Tool for Windows Bypass for Ethical Hackers (Penetration Testing)

1st Rauf Azam
Dept. of Information and Communication Engineering
The Islamia University Of Bahawalpur, Pakistan
azamrauf514@gmail.com

2nd Muhammad Saud Shuraim
Dept. of Information and Communication Engineering
The Islamia University Of Bahawalpur, Pakistan
saudb699@gmail.com

3rd Aoun Muhammad
Faculty Of Engineering
The Islamia University Of Bahawalpur, Pakistan
aoun.muhammad@iub.edu.pk

4th Umar Fayyaz
Faculty Of Engineering
The Islamia University Of Bahawalpur, Pakistan
umar.fayyaz@iub.edu.pk

5th Sehrish Raza
Faculty Of Engineering
The Islamia University Of Bahawalpur, Pakistan
sehrishraza6@gmail.com

***Abstract*—The use of rules has been implemented in windows operating systems today as defensive mechanisms like the Anti-malware. Scan Interface (AMSI) to scan and prevent malicious code execution, especially inside the.NET Common Language Runtime (CLR) [1]. While these protections are fundamental to enterprise protection, they are also introducing. problems of authorized penetration testers who should measured defensive resiliency. A detailed research oriented paper is presented in this paper a modified windows bypass tool, initially analyzed a command line tool, which has been systematically extended to a graphical user interface (GUI) application on organized security tests. The study analysis of tools architectural redesign, simulated persistence mechanisms and its functioning.Contracting within approved red team engagements. Through experiments, we test the degree of GUI based testing affects configuration accuracy, repeatability and defensive insight generation. Our findings indicate that GUI based test tools will greatly cut off, correct and enhance repeatability and facilitate configuration errors for Better assessment of endpoint identification (EDR) capabilities.**

***Index Terms*—Windows Security, AMSI, Penetration Testing, Ethical Hacking, .NET CLR, Endpoint Defense, GUI Based Tools, Persistence Detection, Security Evaluation, Defensive Testing**

A. *I. Introduction*

The field of cybersecurity is in the process of change. as the method of attack becomes more ever refined defensive technologies. Microsoft Win- dows still is the leading desktop operating system. In business settings, which are roughly. Desktop operating market share is 72systems as of 2023 [2]. This popularity contributes to making windows. Cyberattacks are a key target of the systems that require more and more elaborate defensive techniques [3]. Microsoft has introduced response to threats that are changing a tiers based security architecture, which includes windows Real time behavioral monitoring and defender. These systems are aimed to identify and manage a wide variety. of threats by way of several intersecting processes, developing defense in depth security on enterprise. environments. With these improvements, there are advanced enemies. they constantly devise methods of circumventing such pro defensive, and require continuous assessment. Defensively speaking, it is knowledge. these avoidance strategies prior to their weapons by malicious actors is essential towards better detection. capabilities. White hat hackers and penetration testers working on the basis of rigid approval and limited authority [4]. they determine weaknesses of a system simulate real world attacks. tied up so that they are not exploited maliciously. This proactive method towards security testing has been turned into. common in the mature security programs, and with red team activities being key constituents of full scale security testing. In this paper, the analysis and revision are addressed of a windows AMSI bypass tool in the environment of legalized penetration testing. The research addresses a significance between the existing security testing practices: The effect of the tool usability and interface design on quality of assessment and defensive insight generation. The command Line Interface tool was reconfigured into a graphical user inter-systematically. Face (GUI) application to investigate the inter-structured study how well structured.

B. *II. Background and Related Work*

1) ***A. Evolution of Windows Security Architecture****:* Microsoft windows security has developed a lot since early systems of protection were enhanced. The existing security architecture is a complex multi layered strategy integrating detectives and preventative controls [5]. Windows Defender was firstly being launched as a simple anti spyware application in Windows Vista, has developed to become a all-encompassing

DOI: https://doi.org/10.67535/tsp.000002.013
Published under the ISAISS 2026 Proceedings

endpoint protection platform incorpor- based detection rating signature, heuristic analysis. behavioral surveillance and machine learning models. This development is indicative of the general transformation in the industry. a signature only detection to behavioral and anomaly based approaches. The Antimalware Scan Interface (AMSI) introduced in Windows 10, is a specifically notable sign. icant improvement runtime protection. AMSI pro- supplies a standard interface which permits applications. and services to scanner content provided by installed. security products. As opposed to conventional file scanning,

2) ***B. AMSI Architecture and Implementation Details****:* AMSI is a part of the Windows. Communication be- operating system that makes it easy. antimalware products and applications between. content, interface is sent to antimalware registered. providers for inspection. The architecture supports scanning all types of contents such as strings, buffers and streams and therefore, effective. against script based assaults and fileless malware tech- niques. AMSI can be integrated into the.NET ecosystem using oc. curs in the loading process of CLR at certain points.

