

MODEL AND METHOD OF POLICY-DRIVEN BACKDOOR DETECTION ON ENDPOINTS WITH SECURE SIGNATURE DISTRIBUTION AND OPTIMIZED SCAN EXECUTION

I. I. Protsyshyn^{1,a}, I. V. Stopochkina¹

¹ Institute of Physics and Technology

Abstract

This paper proposes a policy-driven model for backdoor detection on endpoints, in which signature-based scanning is combined with scan execution policies and secure rule-set updates. The practical implementation of the approach is oriented toward a prototype that includes an endpoint agent, a trusted local cache of signature sets, scanning profiles, and a YARA-compatible engine. A priority function is proposed that takes into account whether an object belongs to autorun locations, whether it has been modified recently, whether it is located in a privileged area, and whether it lacks a digital signature. Secure signature updates include verification of the signature, hash, and version number of the rule set. The result of the work is a structural diagram of the prototype, a fragment of a YARA rule, and an experimental evaluation scheme focused on comparing full and priority-based scanning.

Keywords: backdoor, endpoint, signature-based detection, scanning policy, YARA, secure signature distribution.

Introduction

Backdoors on endpoints pose a particular danger because they can maintain a hidden presence for a long time, receive remote control, and restore activity after system restart. In MITRE ATT&CK terms, such tools are associated in particular with the persistence tactic, where an adversary attempts to maintain access despite rebooting, credential changes, or partial system recovery [1]. In practice, signature-based mechanisms remain widely used to search for known malicious artifacts and malware families. The YARA documentation explicitly defines it as a tool for identifying and classifying samples by textual and binary patterns [2]. At the same time, for host-based detection tools, timely signature updates and correct agent management remain important because signature updates add or refine detection capabilities [3].

Despite the maturity of signature-based approaches, their use for detecting backdoors on endpoints has several typical limitations. First, treating all scanned objects equally increases scan time and system load. Second, compromise of the update channel or the local signature cache creates a risk of rule substitution or rollback to an outdated set. Third, the performance of the scanning mechanism itself strongly depends on the structure of rules and on the processing order of scanned objects; for YARA, this is confirmed by studies devoted to the impact of regular expressions, atoms, and the internal pattern-matching mechanism on performance [4, 5]. Therefore, it is reasonable to move

from «full» signature scanning to a policy-driven model in which scanning first focuses on objects with the highest probability of malicious persistence or execution.

This paper considers the endpoint file system and related autorun points as the main sources of artifacts for signature-based monitoring. Such a formulation corresponds to typical backdoor persistence mechanisms and makes it possible to study the interaction among scanning policies, signature rules, and priority-based object processing within a unified approach. The object of research is the process of detecting and localizing backdoors on endpoints. The subject of research is scanning policies, secure signature updates, and methods of priority-based object verification. The aim of the work is to improve the efficiency and security of backdoor detection by means of a policy model, verification of trusted signature sets, and prioritization of scanned objects.

1. Problem Statement and Model

Let an endpoint at time t be described by the set of monitored objects

$$O(t) = \{o_1, o_2, \dots, o_n\}, \quad (1)$$

where o_i is a file of an arbitrary type or an autorun element that is subject to verification for signature-based indicators of a backdoor. Thus, the model is not limited to executable files only: scanned objects may include archives, documents, scripts, libraries, temporary files, and other file-system artifacts.

^atoken0078@gmail.com

In this work, the scanning policy is defined by one of three profiles: fast, standard, or full. The profile selection is determined by the operating mode of the endpoint and the object type. The scanning context is formed from factors that affect object risk, in particular the presence of recent changes, autorun properties, and privileged placement.

To order object verification, a linear risk score is used:

$$r(o_i) = w_1 p_i + w_2 m_i + w_3 a_i + w_4 u_i, \quad (2)$$

where p_i reflects whether the object belongs to autorun locations, m_i indicates recent modification, a_i indicates placement in a privileged area or execution with elevated privileges, and u_i denotes the absence of a digital signature or trusted origin. The weight coefficients w_j are defined by policy. Such a function makes it possible to scan first the objects with the highest probability of malicious persistence.

