Thinking of "Attack Primitive" Based on Dirty Frag Vulnerability Discovery Process -- Re-discussion on Human-machine Division of Vulnerability Discovery and Analysis

The original report is in Chinese, and this version is an AI-translated edition.

Based on the analysis and mining of relevant information that Theori researcher Taeyang Lee discovered and exposed Copy Fail(CVE-2026-31431) with the help of Xint Code platform, Antiy CERT wrote an article entitled "Analysis of Copy Fail Discovery Process, a New Paradigm for Vulnerability Analysis of Human-AI Collaboration". The article points out that in the process of accelerating the study of kernel vulnerabilities from the "craftsman era" to the "man-machine collaboration era", the working paradigm is being reconstructed-"people focus on 'asking the right questions' and AI are responsible for 'asking all relevant questions '". Some netizens asked: How do you understand "asking the right question"? We rely on the Dirty Frag vulnerability disclosed by Hyunwoo Kim(@ v4bel), a former researcher at Theori, to follow up and sort out the relevant details to improve the understanding of "asking the right question", that is, a new generation of security researchers are using "attack primitives" as an anchor to reconstruct the paradigm framework for vulnerability discovery and analysis.

1. Introduction

1.1 Event Background: Technical Summary and Social Disclosure Impact of Dirty Frag Vulnerability

In May 2026, independent security researcher Hyunwoo Kim(@ v4bel) disclosed a local privilege escalation (LPE) vulnerability-Dirty Frag (CVE-2026-43284,CVE-2026-43500)-that affects a full range of major Linux distributions.

The technical essence of Dirty Frag is not a single vulnerability, but a chained combination of two Page Cache write primitives: xfrm-ESP Page-Cache Write (introduced in January 2017) rooted in the IPsec/xfrm subsystem and RxRPC Page-Cache Write (introduced in June 2023) rooted in the RxRPC subsystem. There are environmental restrictions on the separate use of both, but Hyunwoo Kim has realized stable rights promotion across distributions and configurations through chain combination, covering mainstream systems such as Ubuntu 24.04.4, RHEL 10.1, openSUSE Tumbleweed, CentOS Stream 10, AlmaLinux 10, Fedora 44, etc.

The disclosure of Dirty Frag appeared within a very short time after Copy Fail(CVE-2026-31431) was exposed. The two share the same attack philosophy-page cache controlled write (Page-Cache Write Primitive). However, Dirty Frag promotes the versatility and reliability of utilization to a new level by complementing the primitives across subsystems. This incident is not only a technical vulnerability exposure, but also a profound challenge to the current vulnerability response mechanism, responsible disclosure process and human-machine collaborative mode of security research.

1.2 Questions: Cognitive Upgrading from ″Crash Point Discovery″ to ″Original Language Abstraction (Primitive)″

Traditional vulnerability research often starts with "finding the crash point" and ends with "obtaining the CVE number". Researchers focus on the trigger conditions, crash site and patch differences. However, Dirty Frag's discovery process reveals a higher-order research paradigm: the discoverer Hyunwoo Kim is not looking for "another buffer overflow" or "another UAF", but is systematically hunting for a scarce capability-page cache write primitives. This paradigm shift requires researchers to move from "phenomenal description" to "capability abstraction", from focusing on "which line of code is wrong" to focusing on "what atomic capabilities this code will give to the user".

An attack primitive (Attack Primitive) is a core engineering concept in the study of vulnerability exploitation, which refers to the basic operation capability that an attacker obtains through a vulnerability or mechanism and can be stably reused. It is not a complete

exploit, but an "atomic operation" that constitutes a exploit ". Understanding attack primitives is a key watershed that distinguishes "vulnerability analysis researcher" from "vulnerability exploitation engineer.

1.3 research objective: to reconstruct the paradigm framework of vulnerability discovery and analysis with attack primitives as anchor points

The purpose of this report is to analyze the basic framework and classification system of attack primitives, deconstruct the double primitive chain structure of Dirty Frag, examine the methodological path from subsystem audit to "primitive hunting" of Hyunwoo Kim, and re-discuss the human-computer division of labor in vulnerability discovery and analysis on this basis. In the end, we will try to answer: in the context of the explosion of AI-assisted vulnerability mining capabilities, what are the core values of human researchers (i.e., how to ask the right questions)? How should tool chains evolve? How should defense systems be transformed?

