LinkGuard: A Lightweight State-Aware Runtime Guard Against Link Following Attacks in Windows File System

Bocheng Xiang, Yuan Zhang, Hao Huang, Fengyu Liu, Youkun Shi

Fudan University

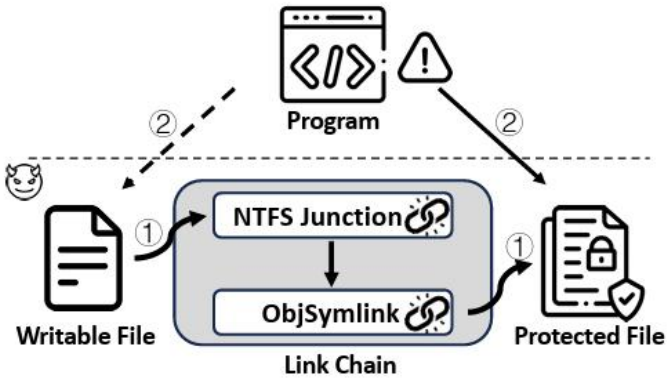
Background

- Large and Persistent Attack Surface
  - As of August 2025, 1,000+ CVEs have been assigned to Link Following attacks on Windows.
- Gap in Existing Defenses
  - Most prior LF defenses focus on Unix-like systems, with **no dedicated protection** designed for Windows.
    - 1） *Poor Portability*: Existing solutions rely on deep Linux kernel integration which **do not transfer to Windows**[1] .
    - 2） Limited Scope: These works **only track individual file operations**, and thus fail to detect multi-step Link Following attacks[2].

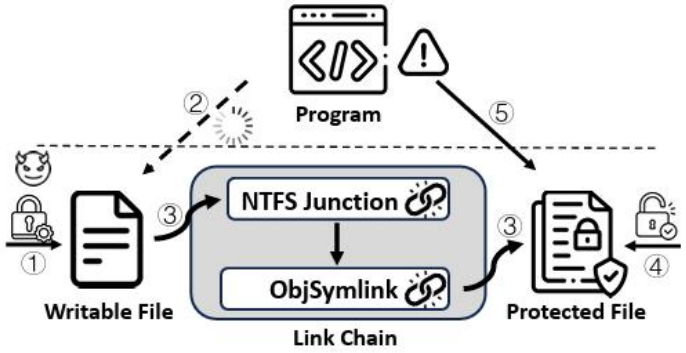
[1] Watson R N M, Anderson J, Laurie B, et al. Capsicum: Practical capabilities for {UNIX}[C]//19th USENIX Security Symposium (USENIX Security 10). 2010.
[2] S. Chari, S. Halevi, and W. Z. Venema, “Where do you want to go today? escalating privileges by pathname manipulation.” in NDSS. Citeseer, 2010.

Types of Link Following Attack

LF attacks can be categorized into **single-step** and **multi-step** types.



(a) Overview of a single-step Link Following Attack



(b) Overview of a multi-step Link Following Attack

Fig. 1: Two Types of Link Following Attacks

Single-step LF attack

① Attackers prepares a link chain from a writable file to a protected target.

② Victim privileged program performs a single file operation that implicitly follows the link chain, redirecting access to a protected file.

Multi-step LF attack

① Attackers place an oplock on the writable file.

② Any file operations on the locked file are suspended.

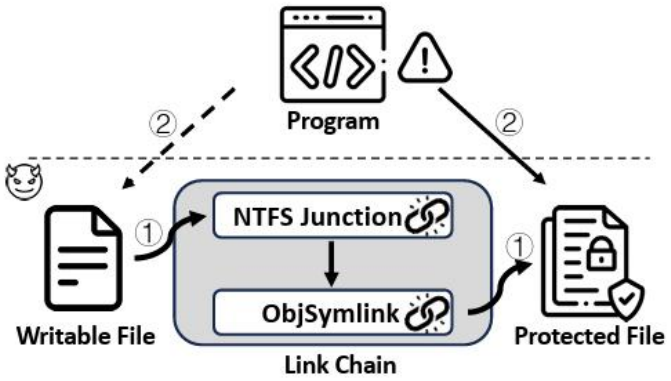
③ Attackers prepares the link chain pointing to the protected target.