The scan plan is defined by the queue

$$Q = \text{sort}(O, r, \text{desc}), \quad (3)$$

where objects with larger values of $r(o_i)$ are scanned earlier. This formalization provides ordered processing of objects in accordance with their risk profile.

2. Method of Secure Signature Updates

Signature-based detection requires continuous rule updates. However, the signature delivery channel itself is a critical security element: substitution or rollback of a rule set directly affects the ability to detect malicious artifacts. In ClamAV, the importance of the standard process for managing signature databases and preserving their integrity during updates is emphasized separately [6, 7]. In this paper, the signature set is considered a managed update object.

In this work, a signature set is understood as a collection of YARA rules distributed as a single logical update together with a version number, a cryptographic hash, and a digital signature. During an update, the agent performs three checks: 1) the signature must be valid; 2) the hash of the locally obtained content must match the reference value; 3) the version number of the new set must not be lower than the version number of the latest trusted set. If at least one of these checks fails, the update is rejected and the system continues to operate with the previous valid rule set.

The proposed mechanism ensures that the endpoint agent uses only valid and up-to-date rule sets. For the experimental study, a rule set focused on persistence indicators and typical execution strings is used. A fragment of such a rule is shown in Listing 1.

3. Algorithm of Optimized Scan Execution

The key idea of the algorithm is priority-based ordering of objects according to formula (2) and the selection of one of three scanning profiles. YARA documentation

Table 1. Fragment of a YARA rule for detecting suspicious autorun activity

```
rule Suspicious_RunKey_PowerShell
{
  strings:
    $rk = "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
    $ps = "powershell -windowstyle hidden"
  condition:
    all of them
}
```

and practical performance recommendations show that the cost of scanning depends not only on the number of files but also on the structure of the rules themselves [2, 4, 5]. Therefore, even a simple reduction in the number of full scans for low-risk objects may already provide a performance benefit.

After loading a valid signature set, the agent inventories the objects $O(t)$, calculates $r(o_i)$, and builds the queue Q . For high-priority objects, the full rule profile is applied; for medium-priority objects, the standard profile is used; for low-priority objects, a reduced scan is performed or the object is postponed to the next cycle. If a rule matches, the agent records the event in the log and moves the object to deeper analysis or local quarantine.

The high-level cycle of the endpoint agent is shown in Fig. 1. It reflects the sequence of the main processing stages: verification of the rule set, queue construction, object scanning, and response to detected matches.

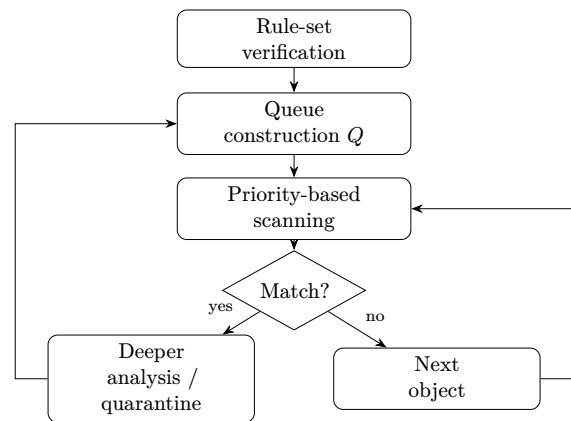


Figure 1. High-level operating cycle of the endpoint agent

4. Prototype and Experimental Evaluation Scheme

The interaction of the prototype components is shown in Fig. 2. The prototype consists of a repository of signed signature sets, a policy file, an endpoint agent, a trusted local cache, and a YARA-compatible engine. The signature repository provides rules and metadata, while the endpoint agent verifies them, selects the scanning profile, and initiates the scanning of objects.