1.4 Report Scope and Terminology Agreement

This report focuses on the Linux kernel local empowerment scenario. The core terms are defined as follows:

• Attack primitives: basic attack capability units that can be stably reused, such as page cache writing, arbitrary address reading, control flow hijacking, etc.

• Page-Cache Write Primitive: The ability to write controllable data to the page cache page in memory without touching the disk file.

• Chaining: Multiple independent attack primitives are stitched together in a logical order to bypass their respective environmental constraints and build a complete kill chain.

• Dirty family: refers to the vulnerability pedigree of shared page cache tampering paradigms such as Dirty Pipe(CVE-2022-0847), Copy Fail(CVE-2026-31431), and Dirty Frag.

2. The theoretical framework and classification system of attack primitives.

2.1 Definition of Concepts: Semantic Migration from Cryptographic Prima-tuo to Exploit Prima-tuo

In cryptography, "primitives" refer to basic algorithms such as hashing, symmetric encryption, and signatures-they do not solve specific business problems by themselves, but can be combined to build complex protocols such as TLS, VPN, and blockchain. The attack primitive borrows this concept: it is the "smallest functional unit" that constitutes an exploit ". A single primitive is often not enough to complete the effect, but it is the basis for supporting the complete kill chain.

The semantic migration from cryptographic primitives to attack primitives reflects the paradigm expansion of security research from "defensive abstraction" to "offensive abstraction. Cryptographic primitives answer "what can I do safely" and attack primitives answer "what can I do with the system". Both follow the three principles of Composability, Determinism and Reusability.

Original Quadruple of 2.2 Kernel Local Priming (LPE) Scenario: Read, Write, Executive, Degradation

In the study of Linux kernel native empowerment, attack primitives can be divided into four categories according to the type of capability:

Type	Capability Descripti	Typical sources

	on	
Read Primitive	Read kernel/user state arbitrary address data	Information leakage, side channel, UAF read
Write primitives (Write Primitive)	Write controllable data to any address in kernel/user mode	Page cache tampering, OOB write, UAF write
Execution primitive (Execute Primitive)	Hijacking control flow (modifying function pointers, return addresses)	ROP, JOP, Kernel UAF
Degradation primitive (Downgrade Primitive)	Bypass security perimeter (disable COW, bypass SELinux)	Logic defects, configuration tampering, reference count manipulation

Dirty Frag and Copy Fail provide the most scarce of the write primitives-page cache controlled write primitives. The scarcity of such primitives is that it allows an attacker to modify the page-cached copy of the setuid binary in memory without touching the disk, thereby bypassing the file integrity monitoring and signature verification mechanisms.

2.3 Special Status and Scarcity of Page-Cache Write Primitive

Page Cache is the core mechanism of the Linux kernel for caching file system data. When a user-state program reads a file, the kernel maps the disk block to the memory page cache page, and the program actually reads the page cache copy. Normally, the page cache page is read-only to the user-mode process that maps it; any write should trigger a COW(Copy-On-Write), with the kernel allocating a new page and copying the data.

The essence of the page cache write primitive is to find a kernel code path that writes attacker-controlled data to the page cache page if it is mistakenly believed that it can be safely modified in situ. The particularity of this primitive lies in:

1. No file write permission requirement: the attacker does not need to have write permission to the target file, or even the directory where it is located.

2. Bypass integrity monitoring: Because only the page cache in memory is modified and disk write-back is not triggered, file system-level integrity checks (such as IMA/EVM) cannot be sensed.

3. High certainty: page cache writes caused by logic vulnerabilities do not rely on race conditions, with a success rate of nearly 100 percent.

4. Cross-version stability: The page cache mechanism is the core architecture of the kernel, and the utilization method is highly stable between different kernel versions.

Because of the above characteristics, page cache write primitives become the "Lord of the Rings" level capability in local empowerment research. For the first time, Dirty Pipe proves that the primitive is feasible, Copy Fail proves that it can be reproduced, and Dirty Frag proves that it can be engineered and generalized.

