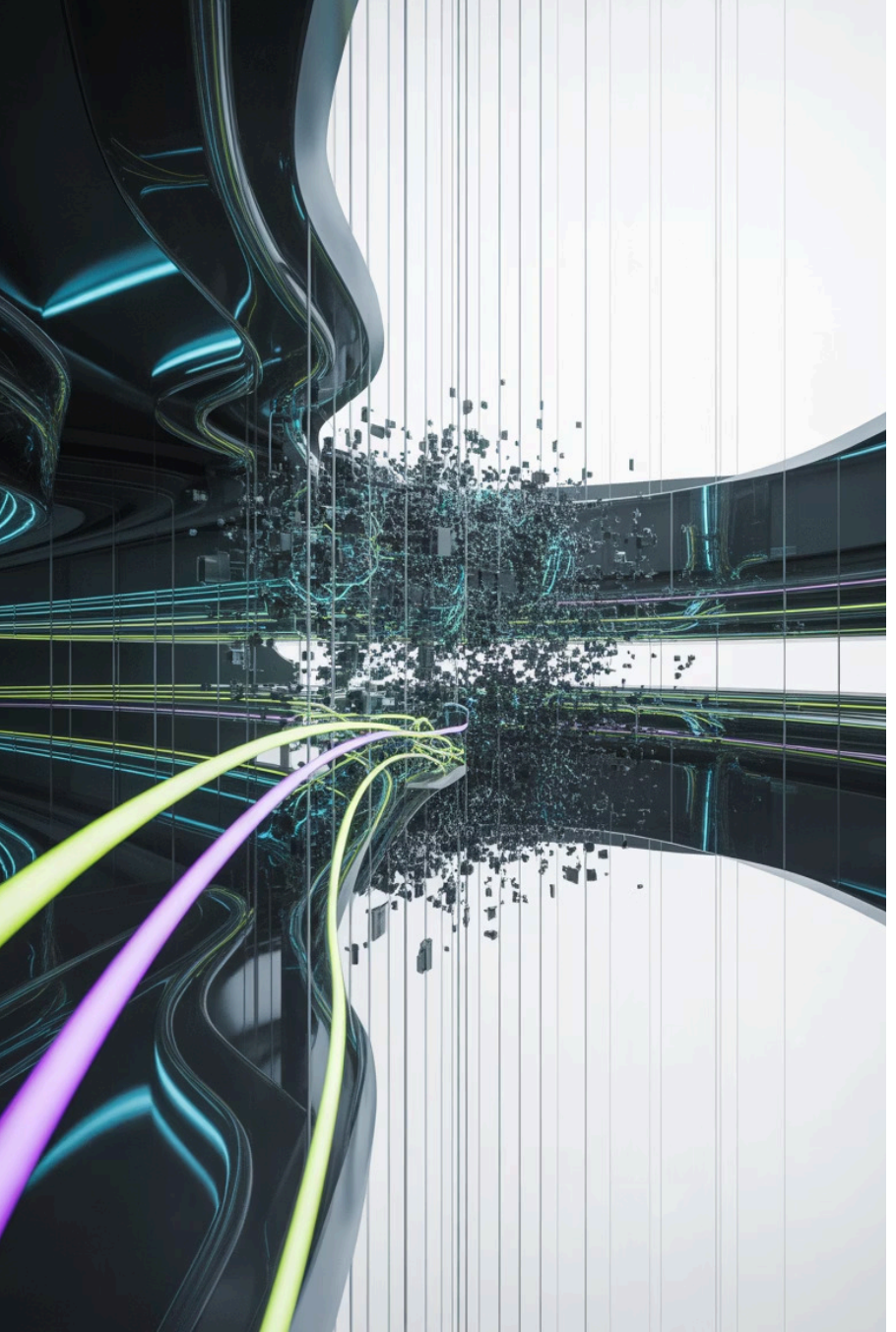


Understanding Teardrop Attacks: IP Fragmentation Vulnerabilities

IP fragmentation is a fundamental network mechanism that enables large IP datagrams to traverse networks with varying Maximum Transmission Unit (MTU) limits. When a datagram exceeds the MTU of a network segment, it must be divided into smaller fragments that can fit within the transmission constraints.