④ Attackers actively release oplock after completing all necessary attack preparations.

⑤ Previously suspended file operation resumes and follow the link.

Types of Link Following Attack

LF attacks can be categorized into **single-step** and **multi-step** types.



(a) Overview of a single-step Link Following Attack

Single-step LF attack

① Attackers prepares a link chain from a writable file to a protected target.

② Victim privileged program performs a single file operation that implicitly follows the link chain, redirecting access to a protected file.

What is a Link Chain?

A link chain is a combination of non-privileged Windows indirection mechanisms (e.g., directory junctions[3] and Object Manager symlinks[4]) that together provide linking power equivalent to privileged NTFS symbolic links, enabling file accesses to be transparently redirected across protection boundaries.

[3] https://learn.microsoft.com/en-us/windows/win32/fileio/hard-links-and-junctions
[4] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-object-manager

Empirical Study: LF Attacks and Real-World Defenses

- A large-scale empirical study of real-world Link-Following attacks and their deployed countermeasures on Windows.
- Goal: 1) **Characterize** how developers defend against LF attacks in practice

  2) Identify the **fundamental limitations** of these defenses

- **Research Questions**

  I. What countermeasures are most commonly adopted by developers?

  II. What weaknesses do these countermeasures exhibit in practice?

  III. What key insights can be drawn to guide future defenses?

Empirical Study — RQ1

- We conduct a comprehensive analysis of the countermeasures adopted for 152 CVE-assigned LFVulns.
- **Q1:** What countermeasures are most commonly adopted by developers?
  - C1, File ACL Hardening (67/152, **44.08**%): restricts access to sensitive files by **enforcing strict ACLs**, typically removing write permissions for regular users.
  - C2, Secure Path Binding (12/152, **7.89**%): binds temporary or user-specific directories to **trusted file paths** (e.g., via environment variables such as *%SystemRoot%*), ensuring that file operations are confined to controlled locations.
  - C3, Redirection Guard (18/152, **11.84**%): restricts link following to **administrator-created symbolic links only**, effectively blocking untrusted link chains.
  - C4, File Name Randomization (8/152, **5.26**%): prevents LF attacks by **randomizing writable file names** (e.g., UUIDs or random strings), making paths unpredictable.
  - C5, Program Least Privilege (15/152, **9.87**%): **reduces unnecessary privileges**, typically via *Windows impersonation*[5] or by explicitly dropping elevated privilege.
  - C6, File Path Validation (39/152, **25.66**%): **validating file paths before use,** typically by detecting symbolic links or verifying that the resolved final path matches the intended one,.

[5] https://learn.microsoft.com/en-us/windows/win32/com/impersonation

Empirical Study — RQ2

- Q2: **What weaknesses do these countermeasures exhibit in practice?**
- High Modeling and Engineering Overhead
  - Many defenses require exhaustive identification and instrumentation of security-sensitive file operations or resources. (C1,C4,C6)
- Limited Compatibility and Applicability
  - Several countermeasures apply only to specific programs, paths, or execution contexts, and thus fail to provide comprehensive protection across diverse Windows applications. (C2,C5)
- Incomplete Protection
  - Existing defenses can still be bypassed under realistic conditions, especially by attacker-controlled paths. (C3)

Empirical Study — RQ3

- Q3: **Key Observations from Real-World LF Attacks**
- Observation 1: **Cross-Subject** LF Signature
  - Every LF attack consists of two indispensable operations:
    - 1) Link chain construction by the attacker
    - 2) Link-following file operations by a victim program.
  - These operations are **inherently cross-subject**, jointly forming the defining signature of LF attacks.
- Observation 2: State-Driven Attack Semantics
  - Although LF attacks appear complex and unordered, they follow well-defined file-operation states and transitions, which means can **be characterized by specific state transition patterns**