2.4 Attack Primtives and Classical Security Models: Complementary Perspectives of Kill Chain, ATT & CK, STRIDE

Attack primitive thinking can form a complementary mapping with the classical security model:

• Kill Chain: The attack primitive corresponds to the "weaponization" stage in the kill chain. The page cache write primitive is a generic weaponized component that adapts to different initial access vectors.

• ATT&CK Framework: Page cache write primitives map to T1068 (Exploitation for Privilege Escalation), but reveal the "exploit mechanism" rather than just the "exploit behavior" at a finer granularity ".

STRIDE: From a threat modeling perspective, page-cache write primitives take advantage of the instantiation of Tampering threat-but the tamper object is promoted from a traditional "data" to an "in-memory image of the execution body".

Incorporating attack primitives into the classic model helps the defender reconstruct the defense system from the dimension of "capability elimination" rather than just "vulnerability repair.

3.Dirty Frag's attack primitive deconstruction

3.1 The essence of technology: not a single vulnerability, but a double-primitive chain combination structure.

The technical nature of Dirty Frag must be understood precisely: it is not a defect introduced by a single commit, but rather a chained combination of two independent page cache write primitives. Hyunwoo Kim explicitly defined it in the disclosure as "the chain exploitation of two Page-Cache Write primitive", this self-positioning reveals the engineering height of modern vulnerability research-researchers no longer seek only to "find a perfect vulnerability", but to pursue "a set of complementary primitives".

3.2 xfrm-ESP write primitives: capability boundaries and namespace constraints for 4-byte

controlled writes.

xfrm-ESP write primitives are rooted in the IPsec/ESP subsystem. In January 2017, Steffen Klassert submitted esp4/6: Avoid skb_cow_data whenever possible to optimize ESP data path performance, which is intended to reduce unnecessary COW overhead. This optimization uses skb_page_frag_refill to directly allocate page fragments in the ESP output path, but when externally shared page cache pages are injected into skb's frag through splice(), the ESP decryption path writes to these shared pages in place.

Specifically, the attacker uses splice() to map the page cache page of the setuid binary (such as/usr/bin/su) to the pipe, which is then fed into the IPsec stack as part of the ESP packet. The esp_input() series of functions in the ESP decryption path write a 4-byte seq_hi field to the tail of the skb via skb_put() or pskb_put() when decrypting the tail of the ESP -- and this write occurs directly on the page cache page injected by the attacker. Since the value of seq_hi can be directly controlled by an attacker by constructing an ESP packet, a 4-byte fully controllable write is achieved.

However, the xfrm-ESP primitive has a key limitation: it requires the user to create a network namespace. In distributions such as Ubuntu that are enabled by default and strictly configured for AppArmor, the ability of unprivileged users to create user namespaces is limited through policies such as apparmor_restrict_unprivileged_userns, making the primitive unexploitable alone.

3.3 RxRPC write primitive: 8-byte write unprivileged path and fcrypt key brute force mechanism

The RxRPC write primitive is Dirty Frag's "second key", and its core value is that it doesn't require any privileges at all. RxRPC(AF_RXRPC) is a remote procedure call protocol for AFS(Andrew File System) and Kerberos authentication in the Linux kernel. In the Kerberos authentication path rxkad_verify_packet_1() of RxRPC, the first 8 bytes of the received packet will perform an in-place PCBC(fcrypt) decryption:

```
skcipher_request_set_crypt(req, sg, sg, 8, iv.x);
//                          ^^  ^^
//                          src==dst → 原地操作!
ret = crypto_skcipher_decrypt(req);  // 8字节写入发生在这里
```

skb_to_sgvec() directly converts skb's frag (including the page cache page injected by the attacker through splice()) to scatterlist, and src and dst point to the same sg. Therefore, the decryption operation writes the 8-byte "decryption result" back to the read-only page cache page.

Unlike xfrm-ESP, the value of an RxRPC primitive cannot be directly controlled-it is the result of fcrypt_decrypt(C, K). The attacker needs to register a key k through the add_key("rxrpc",...), and then brute force crack in the user state to find the k that can produce the target 8 bytes of plaintext. Fcrypt is a 56-bit key and 8-byte AFS-specific password. The brute force cracking space is controllable. The most important thing is that RxRPC paths do not require user namespaces, network namespaces, and CAP_NET_ADMIN. And Ubuntu loads the rxrpc.ko module by default.