Each fragment carries critical metadata: an identification field shared across all fragments of the original datagram, an offset value indicating the fragment's position measured in 8-byte units, and a "more fragments" (MF) flag signaling whether additional fragments follow. The receiving host uses these parameters—source IP, destination IP, protocol, and identification—to collect and reassemble fragments into the original datagram using the offset information.

The Teardrop Attack Vector

Exploiting Reassembly Logic

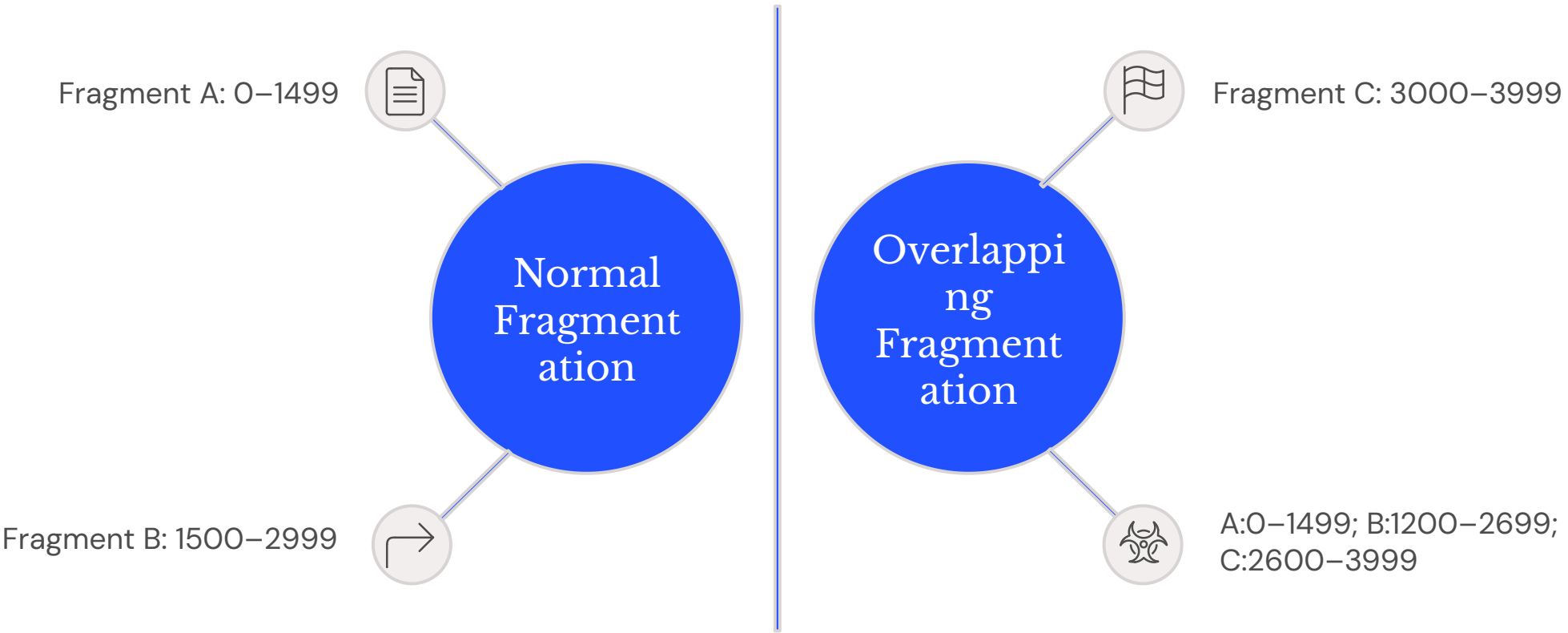
A teardrop-style attack deliberately sends malformed fragments with overlapping or contradictory offset and length fields. These malicious fragments are designed to confuse the reassembly logic of vulnerable network stacks.

