of the 2016 Cybersecurity and Cyberforensics Conference (CCC), August 2016, Amman, Jordan, IEEE, 77–81, https://doi.org/10.1109/CCC.2016.14.
42. Du, J., Raza, S.H., Ahmad, M., Alam, I., Dar, S.H. and Habib, M.A., 2022. Digital Forensics as Advanced Ransomware Pre-Attack Detection Algorithm for Endpoint Data Protection. Security and Communication Networks, 2022(1), p.1424638. https://doi.org/10.1155/2022/1424638.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2025 doi:10.20944/preprints202501.0416.v1

43. Lapso, J.A., Peterson, G.L. and Okolica, J.S., 2017. Whitelisting system state in Windows forensic memory visualizations. Digital Investigation, 20, pp.2-15. https://doi.org/10.1155/2022/1424638.
44. Karapoola, S., Singh, N., Rebeiro, C., 2022, October. RaDaR: A Real-Word Dataset for AI-powered Run-time Detection of Cyber-Attacks. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management (pp. 3222-3232). https://doi.org/10.1145/3511808.3557121.
45. Zhu, M., Hu, Z. and Liu, P., 2014, November. Reinforcement learning algorithms for adaptive cyber defense against heartbleed. In Proceedings of the first ACM workshop on moving target defense (pp. 51-58).https://dl.acm.org/doi/abs/10.1145/2663474.2663481.
46. Zimmerman, E. 2019. I KAPE - Kroll artefact Parser and Extractor, Cyber Incident Division- Kroll LLC, https://ericzimmerman.github.io/KapeDocs/ (Accessed July 21, 2024).
47. Hore, S., Shah, A. and Bastian, N.D., 2023. Deep VULMAN: A deep reinforcement learning-enabled cyber vulnerability management framework. Expert Systems with Applications, 221, p.119734.https://doi.org/10.1016/j.eswa.2023.119734.
48. Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z. and Zhang, Y., 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. High-Confidence Computing, p.100211. https://doi.org/10.1016/j.hcc.2024.100211.
49. Khalid J. Rozner E. Felter W. Xu C. Rajamani K. Ferreira A. Akella A. Seshan S. Banerjee S(2018) Iron. Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation. https://dl.acm.org/doi/10.5555/3307441.3307468.
50. Guestrin, C., Koller, D., Parr, R. and Venkataraman, S., 2003. Efficient solution algorithms for factored MDPs. Journal of Artificial Intelligence Research, 19, pp.399-468.https://doi.org/10.1613/jair.1000
51. Littman, M.L., Dean, T.L. and Kaelbling, L.P., 1995, August. On the complexity of solving Markov decision problems. In Proceedings of the Eleventh conference on Uncertainty in artificial intelligence (pp. 394-402). https://dl.acm.org/doi/abs/10.5555/2074158.2074203.
52. Sanner, S. and Boutilier, C., 2009. Practical solution techniques for first-order MDPs. Artificial Intelligence, 173(5-6), pp.748-788. https://doi.org/10.1016/j.artint.2008.11.003.
53. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A. and Wojtczak, D., 2020, April. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems (pp. 306-323). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-45190-5_17.
54. Kimball, J., Navarro, D., Froio, H., Cash, A. and Hyde,J. 2022. Magnet Forenisics Inc. CTF 2022 dataset. https://cfreds.nist.gov/all/MagnetForensics/2022WindowsMagnetCTF.(Accessed 08/12/2024)
55. Demir, M. 2024. IGU-CTF24: İstanbul Gelişim Üniversitesi Siber Güvenlik Uygulamaları Kulübü bünyesinde düzenlenen İGÜCTF 24'. https://cfreds.nist.gov/all/GCTF24Forensics. (Accessed 08/12/2024)
56. Givan, R., Dean, T. and Greig, M., 2003. Equivalence notions and model minimization in Markov decision processes. Artificial intelligence, 147(1-2), pp.163-223. https://doi.org/10.1016/S0004-3702(02)00376-4.
57. Patiballa, A.K. 2019. MemLabs Windows Forensics Dataset. https://cfreds.nist.gov/all/AbhiramKumarPatiballa/MemLabs.(Accessed 08/12/2024)
58. Digital Corpora. 2009. NPS-Domexusers 2009 Windows XP Registry and RAM Dataset. https://corp.digitalcorpora.org/corpora/drives/nps-2009-domexusers.
59. Loumachi, F.Y. and Ghanem, M.C., 2024. GenDFIR:Advancing Cyber Incident Timeline Analysis Through Rule Based AI and Large Language Models. arXiv preprint arXiv:2409.02572. https://arxiv.org/abs/2409.02572.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.