3.4 Chain complementarity of primitives: a transition from a single point defect to a cross-subsystem universal lifting framework

Dirty Frag's breakthrough lies in the engineering design of primitive chain complementarity:

xfrm-ESP primitives provide 4-byte directly controllable writes, but are limited to namespaces;

The RxRPC primitive provides 8-byte indirect controlled writes, but does not require any privileges.

Hyunwoo Kim's utilization chain logic is as follows: first, the RxRPC primitive is used to obtain preliminary write capability in an unprivileged environment, or the xfrm-ESP primitive is used to implement precise write in an environment that allows namespaces. This "splicing" approach means that even if a single primitive has environmental constraints, an attacker can still achieve universal empowerment by complementing the primitives across subsystems.

This is the engineering upgrade of modern vulnerability exploitation: from "single-point primitive" to "multi-primitive chain combination", from "vulnerability of specific subsystems" to "cross-subsystem, cross-release universal empowerment framework".

3.5 ″Dirty″ Family Pedigree: Original Language Inheritance and Ability Evolution of Dirty Pipe → Copy Fail → Dirty Frag

Dirty Frag, Copy Fail and the earlier Dirty Pipe constitute an increasingly mature attack spectrum. The three generations of evolution reflect the discovery, repetition and generalization of page cache write primitives:

Intergenerational	Vulnerability	Core Subsystem	Page Cache Write Capability	Iconic significance
First generation	Dirty Pipe(CVE-2022-0847)	Pipe Subsystem	First time proving page cache tamperable	Creating the "Dirty" attack paradigm, Max Kellermann found
Second Generation	Copy Fail(CVE-2026-31431)	'AF_ALG' encryption subsystem	Write through cryptographic operations	AI-aided discovery, Theori/Xint Code proof paradigm is reproducible
Third Generation	Dirty Frag(CVE-2026-43284,CVE-2026-43500)	xfrm-ESP RxRPC	Double primitive chain utilization, bypassing environmental restrictions.	Generalized, engineered, weaponized, implemented Hyunwoo Kim

The core rule of the evolution of three generations of vulnerability discovery: each generation proves the feasibility of the same type of attack primitive, but each generation realizes the leap in subsystem selection, environment adaptation and engineering reliability. Dirty Frag is called the "successor" (successor) of Copy Fail, not because it has some kind of "task" inheritance relationship, but because both provide the same kind of attack primitive-page cache controlled write-not because they exploit similar code defects.

4. Methodological perspective of the vulnerability discovery process

4.1 Research Path: Hyunwoo Kim's Thinking Transformation from Subsystem Audit to "Primitive Hunting"

Hyunwoo Kim(@ v4bel, Kim Hyun-woo) is an independent vulnerability researcher focusing on Linux kernel. From 2022 to 2025, he worked for Theori, a South Korean security company. He joined Theori a year earlier than Lee (Li Taiyang), the discoverer of Copy Fail. Kim has won the Google kernelCTF Award and the Pwnie Awards 2025 Best Rights Award. Its research path presents distinct paradigm characteristics: from subsystem function audit to attack primitive hunting.

The traditional audit path is: select a subsystem → read code → find bugs → construct POC. Kim's path is: define the target primitive (page cache write) → enumerate all subsystems that may provide the primitive → verify the deterministic logic defects of each candidate path. The essence of this shift in thinking is to shift from "code-centric" to "competence-centric"-code is only a way to obtain primitives, and primitives themselves are the research goals.

In Dirty Frag's discovery, Kim systematically combed through all the subsystems in the kernel that involve "in-place decryption/in-place modification" and interact with the page cache: xfrm-ESP(IPsec), RxRPC(AFS/Kerberos), and possibly other encryption/network

paths. This "primitive hunting" thinking allows it to cross seemingly unrelated subsystems (IPsec and AFS) and discover the page cache write capabilities they collectively provide.

4.2 discovery logic: systematic combing of page cache tampering attack surface and deterministic logic defect identification

Dirty Frag's discovery logic can be broken down into three levels:

Layer 1: Attack surface enumeration. Identify code paths in all kernel subsystems that meet the following criteria:

- Processing data from the user state (such as the page cache page injected by splice());
- Perform write-in-place operations (decryption, decompression, header modification, etc.);
- The COW check is not performed on the shared page.