Historically, operating systems and network devices contained reassembly code that lacked robust validation. When encountering fragments with overlapping offsets or inconsistent lengths, these systems would crash, hang, or incorrectly reassemble packets—triggering denial-of-service conditions.

The attack exploits a simple principle: it supplies fragments that violate the fundamental assumptions of correct reassembly—namely, that fragments should be non-overlapping with consistent lengths and offsets. Vulnerable implementations failed to validate or handle these contradictions safely.

Visualizing Fragment Behavior

Understanding the difference between legitimate and malicious fragmentation patterns is essential for defensive network engineering.



Normal Fragmentation

Fragment A: offset=0, length=1500 → bytes 0-1499

Fragment B: offset=1500, length=1500 → bytes 1500-2999

Fragment C: offset=3000, length=1000 → bytes 3000-3999

Clean, sequential coverage with no overlaps enables straightforward reassembly.

Overlapping Fragmentation

Fragment A: offset=0, length=1500 → bytes 0-1499

Fragment B: offset=1200, length=1500 → bytes 1200-2699 ← overlaps A

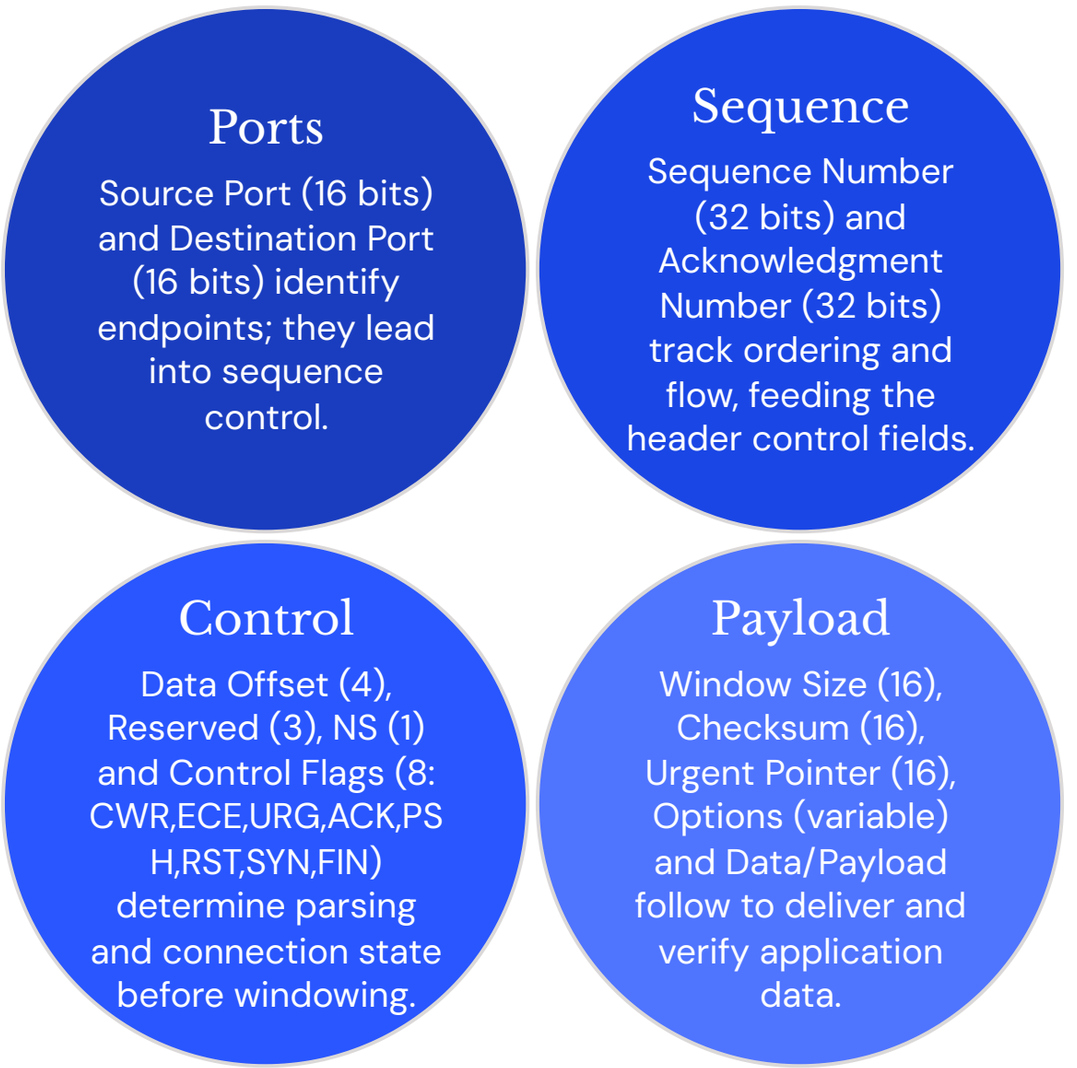
Fragment C: offset=2600, length=1400 → bytes 2600-3999 ← overlaps B