3) ***C. Bypass Techniques in Academic and Industry Research****:* The interpretation of research findings must be done within several.

1) Specificity of the tools: It is founded on specific findings. tool transformation and it might not be generalizable to all.
2) Population of Operators: it was tested on a limited number of operators. population of operators possessing characteristics. are not necessarily the security professionals.
3) Environment Artificiality: Laboratory environ- ments are not capable of ideal reproduction of production sys- tem complexity and variability.
4) Technology Evolution: Blistering development of the two. attack methodology and defensive technologies. may change the applicability of particular results to time.

4) ***D. The Usability Gap in Security Testing Tools****:* Security testing tools have conventionally been used to focus on functionality rather than usability and this has led to complex command line interfaces with steep learning curves. The study of human computer interaction has indicated that interface design is a major factor of influence on performance of tasks especially in complex areas which demand specific configuration.

5) ***E. Persistence Detection Research****:* Persistence mechanisms are also an urgent focus field in offensive security as well as defensive research. Literature has enumerated many persistence methods on the Windows platform, the Linux platform and on the MacOS platform and in particular, registry edits, scheduled tasks, service installations and fileless persistence strategies.

C. ***III. Methodology: Tool Modification and GUI Implementation***

1) ***A. Research Design and Approach****:* This study utilized a mixed research design of technical implementation, usability test and defensive impact analysis. The work started by taking an open source pre-existing command line AMSI bypass tool that used the simple evasion methods of patching memory and manipulating CLRs. The research design was a three phase structured study:

1) Analysis Phase: Overall scrutiny of the current CLI tool architecture, functionality and constraints.
2) Redesign Phase: Continued reimplementation of the basic functionality in a GUI framework with an increased capacity.
3) Evaluation Phase: Controllable testing of the CLI version in addition to GUI version under various dimensions such as ease of use, precision and defensive insight creation.

2) ***B. Original CLI Tool Analysis****:* The first command line tool used a rather simple AMSI bypass method against AmsiScanBuffer() function by patching memory. The tool was run using a number of command line parameters to manage a variety of functions related to the bypass such as the choice of the targeted process, method of injecting and cleanup tasks. The tool was found to have a number of testing limitations:

1) Poor Feedback Process: Only gave limited output on the success or failure of individual operations.
2) None Management: Could not save or reload testing configurations.
3) None Complete Documentation: External reference that must be used.

3) ***C. GUI Design Principles and Architecture****:* The GUI design was informed by the principles that have been developed in the field of human computer interaction, especially the Nielsen heuristics of user interface design. The design process that was focused on:

1) System Status Visibility: There should always be feedback on the state of the tool and its progress of operation.
2) System-Real World FIT: Terminology and workflows were also consistent with the practices of penetration testing.
3) User Control and Freedom: The control of actions and configuration is easily reversed.
4) Consistency and Standards: Compliance with set conventions of windows interface.

4) ***D. Core Functional Components Implementation****:* The redesigned tool incorporated three primary functional modules exposed through the graphical interface:

a) 1. Embedded Resource Handling Module: This module has made it easy to package testing artifacts into the application executable itself making it easier to deploy and execute in the controlled environment. The implementation made use of the .NET embedded resources that have the on demand extraction features. The module included:

- Embedded resources verification (cryptography).

Fig. 1. Methodology

- Auto-clean up procedures to eliminate extracted artifacts.
- Blending with execution governance controls.

The embedded resource approach offered a familiar issue with penetration testing, which was dealing with testing artifacts in varying environments and keeping them consistent and avoiding excessive external reliance.

b) 2. CLR Version Targeting Module: This module enabled a clear command of the version of the Common Language runtime of .NET which was to be used to execute the test. Various versions of CLR have different security characteristics and integration positions of AMSI hence the choice of version is very important in terms of testing. We can implementation it as:

- Scanning version 4.0 of installed CLR on target machines.
- Hosting interfaces Explicit selection of runtime version.
- Miscellany in the event that the chosen versions were not available.
- Version specific interactions and behavior logged in details.

c) 3. Execution Governance Module: This module gave the ability to have fine grained control on the timing, sequencing and context of the execution of tests. Instead of concentrating on evasion success, this module allowed examining the effect of execution pattern on detection and alerting. Key features included:

- Adjustable time between steps of execution.
- Parent child process relationship control.
- System state monitoring: integration.