System Design — LinkGuard

- Design Goals
  - G1: **Practicality**. LinkGuard aims to provide accurate LF attack detection with low runtime overhead and latency, while maintaining high compatibility.
  - G2: **Effectiveness and Extensibility**. LinkGuard is designed to mitigate both single-step and multi-step LF attacks, and to easily incorporate new attack patterns
- Insights
  - Effective detection does not require system-wide monitoring but only **tracking link-related operations across different subjects**. (Observation 1)
  - LF attacks follow well-defined file-operation state transitions. This enables detection via lightweight FSMs that monitor whether cross-subject operations reach attackable terminal states. (Observation 2)

System Design — LinkGuard

- **Overview**

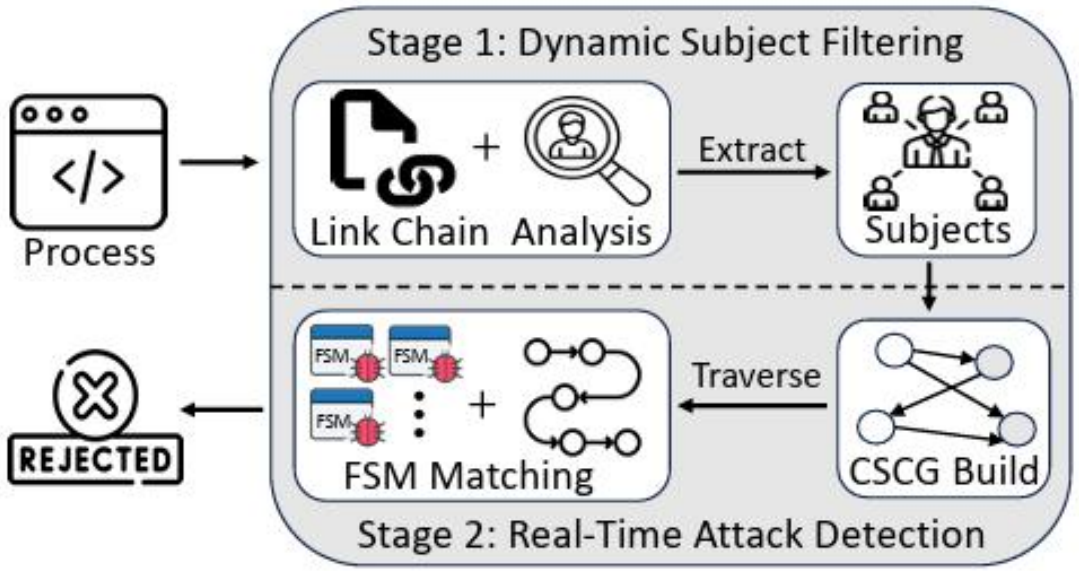


Fig. 3: Design Overview of *LinkGuard*

- **Stage I: Dynamic Subject Filtering**
  - LinkGuard selectively monitor and record file operations related to link chain creation and following at runtime
- **Stage II: Real-Time Attack Detection**
  - LinkGuard correlates cross-subject file operations into a CSCG(Cross-Subject Chain Graph) and continuously evaluates their state transitions with parallel FSMs, enabling timely interception of single-step and multi-step LF attacks

Stage I: Dynamic Subject Filtering

- **Dynamic Subject Filtering**
  - Step 1: Subject Definition

$$\text{Sub}(op) = \begin{cases} \text{Token}(T) & \text{if } op \text{ succeeds} \\ \text{Group}(f, p) & \text{if } op \text{ fails during execution} \end{cases} \quad (1)$$