Layer 2: deterministic logic defect identification. Prioritize logic defects that do not rely on race conditions. Both of Dirty Frag's primitives are deterministic logic vulnerabilities-they don't rely on time windows, fail without causing kernel panic, and have a very high success rate. This is in sharp contrast to traditional race condition vulnerabilities (such as UAF), which reflects the ultimate pursuit of reliability in engineering exploit development.

Layer 3: environment adaptation verification. Verify the environment constraints (namespace requirements, module loading state, privilege requirements) for each primitive individually, and then look for complementary primitives to cover the constrained environment.

4.3 Engineering thinking: multi-primitive chain combination, environment adaptation and

utilization reliability design

Dirty Frag's disclosure document demonstrates a high degree of engineering thinking:

Multi-primitive chain combination: instead of pursuing a single "perfect loophole", two flawed primitives are spliced together into a common weapon.

Environment adaptation: Design adaptive utilization paths for different default configurations of Ubuntu(AppArmor restricted namespaces) and RHEL (possibly allowed namespaces).

Reliability design: Deterministic logic vulnerabilities ensure that the success rate of exploitation is close to 100 percent, and the failure does not cause the system to crash, suitable for use in the actual combat environment.

This engineering thinking marks the transformation of kernel vulnerability research from "artisan handwork" to "industrial production"-researchers design attack chains like supply chains and manage primitive combinations like inventory.

5. The man-machine division of labor in vulnerability discovery and analysis is rediscussed.

5.1 Paradigm Shift: Explosion of AI-assisted Vulnerability Mining-Still Taking Theori / Xint Code's Copy Fail Scanning Practice as an Example

The discovery process of Copy Fail is an example sample of human-machine collaborative vulnerability mining. Taeyang Lee identified a Copy Fail vulnerability in the Linux kernel AF_ALG subsystem through AI-assisted scanning with the Xint Code platform.

AI core value in this process is to ask all relevant questions:

- Traverse the possible page cache interaction paths in all subsystems;
- Identify all code patterns where COW checks are not performed;

Associates historical patches (such as the Avoid skb_cow_data series) with the current code state;

- Pattern matching across versions, across subsystems, in timescales where human effort is not feasible.

5.2 The Application Domain of Machine Intelligence: Large-scale Pattern Recognition, Primacy Candidate Discovery and Code Path Association

The applicable domain of machine intelligence in vulnerability discovery can be clearly defined:

Applicable Domain	Machine capacity	Human Limitations
large scale pattern recognition	Can match "Write-in-place without checking shared pages" mode in millions of lines of code in seconds	Manpower cannot audit all subsystems of the kernel line by line.
primitive candidate discovery	Automatically enumerates all possible code paths that provide page cache writes	Humans are susceptible to cognitive biases, omitting non-intuitive paths
Code Path Association	Automatic Association of 2017 'Avoid skb_cow_data' Submission and 2026 'splice()' Interaction Path	Humans have difficulty building causal chains across nine-year timescales
Historical vulnerability reuse	Identify CVE-2022-27666 from same module	Humans May Factor Systematic Differences

	as Dirty Frag vulnerability	and Ignore Associations

5.3 Core Values of Human Researcher: Semantic Reasoning of Primordia, Chain Combinatorial Innovation and Ethical Judgment

However, the boundaries of machine intelligence are equally clear. In Dirty Frag's discovery, Hyunwoo Kim shows the irreplaceable threefold value of human beings:

First, primitive semantic reasoning. The machine can recognize "there is a write operation here", but only humans can judge "whether this write operation constitutes a page cache write primitive", "whether the value written is controllable", "whether the degree of controllability is sufficient to cover the setuid binary". Semantic reasoning involves a deep understanding of kernel behavior, cryptographic protocols, and memory management semantics, which are weaknesses of current AI.

Second, chain combination innovation. Machines can find xfrm-ESP defects and RxRPC defects separately, but identifying the two as "complementary primitive pairs" and designing chain utilization logic requires human-level creative thinking and system-level architecture vision.