*5) **E. Persistence Simulation Implementation**:* Simulation module: The persistence simulation module (named a backdoor in the strictly academic sense) was a model of post exploitation behavior that did not implement the persistence mechanisms. This difference played a vital role in keeping ethical boundaries in place as well as providing valuable defensive assessment. The module implemented:

1) Pattern Based Persistence Modeling: Persistence behaviors are simulated by use of controlled patterns of execution instead of system modifications.
2) Context Variation: Working under varying security contexts to evaluate how it can be detected over varying levels of privilege.
3) Generation of Behavioral Signatures: Generation of scanable patterns devoid of system compromise.

*6) **F. Interface Design and Implementation**:* The graphical interface was developed with Windows Presentation Foundation (WPF) that would offer a modern and responsive interface and would not lose the compatibility with the features of .NET security. The interface was designed as a workflow based design with:

1) Configuration Panel: Nationwide configuration of all testing parameters and validation and preview option.
2) Execution Control Panel: Real time control of the execution and status monitoring and progress indicators of the test execution.
3) Result Visualization panel: table and graphical display of the test results with filtering and export option.

D. IV. Experimental Setup and Ethical Considerations

*1) **A. Testing Environment Configuration**:* Configuration of testing environment: All tests were done in a controlled laboratory setup that was configured to simulate enterprise windows deployments without any connection to production systems. The test environment was composed of:

1) Target Systems Ten Windows 10 and Windows 11 virtual machines at a range of patch versions and security settings.
2) Security Software: There are several endpoint protection platforms such as Microsoft Defender for Endpoint, CrowdStrike Falcon and SentinelOne.
3) Infrastructure monitoring: logs and monitors.
4) Network Design: VPN network segmenting and traffic monitoring of communication channels [6].

*2) **B. Test Scenarios and Evaluation Criteria**:* The experimental design included three primary test scenarios designed to evaluate different aspects of the testing tool and defensive capabilities:

a) Scenario 1: Basic AMSI Bypass Effectiveness: This situation tested the system bypass capability of multiple security products and Windows operating systems. Tests measured:

- Pass rate of AMSI bypass test samples.
- Detection rate of various security products.
- Effects of bypass techniques on operations of a system.

b) Scenario 2: Configuration Accuracy and Repeatability: This was a situation that compared the CLI and GUI version of the tool among different operators of different levels of experience. The criteria of evaluation were:

- The error rates in configuration of various interface types.
- Time to go through standardized testing processes.
- Stability of outcomes in a variety of operators.
- Training of new operators.

c) Scenario 3: Persistence Detection Evaluation: In this case, the persistence simulation module was employed to test defensive attributes against repeat patterns of execution. Tests measured:

- The rates of detection of first implementation and the second implementation.
- Correlation of alert amid various execution occurrences.
- Detection time of various execution patterns.
- Since simulated persistence behavior is investigated, analyst workload is involved.

*3) **C. Ethical Framework and Authorization**:* The whole research activities were conducted according to a strict ethical guideline that was oriented to the absence of possibilities of abuse and unexpected side effects. The significant characteristics of this framework were:

1) Institutional Review: All the research procedures are checked and approved by institutional ethics committee.
2) Containment Protocols: Various degrees of isolation where there is no possibility of escapism of test code to controlled conditions.
3) Authorization Documentation: Official permission of all the testing activities within a particular scope and limitation.
4) Transparency:Full revelation of testing procedures to the stakeholders.
5) Destruction Protocols: Systematic Destruction All artifacts of testing will be destroyed upon completion of a research.

*4) **D. Data Collection and Analysis Methods**:* The quantitative and qualitative data was collected using the Multi modal data collection technique:

1) Performance data of Automated Metric Collection System, data of detection events and tool operation logs are recorded via automated instrumentation.
2) Operation Observations: This is an observation of operator behavior and decision making in the course of the testing.