- **T: Execution Thread**
- **f: Files**
- **p: Permissions**

  - Step 2: Monitor and Extract

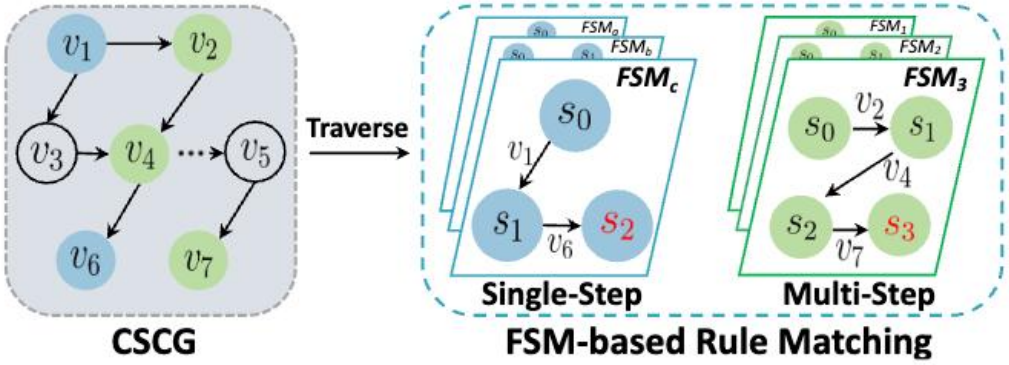
Primary Operation	Corresponding IRP requests/flags	link chain construction	link chain following
Create/Open	`IRP_MJ_CREATE` [50]	✓	✓
Write	`IRP_MJ_WRITE` [51]	✗	✓
Rename	`FileRenameInformation` [52]	✗	✓
Delete	`IRP_MJ_SET_INFORMATION` [53], `FILE_FLAG_DELETE_ON_CLOSE` [54]	✗	✓
Set DACL	`IRP_MJ_SET_SECURITY` [55]	✗	✓
Symbolic Link Create	`IRP_MJ_FILE_SYSTEM_CONTROL` [56]	✓	✗
Lock	`FSCTL_REQUEST_OPLOCK` [57]	✓	✓

✓/✗: involved / not involved

Table: Monitored file operations and corresponding IRP requests in link chain construction and following

Stage II: Real-Time Attack Detection

- **Real-Time Attack Detection**
  - CSCG(Cross-Subject Chain Graph) Definition
    - The CSCG is a **directed graph** that models temporally ordered, file-dependent link-related operations **across different subjects**
  - FSM-based Rule Matching
    - LinkGuard conducts parallel FSM matching over the CSCG to detect coexisting single-step and multi-step LF attacks



The path P1 = {v1, v3, v4, v6} corresponds to a single-step attack,
while the path P2 = {v1, v2, v4, . . . , v5, v7} spans multiple operations and ultimately triggers a multi-step attack.

Evaluation

- Effectiveness Evaluation
  - **LinkGuard** successfully mitigates **68** of **70** (**97.1**%) real-world vulnerabilities
    - including all single-step attacks (26/26) and **95.45**% multi-step attacks (42/44)
  - The average and P95 runtime latency increased **~30** ms when LinkGuard was deployed to mitigate LF attacks

TABLE III: ***LinkGuard*'s effectiveness for mitigating real-world LF attacks in dataset-α**

Operation Category	Impact	Attack Type	
		S	M
File Creation and Overwriting	Denial of Service	15 / 15	4 / 4
File Move and Copy	Information Disclosure	1 / 1	8 / 8
File Permission Assignment	Information Disclosure Privilege Escalation	7 / 7	2 / 2
File Deletion	Denial of Service Privilege Escalation	3 / 3	28 / 30
Overall	–	26 / 26	42 / 44
Proportion	–	100%	95.45%

S: Single-step attacks. M: Multi-step attacks.

Fig. 8: Runtime latency under defense scenarios (dataset-α) and benign deployment in real-world applications (dataset-γ).