Third, ethical judgment. The boundaries of responsible disclosure, the timing of the exploitation of code, impact assessment on critical infrastructure-these decisions involving value tradeoffs must be made by humans. AI can calculate the probability of risk, but cannot assume ethical responsibility.

5.4 Ideal Model of Human-Machine Collaboration: Division Boundary of ″Human Construction Paradigm, Machine Hunting Primus″

Based on the above analysis, we once again emphasize the division of labor on human-machine collaboration in the previous Copy Fail analysis:

Humans are responsible for "asking the right questions"-constructing the paradigm; machines are responsible for "asking all relevant questions"-hunting primitives.

The specific division of labor is as follows:

-Human (security researcher): undertakes cognitive-intensive tasks such as semantic verification of primitives, chain combination design, use of reliability engineering, ethical judgment and disclosure strategy development, etc., to transform primitive candidates into practical capabilities.

Machines (AI/automation tools): undertake computation-intensive tasks such as primitive candidate discovery, pattern matching, cross-version association, historical vulnerability comparison, etc., and output the "primitive candidate list".

This division of labor is not simply "machine-assisted human", but "human-machine high-speed iterative transmission"-machines expand the space of primitives accessible to humans, and humans enhance the value of primitives discovered by machines.

5.5 The Enlightenment of on the Cultivation of Vulnerability Analysis Talents: From ″Digger″ to ″Original Language Architect″

The case of Dirty Frag puts forward profound enlightenment to the training of safety talents. The traditional talent training model focuses on "digging" skills-fuzzing, reverse, debugging. But in the era of man-machine collaboration, a single hole-digging skill is being partially replaced by machines. The core competence of the future should be the original language architect literacy:

• System-level vision: understanding how the subsystems of the kernel interact and identifying the possibility of combining primitives across subsystems;

• Abstract thinking ability: extract attack primitives from specific vulnerabilities, and design and exploit frameworks from the primitive level;

• Human-machine collaboration: extending hunting range with AI tools while maintaining critical validation of AI output;

• Engineering capabilities: combining primitives into reliable, generic, environmentally adaptive utilization chains.

5.6 Implications for Toolchain Construction: An Intelligent Platform for Original Recognition, Semantic Annotation and Chain Combinatorial Deduce

The construction of tool chain should evolve from "single point vulnerability scanning" to an intelligent support platform for "attack primitive recognition, semantic annotation and chain combination deduction. Specifically:

Primacy recognition engine: not only recognizes the crash point, but also the "atomic power" (read/write/execute/downgrade) given by the code path;

Semantic labeling system: labeling each candidate primitive with its controllability, certainty, environmental restrictions, and required privileges;

Chain combination deduction: based on the label data, automatically deduce the possibility of multi-original combination, output "environment, original language, chain" mapping map;

Human-computer interaction interface: allows researchers to make semantic corrections to machine output, supplement domain knowledge, and mark ethical constraints.

6. Primus Confrontation and Architectural Reflection from a

Defense Perspective

6.1 Defense Paradigm Shift: A Strategic Upgrade from ″Patching CVE″ to ″Eliminating Primtives″

The continuous exposure of Dirty Frag and Copy Fail exposes the fundamental limitations of the traditional "repair CVE" defense paradigm: CVE is the instantiation of primitives, and repairing a CVE only eliminates a primitive acquisition path, not the primitive itself. As long as the kernel base conditions for page cache write primitives exist (shared pages can be written in-place and COW is not enforced), new CVEs will emerge repeatedly in different subsystems.

The defense paradigm must be upgraded: from "patching CVEs" to "eliminating primitives". Specifically, the defender should identify all possible code paths in the kernel that provide page cache write primitives, and eliminate the existence of such primitives through architecture-level modifications (e. g., forced COW, shared page tag isolation).

6.2 Mitigation strategy for page cache write primitives: COW enforcement, shared page tags, and path isolation

For page cache write primitives, a three-tier mitigation strategy is available:

Layer 1: COW mandatory. Forces COW on shared pages in all kernel paths involving external data, such as splice() sources. The upstream repair patch (commit f4c50a4034e6) of Dirty Frag uses this idea: mark the external shared frag from splice(), identify and force the COW in the ESP input path.