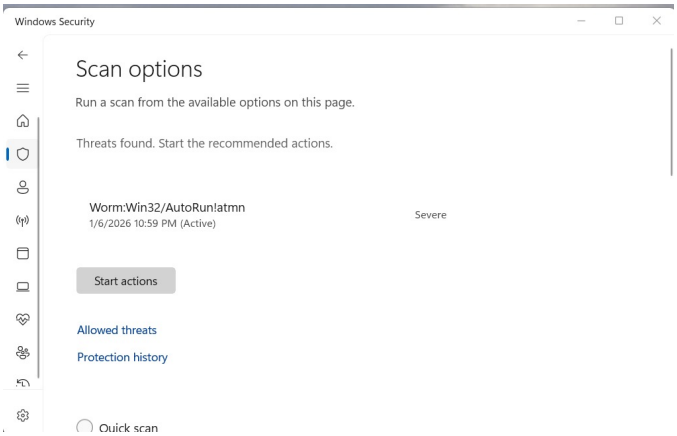

Fig. 2. After Processing in Wrap Builder

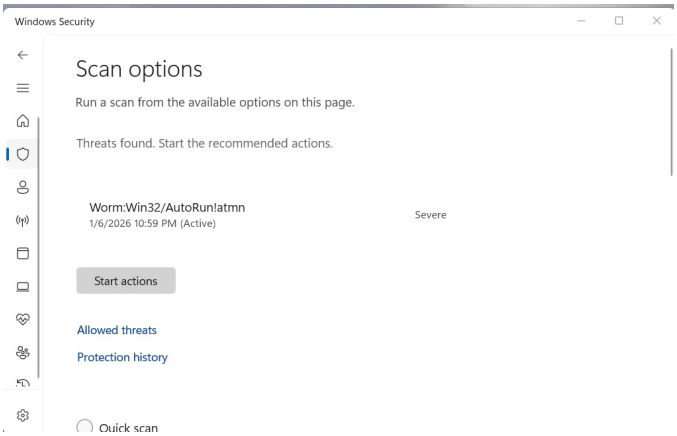

Fig. 3. Before Processing in Wrap Builder

3) Defensive Telemetry: These logs and alerts of security products are collected to be analyzed in order to discern the detection patterns. Post Test Interviews: Interviewing of the operators and defensive analysts in a structured way to incorporate the subjective experience and impressions.

E. V. Results and Analysis

Prior to using the proposed tool, the payload was recognized using Windows Defender as a high-risk threat. Having subjected the same payload to the built-bypass tool, it was not detected any longer in the following scan, which means that it was able to evade signature-detection and runtime-detection systems in the test environment.

F. VI. Discussion

*1) **A. Theoretical Implications**:* The theoretical implications of the research findings on the cybersecurity testing techniques are a number of the following:

1) Usability as a Security Concern: The proven influence of interface design on the quality of testing indicates that the concept of usability is a plausible security consideration as opposed to a convenience factor. The

quality of security assessment is directly linked to the use of tools that minimize configuration errors and lead to a greater degree of consistency.

2) Structured Testing Methodologies: Given that the GUI tool was successful in producing repeatable and consistent results, it is possible to endorse the importance of structured testing methodologies as opposed to ad hoc methodologies. This observation fits larger tendencies on standardization in security testing, but gives concrete evidence of how interface design can be used to execute such standardization.

3) Defensive Insight Generation: The study shows that testing tools can actually be made in such a way that they produce defensive knowledge as opposed to just displaying weaknesses [7]. This is a change of the mindset of proving the possibility of exploitation, to detecting powers.

*2) **B. Practical Implications for Security Teams**:* The conclusions provide a number of realistic implications of security operations:

1) Tool Selection Criteria: The various factors of usability and functional capabilities should be put into consideration by security teams when choosing testing tools [8]. There is a direct influence of interface design on testing quality and efficiency.
2) esigning Testing Process: Organizations need to develop testing processes, which exploit the virtues of structured interfaces and remain flexible to specialized cases that might need CLI tools.
3) Development of Training Programs: Training programs need to include interface specific training instead of believing tool familiarity in one interface paradigm to be familiar in other interface paradigms.
4) Defensive Rule Tuning: The systematic testing strategy allows more fine-tuning of detection rules according to real evasion ability, instead of potential vulnerability.

*3) **C. Limitations and Boundary Conditions**:* The research results need to be viewed under the few critical limitations:

1) Tool Specificity: Results are related to a definite tool transformation, and they might not be applicable to the whole set of security testing tools.
2) Operator Population: There was a small population of operators that was tested whose attributes are not always representative of all security professionals.
3) Environment Artificiality: It is not possible to perfectly recreate production system, complexity and variability in a laboratory environment.
4) Technology Evolution: The fast change in both methods of attack and technologies used in defense can cause the findings to change with time.