Conflicting byte ranges force reassemblers to make decisions about which data "wins" in overlap zones—buggy implementations crash or fail.

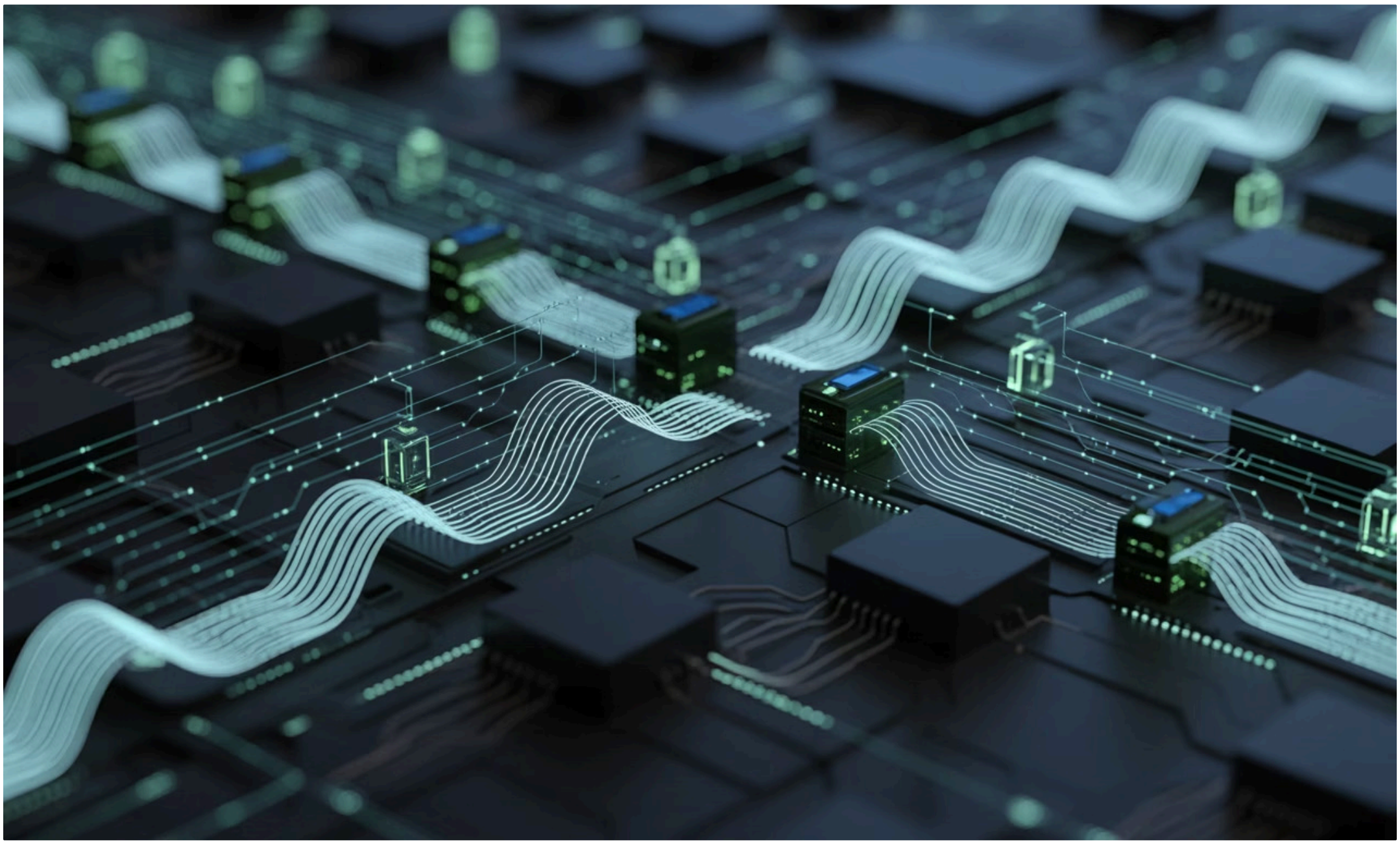
TCP Packet Structure

The Transmission Control Protocol (TCP) uses a structured header to ensure reliable, ordered, and error-checked delivery of data. Understanding its fields is crucial for network analysis and troubleshooting.

This visual representation breaks down the TCP header, highlighting the purpose and size of each critical field from the initial Source and Destination Ports to the final Data payload, enabling a clear understanding of how TCP packets are constructed and managed.



Technical Root Causes of Vulnerability



Bounds and Length Validation Failures

Reassembly code must track which byte ranges have been received. Implementations assuming non-overlapping fragments may use simple list operations or pointer arithmetic that becomes invalid when overlaps occur. This can lead to buffer overflows, memory corruption, or incorrect state management.

Integer Arithmetic and Memory Allocation Issues

Computing total reassembled size from malicious offset values can trigger integer overflows or cause incorrect memory allocation. When fragment fields are intentionally contradictory, size calculations may wrap around or produce nonsensical values, leading to crashes or exploitable conditions.

Stateful Reassembly Exhaustion

Reassemblers maintain state per IP identification value. Attackers can flood the system with numerous contradictory fragment sets to exhaust memory or overflow state tables, causing resource exhaustion and denial of service even without triggering a crash bug.

Implementation Variation Across Systems