Evaluation

- Performance Evaluation
  - **LinkGuard** increases the execution time by an average of **10.8**% in 6 standard file operations
  - On average, **LinkGuard** introduces **1**% overhead in microbenchmarks and **3.4**% overhead in real-world application workloads

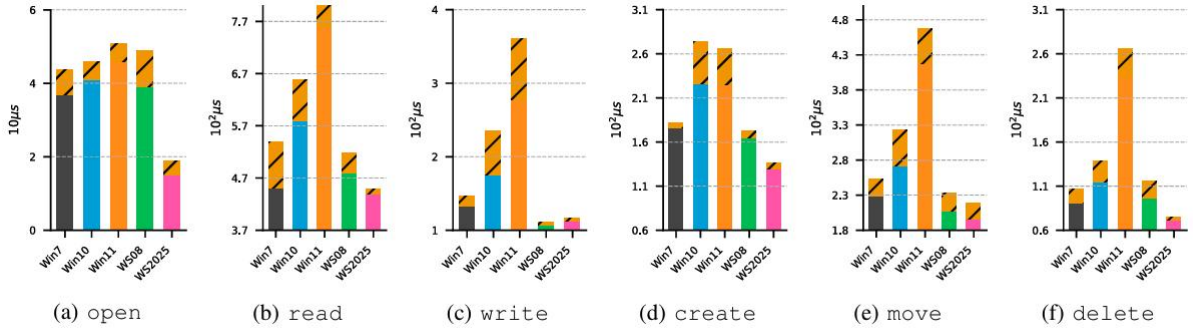


Fig. 5: Execution time of standard file operations across five Windows versions. Std.Dev. below 8.0%

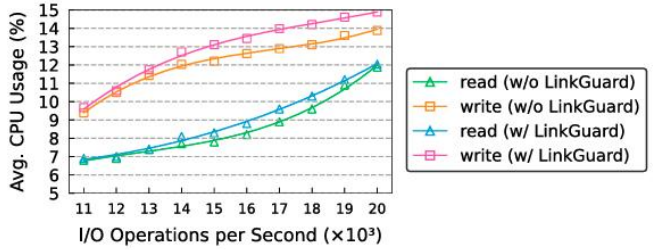


Fig. 6: Impact of *LinkGuard* on average CPU usage under varying IOPS for read and write. Std.Dev. below 0.5%

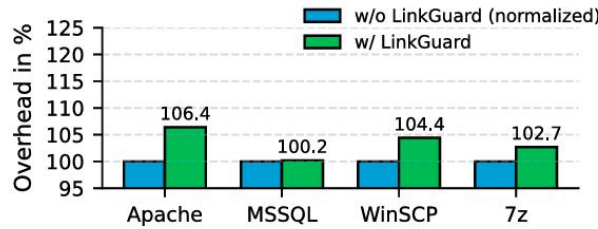


Fig. 7: Normalized overhead of different Windows common applications. (Avg. Increase: ~3.4%, Std. Dev.: < 2.4%)

microbenchmarks

application workload

Conclusion

- We conduct the first systematic empirical study of **6** practical countermeasures against LF attacks.
- Guided by key observations, we propose LinkGuard, **a lightweight state-aware runtime guard against LF attacks** in the Windows file system.
- LinkGuard mitigates all single-step attacks and **95.45**% of multi-step attacks, only introduces an average of **1**% overhead in microbenchmarks and **3.4**% overhead in real-world application workloads.

Thanks for listening to our presentation!

For more details, welcome to follow our paper.

LinkGuard: A Lightweight State-Aware Runtime Guard Against Link Following Attacks in Windows File System

Bocheng Xiang, Yuan Zhang, Hao Huang, Fengyu Liu, Youkun Shi
Fudan University, China
{*bcxiang23, huangh21, fengyuliu23, 21110240048*}*@m.fudan.edu.cn, yuanxzhang@fudan.edu.cn*

Any Question:

bcxiang23@m.fudan.edu.cn