*4) **D. Ethical Considerations in Tool Development**:* The study shows that the development of security tools has significant ethical implications:

1) Dual Use Dilemma: These usability improvements that will increase legitimate testing may actually decrease the impediments of malicious use. This tension needs to be taken into consideration when designing and distributing tools.
2) Knowledge Dissemination: The development of knowledge on evasion methods inevitably generates knowledge that may be abused. The ethical model used in this study offers one of the models of responsible dissemination of knowledge.
3) Defensive Primacy: The study indicates that defensive generation of insight can be a major objective of tools, rather than a collateral advantage of the tool.

G. VII. Conclusion and Future Work

*1) **A. Summary of Contributions**:* This study has contributed a number of issues to the field of cybersecurity testing:

1) Empirical Evidence of Usability Effect: Revealed by experiments, the GUI interfaces are significantly better in terms of testing accuracy, consistency and efficiency than the CLI alternatives [9].
2) Persistence Evaluation Methodological Framework: Designed and tested a methodological framework of assessing persistence detection capabilities based on grounded simulation assessment and not necessarily system compromise .
3) Tool Design Principles of Defensive Testing: Principles of design of security testing tools that emphasize the defensive generation of insight as well as the functions.
4) Implementation of Ethical Framework: Showed a practical approach to ethical framework of conducting sensitive security research with proper safeguards and controls.

*2) **B. Future Research Directions**:* Recommendations as to future research have been derived from this work:

1) Adaptive Interface Research: Study of adaptive interfaces which vary in complexity depending. on operator skill, which may be the combination of the adaptability of CLI using GUI.
2) Cooperative Testing Systems: Construction. of experimenting environments, which facilitate co-operation. relationship between blue team and red team by sharing.
3) Cross Platform Testing Tools: Augmentation of the formal testing methodology of other platforms. beyond windows and especially Linux and cloud.
4) Quantitative Security Measures: Construction of measures of testing tool that are pre-defined.

*3) **C. Concluding Remarks**:* The development of cybersecurity threats requires the development of testing tools and methods. This study has shown that interface design has a great influence on the quality of testing and on the generation of defensive insights. We have demonstrated how testing tools can be created to improve, but not simply illustrate, security vulnerabilities because we have turned a CLI based bypass tool into an organized GUI based application.

H. Acknowledgment

The authors acknowledge the cybersecurity research community to continue to debate on ethical testing. This is in recognition of the creators of the original open source testing tools that contributed to this research. We thank our institutional review. Committee on which to seek wise advice on ethical research. practices and the large number of security professionals. who had been involved in testing and evaluation activities. This study was done with the assistance of our repotential institutions computer security research projects.

REFERENCES

[1] A. Rubin, S. Kels, and D. Hendler, "Amsi-based detection of malicious powershell code using contextual embeddings," 2019. [Online]. Available: https://arxiv.org/abs/1905.09538

[2] M. E. Russinovich and D. A. Solomon, *Microsoft Windows Internals: Microsoft Windows Server (TM) 2003, Windows XP, and Windows 2000 (Pro-Developer)*. Microsoft Press, 2004.

[3] Y. Zheng, "Dynamic defenses in cybersecurity: Techniques, methods and challenges," *Digital Communications and Networks*, 2022.

[4] S. Sinha and D. Y. Arora, "Ethical hacking: the story of a white hat hacker," *International Journal of Innovative Research in Computer Science & Technology (IJIRCST), ISSN*, pp. 2347–5552, 2020.

[5] A. A. Kulshrestha, G. Song, and T. Zhu, "The inner workings of windows security," *arXiv preprint arXiv:2312.15150*, 2023.

[6] Bejtlich, *The practice of network security monitoring: Understanding incident detection and response*. No Starch Press, 2013.

[7] P. S. Shinde and S. B. Ardhapurkar, "Cyber security analysis using vulnerability assessment and penetration testing," in *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*. IEEE, 2016, pp. 1–5.

[8] S. Oesch, R. Bridges, J. Smith, J. Beaver, J. Goodall, K. Huffer, C. Miles, and D. Scofield, "An assessment of the usability of machine learning based tools for the security operations center," *arXiv preprint arXiv:2012.09013*, 2020. [Online]. Available: https://arxiv.org/abs/2012.09013

[9] S. Amgothu and G. Kankanala, "Sap cloud installation cli vs gui: Comparative study," *International Journal of Science and Research (IJSR)*, vol. 11, no. 12, pp. 1395–1395, 2022.