Laye 2: shared page tags. Tag bits such as SKBFL_SHARED_FRAG are introduced to explicitly identify whether skb frag comes from an external shared page cache, so that all subsequent in-place operation paths (decryption, decompression, checksum calculation) are checked for tags before execution.

Layer 3: Path isolation. Isolate kernel paths that involve in-place writes (such as ESP decryption, RxRPC decryption) from paths that may receive external page cache pages (such as splice()) at the architecture level to ensure that they do not intersect on the same skb.

6.3 The Evolution of Kernel Security Architecture: Blocking the Propagation of Primordia between Subsystems and the Principle of Least Privilege

The cross-subsystem chaining of Dirty Frag reveals a deep problem with the kernel security architecture: the lack of a primitive propagation blocking mechanism between subsystems. xfrm-ESP and RxRPC are completely independent at the code level, but the attacker connects the two through the page cache, a shared resource. The defense architecture should introduce:

Inter-subsystem primitive propagation blocking: through memory domain isolation, reference permission refinement, to prevent the defects of one subsystem from being used to amplify the defects of another subsystem;

The principle of least privilege: non-essential subsystems (such as RxRPC) are disabled by default to reduce the attack surface;

Capability audit: Systematic audit of all "write-in-place" operations in the kernel and establishment of a whitelist mechanism.

6.4 mapping of the concept of ″executive governance″: running object-level constraints, audits and behavioral baselines

The concept of "executive governance" put forward by Antiy emphasizes that network security defense should return to the level of operating objects and establish a systematic

answer to "which executive bodies in the system are executing", "where the new executive bodies come from" and "how the executive bodies should be constrained. The Dirty Frag case validates the need for this concept from the attack side:

Which executables in the system are executing: the page cache write primitive is tampered with the execution image of the setuid binary in memory, and the defender must monitor the "integrity of the execution image" rather than just the "integrity of the disk file";

Where does the new execution body come from: the page cache page injected through splice() is an external source, and the defender should mark and audit the "non-file system directly loaded execution body page;

How the execution body should be constrained: For high-risk execution bodies such as setuid binary, it should be forced to run in a separate memory domain and prohibit sharing page cache pages with other subsystems.

7. Conclusions and Prospects

7.1 The Paradigm Significance of Attack Primal Thinking to Vulnerability Research: From Phenomenon Description to Capability Abstraction

Dirty Frag's discovery process proves that attacking primitive thinking is a key cognitive upgrade for vulnerability research from the "artisan era" to the "man-machine collaboration era. It requires the researcher to move from the phenomenal description of "which line of code is wrong" to the ability abstraction of "what atomic power does this code give me. Under this paradigm, CVE is no longer the end of research, but a by-product of primitive acquisition; vulnerability key information is no longer fragmented isolated knowledge about code defects, but a combinable, reusable, and evolvable knowledge structure.

7.2 Rebalancing of Human -Division of Labor:

Reshaping Core Competence and Platform Construction of Security Researchers

The explosive growth of AI is reshaping the human-machine division of vulnerability research. The efficiency advantage of machines in "asking all relevant questions" is irreversible, but the core value of human beings in "asking the right questions" is irreplaceable. Future security researchers must reshape their core capabilities: from "hole digger" to "primitive architect", from "code auditor" to "capability abstraction division", from "individual combat" to "man-machine cooperative commander". At the same time, security agencies must build an intelligent platform to support this transformation-a competence center for primitive recognition, semantic annotation and chain combination deduction.

7.3 The basic trend of future threat countermeasure and vulnerability response: the integration of intensive analysis platform and emergency coordination mechanism

Dirty Frag's CVE number has a vacuum period of about 1-2 days and a third-party disclosure event in advance, exposing the fragmentation dilemma of the vulnerability response system in the era of AI's comprehensive accelerated vulnerability discovery. For this reason, the security mechanism will also be recalibrated. Trend:

The value of large-scale threat and vulnerability analysis platform infrastructure will be highlighted: the construction of a comprehensive analysis platform based on the support of large-scale computing power, the use of common vertical models to serve researchers and defenders, support threat sample analysis, vulnerability mining intelligence sharing capabilities, has become a new type of security infrastructure. From the perspective of artificial intelligence network security, it is far more important to prevent "Skynet awakening.