Different operating systems handle overlapping fragments differently—some accept and overwrite data, others drop suspicious packets, and vulnerable systems crash when encountering unexpected conditions. This heterogeneity makes universal defense challenging and requires tailored mitigation strategies.

Defensive Strategy: Patching and Hardening

Core Defense Principles

Protecting against teardrop-style attacks requires a multi-layered approach combining system updates, configuration hardening, and behavioral monitoring. These defensive measures are practical, legally appropriate, and essential for network security professionals.

01

Patch and Update All Systems

Ensure endpoints, routers, firewalls, and intrusion prevention systems run current software versions. While original teardrop vulnerabilities are historical, reassembly bugs continue appearing in embedded devices and legacy systems. Vendor patches are your first line of defense.

02

Harden IP Reassembly Behavior

Configure network devices to perform reassembly validation and reject suspicious fragments. Enterprise firewalls often include settings to reassemble fragments at the perimeter and drop packets with overlapping or contradictory offsets. Enable "drop overlapping fragments" policies where available.

03

Enable OS-Level Mitigations

Modern operating systems include kernel-level protections against malformed fragments. Review vendor documentation to enable available mitigation options. These settings vary by OS but typically include fragment validation, timeout enforcement, and resource limits on reassembly operations.

Network-Level Filtering Controls

> "If legitimate traffic in your environment rarely uses fragmentation, consider implementing strict edge filtering policies. Defense in depth begins at the network perimeter."