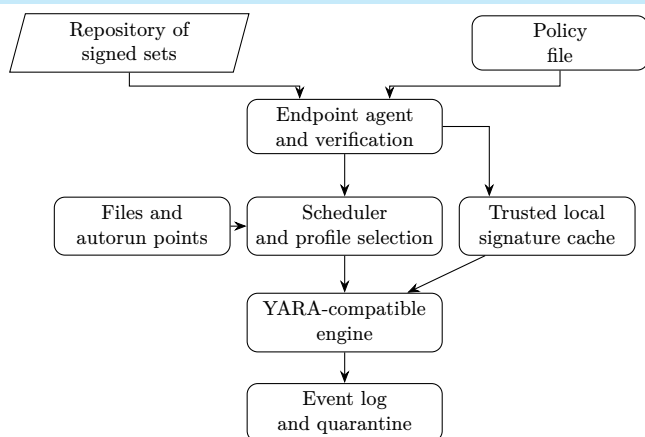


Figure 2. Interaction diagram of the prototype components

The experimental evaluation includes two parts. The first is functional: acceptance of a valid signature set and rejection of a corrupted or outdated set. The second is performance-oriented: comparison of full and priority-based scanning by total execution time, average CPU usage, and the number of detected test samples. Such a scheme makes it possible to evaluate both the correctness of the rule-update mechanism and the effect of priority-based object ordering.

An illustrative dependence of relative time to scan start on the object type is shown in Fig. 3. It reflects the expected practical effect of the approach: autorun objects and recently modified files are scanned earlier than ordinary user documents or temporary directories.

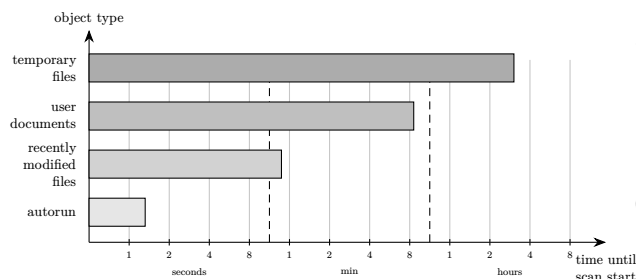


Figure 3. Illustrative timeline of priority-based scan start for different object types

Conclusions

The theses propose a policy-driven model of back-door detection on endpoints that combines verification of signed signature sets, priority-based ordering of objects, and the use of multiple scanning profiles. The practical value of the approach lies in reducing the overhead of full signature-based monitoring and increasing trust in the rule-update process. The proposed prototype provides a basis for experimental verification of both the correctness of the signature-update mechanism and the efficiency of priority-based scanning. Further development of the approach is associated with expanding the rule set, adding context sources, and extending endpoint application scenarios.

References

1. *MITRE ATT&CK*. Persistence, Tactic TA0003 - Enterprise. — 2018. — URL: <https://attack.mitre.org/tactics/TA0003/> (visited on 03/31/2026).
2. *VirusTotal, YARA Project*. Welcome to YARA's documentation. — 2025. — URL: <https://yara.readthedocs.io/> (visited on 03/31/2026).
3. *Scarfone K., Mell P.* Guide to Intrusion Detection and Prevention Systems (IDPS) : NIST Special Publication 800-94 / National Institute of Standards ; Technology. — 2007. — URL: <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-94.pdf> (visited on 03/31/2026).
4. *Regéciová D., Kolář D., Milkovič M.* Pattern Matching in YARA: Improved Aho-Corasick Algorithm // IEEE Access. — 2021. — Vol. 9. — P. 62857–62866. — DOI: [10.1109/ACCESS.2021.3074801](https://doi.org/10.1109/ACCESS.2021.3074801).
5. *Regéciová D.* Yara: Down the Rabbit Hole Without Slowing Down // Journal on Cybercrime & Digital Investigations. — 2022. — Vol. 7, no. 1. — URL: <https://cyberjournal.cecyl.fr/index.php/cybin/article/view/35> (visited on 03/31/2026).
6. *ClamAV Team*. Updating Signature Databases. — 2025. — URL: <https://docs.clamav.net/manual/Usage/SignatureManagement.html> (visited on 03/31/2026).
7. *ClamAV Team*. Signatures - ClamAV Documentation. — 2025. — URL: <https://docs.clamav.net/manual/Signatures.html> (visited on 03/31/2026).