The traditional emergency coordination mechanism should not only not be weakened, but also needs to be further strengthened. It must be transformed from the original event-triggered, process-driven, and collaborative response mechanism to a platform-based public safety service organization.

Predictive defense: Based on the attack primitive map, based on the DevSecOps perspective and the defense point perspective of security products, the code path that has the primitive foundation but has not yet found a specific CVE is pre-reinforced, and the vulnerability response is promoted from "knowing to promote prevention" to "unknown first prevention.

Ubiquitous perception: the more in the era when the vulnerability discovery and utilization time window is compressed, the more ubiquitous the security detection perception ability is, the more likely it is to perceive the targeted attack activity.

Conclusion

We must be aware that the vulnerability is not an object object, it is an attribute of the information system itself. It must exist in the complexity of the scene, can not die. The social resources, the speed of digital evolution, and the laws of practice prove that they do not have the resources to transform all systems into formalized proof systems (even if they do, they cannot die out attacks). The defender's effective target position on the vulnerability is to make the vulnerability difficult to exploit and effect, rather than the exhaustion and demise of the fantasy vulnerability, which is a long-term dynamic process.

Defenders need to build a "shield cube" type of security system, shrink the exposed surface, identify all the execution bodies, control the operation entrance, cut off the vulnerability exploitation chain, intercept the subsequent implantation of vulnerability exploitation, and control the behavior consequences of the execution body operation-all of which must be carried out continuously. The introduction of attack primitive thinking is precisely to more accurately identify the ability basis of vulnerability exploitation, more scientifically allocate the intellectual resources of human-computer collaboration, and more effectively build a defense system for current and future risks.

Appendix A: Dirty Frag / Copy Fail / Dirty Pipe Attack Primitive Competence Comparison Matrix

Comparison dimension	Dirty Pipe (CVE-2022-0847)	Copy Failure (CVE-2026-31431)	Dirty Frag(CVE-2026-43284,CVE-2026-43500)
Core Subsystem	Pipe	'AF_ALG' / 'algif_aead'	xfrm-ESP RxRPC
Page Cache Write Capability	Tampering through pipe splicing	Writing is achieved through encryption operations.	Double primitive chain combination writing
Write Size	Arbitrary (pipe write)	4 bytes (encrypted tail)	4 bytes (ESP) 8 bytes (RxRPC)
value controllability	Directly controllable	Directly controllable	ESP directly controllable/RxRPC indirectly controllable
Certainty	Logic vulnerability, no race	Logical vulnerability, no race condition required.	Logical vulnerability, no race condition required.
PRIVILEGE REQUIREMENTS	None	None	None (RxRPC path is completely unprivileged)
Namespace requirements	None	None	ESP Needs Namespace/RxRPC Needs No
Coverage 范围	Mainstream Releases	Mainstream distribution	Full range of mainstream releases
Discoverer	Max Kellermann	Theori / Xint Code / Taeyang Lee	Hyunwoo Kim(@v4bel)
Iconic significance	Create a page cache tampering paradigm	AI-assisted discovery, the proof paradigm is reproducible	Generalization, engineering, weaponization

Appendix B: Timeline of Historical Evolution of Linux Kernel Page Cache Write Primtives

Time	Event	Technical significance
Jan-17	Steffen Klassert Submission 'esp4/6: Avoid skb_cow_data whenever possible'	The Root-cause Introduction of Dirty Frag xfrm-ESP Prim
Feb-22	Max Kellermann Disclosure Dirty Pipe(CVE-2022-0847)	First proof that page cache can be tampered with, creating an attack paradigm
Mar-22	Steffen Klassert Signing CVE-2022-27666 Fix Patch	For the first time aware of the security risks optimized in 2017
Jun-23	RxRPC related commit introduced ('2dc334f1a63a', etc.)	Root cause introduction of Dirty Frag RxRPC primitive
Apr-26	Theori/Xint Code Disclosure Copy Fail(CVE-2026-31431)	AI-assisted discovery to prove that the page cache write paradigm is reproducible
May-26	Hyunwoo Kim reveals Dirty Frag	Double primitive chain combination, to achieve a universal lifting framework.