Fragment Dropping at Network Edge

For environments where fragmented traffic is uncommon, configure edge routers and firewalls to drop inbound fragmented packets by default. This eliminates the attack vector entirely for networks with predictable traffic patterns.

Controlled Allowance for Required Protocols

Some protocols and tunneled traffic legitimately require fragmentation. Implement targeted allowance rules with strict inspection for these specific cases. Use application-aware filtering to permit only necessary fragmentation while blocking suspicious patterns.

Rate Limiting and Threshold Enforcement

Even when fragments must be permitted, enforce rate limits on fragmented traffic per source. Set thresholds for maximum concurrent reassembly operations and implement aggressive timeouts to prevent state table exhaustion attacks.

Detection and Monitoring Strategies

Monitor Fragment Rates

Track high rates of fragmented packets from individual sources. Legitimate fragmentation occurs occasionally; sustained fragment floods indicate potential attack activity or misconfigured systems requiring investigation.

Identify Inconsistent Fragments

Log fragments with inconsistent total lengths or overlapping offset values. These anomalies directly indicate potential teardrop-style attacks or serious network problems requiring immediate attention and response.

Track Reassembly Timeouts

Monitor for large numbers of incomplete reassembly states and timeouts. Excessive incomplete reassemblies suggest either network reliability issues or deliberate fragmentation attacks designed to exhaust system resources.

Use packet capture tools in controlled monitoring environments to inspect traffic anomalies. Analyze pcap files for repeated fragment IDs with contradictory offset ranges—these patterns provide forensic evidence of attack attempts and help refine detection rules.

Intrusion Detection and Network Segmentation

IDS/IPS Configuration

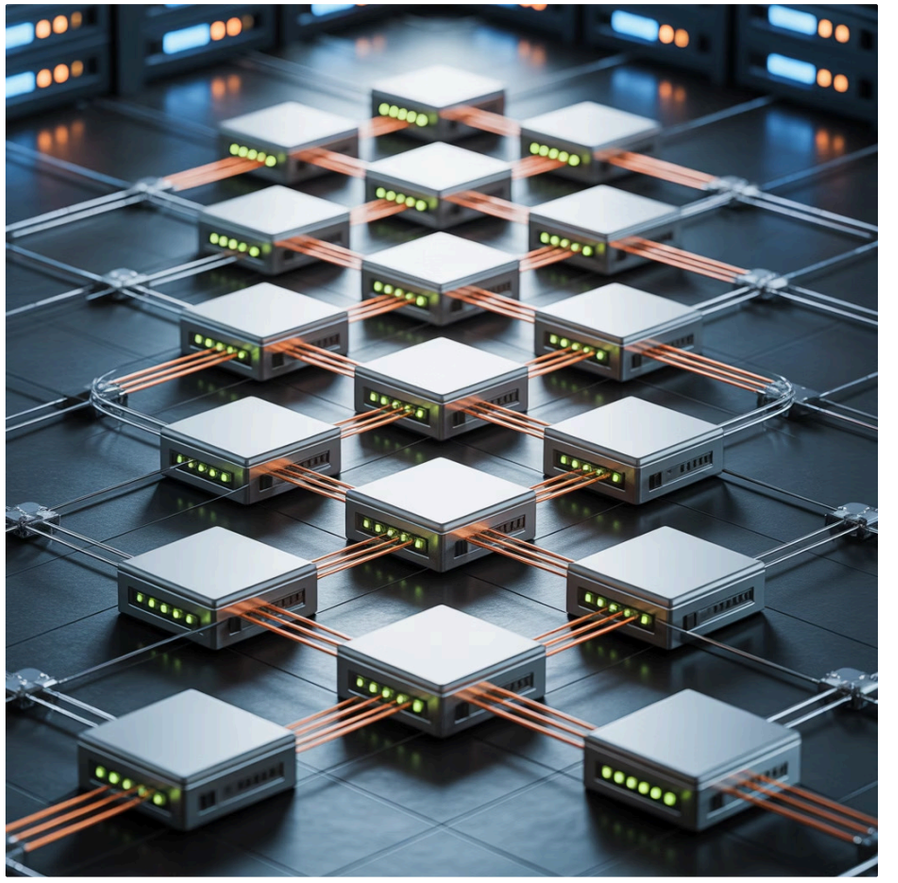
Leverage vendor intrusion detection and prevention system signatures that specifically detect overlapping fragment patterns and known teardrop attack variants. Modern IDS/IPS solutions include pre-built rules for these classic attacks, but signature sets must be kept current to detect emerging variations.

When enabling blocking rules, conduct thorough testing in controlled environments before production deployment. False positives can disrupt legitimate fragmented traffic, particularly in VPN or tunneling scenarios. Implement logging and alerting before automated blocking to understand traffic patterns.

Micro-Perimeter Security

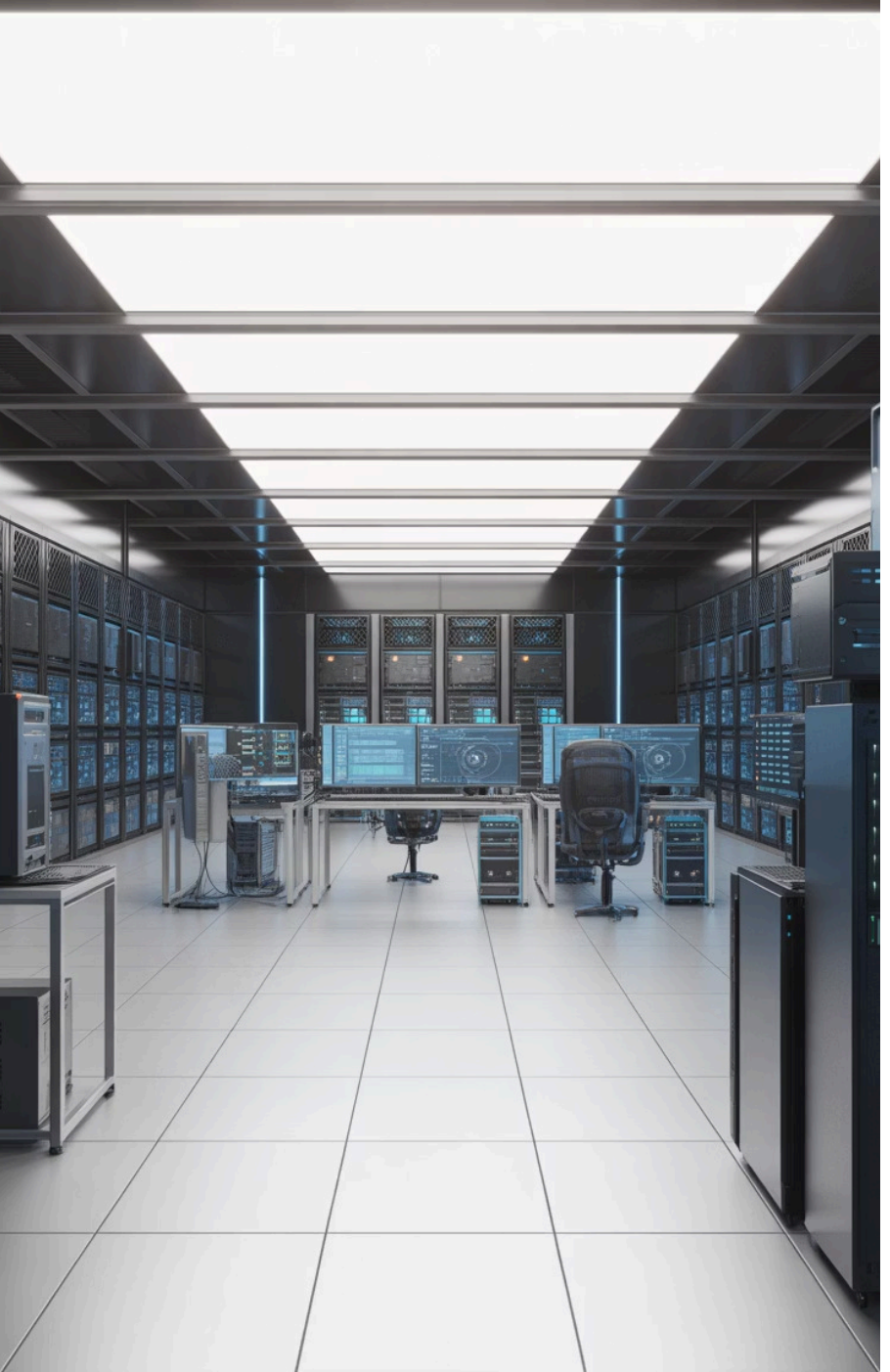
Implement network segmentation to isolate vulnerable legacy devices behind internal firewalls with strict filtering rules. This micro-perimeter approach limits exposure to untrusted networks and contains the blast radius of any successful attack.

For systems that cannot be immediately patched or updated, segmentation provides critical defense-in-depth protection while remediation efforts proceed.



Testing Considerations

Always test IDS/IPS rule changes in lab environments with representative traffic before production deployment. Document baseline fragment patterns in your environment to reduce false positive rates.



Controlled Laboratory Testing

Isolated Lab Environment

Conduct resilience testing only in air-gapped or strictly isolated lab networks with explicit written authorization. Never test attack techniques on production systems or networks you don't own.

Authorized Testing Tools

Use vendor-supported test tools and reputable vulnerability scanners. Commercial security testing platforms provide controlled methods to verify defenses without risking production stability or legal exposure.

Responsible Disclosure

If testing reveals previously unknown vulnerabilities, follow responsible disclosure practices. Report findings to vendors through proper channels with sufficient detail for remediation.

Maintain detailed documentation of all testing activities, including scope, methodology, findings, and remediation recommendations. This documentation provides evidence of due diligence and supports compliance with security audit requirements.

Identifying Problematic Fragments in Captures

Network forensics and defensive packet analysis require the ability to identify malformed fragments in traffic captures. Understanding what to look for enables rapid incident response and effective security monitoring.

1 Identify Common Fragment Identifiers

In packet captures, search for multiple packets sharing the same IP identification field, source/destination addresses, and protocol number. These parameters define a fragment group that should reassemble into a single datagram.

2 Detect Overlapping Offset Values

Compare fragment offset values within each group. Flag cases where a fragment's byte range begins before the previous fragment's range ends—this overlap indicates either network errors or deliberate attack traffic.

3 Track Incomplete Reassemblies

Monitor for fragment sequences where fragments arrive but the final fragment (MF=0) never appears. Repeated incomplete reassemblies from the same source suggest resource exhaustion attacks or serious network issues.

Wireshark Analysis

Display filters like `ip.flags.mf == 1 || ip.frag_offset > 0` help isolate fragmented traffic. Combine with `ip.reassembled.error` to find reassembly failures. Export matching packets for detailed offline analysis.

Additional Resources for Network Defenders

RFC Standards Documentation

RFC 791: Internet Protocol specification defining fragmentation behavior

RFC 815: IP datagram reassembly algorithms and best practices

These foundational documents provide authoritative reference material for understanding correct fragmentation and reassembly implementations.

Vendor and CERT Advisories

Review security advisories from your OS and network device vendors regarding fragmentation vulnerabilities. CERT coordination centers publish detailed vulnerability analyses and mitigation guidance for specific products and versions.

Vulnerability Databases

Search MITRE CVE and National Vulnerability Database (NVD) for historical teardrop vulnerabilities. Understanding past exploits and patches provides context for evaluating current security posture and identifying similar patterns.

Educational Materials

Network security textbooks and professional courses covering packet handling, defensive programming, and secure protocol implementation. Formal education in these areas builds the foundation for effective security engineering.