LORIKEET SECURITY

Responsible Disclosure Report

FastNetMon Community Edition <= 1.2.9

16 Security Vulnerabilities

CVE-2026-48682 through CVE-2026-48697

2 Critical | 11 High | 3 Medium

Remote Code Execution • Command Injection • Memory Corruption
Authentication Bypass • Cryptographic Failures

lorikeetsecurity.com
Responsible Disclosure Document — May 2026

Executive Summary

Lorikeet Security conducted a comprehensive source code audit of FastNetMon Community Edition version 1.2.9, the latest release of this widely-deployed open-source DDoS detection and mitigation tool. The audit identified **16 distinct security vulnerabilities** spanning remote code execution, OS command injection, memory corruption, authentication bypass, and cryptographic failures.

FastNetMon processes untrusted network telemetry (NetFlow v9, sFlow v5, IPFIX) over unauthenticated UDP, implements custom BGP protocol handling, executes shell commands as part of its attack response pipeline, and typically runs with root privileges for raw packet capture. The combination of unauthenticated network-facing parsers, memory-unsafe C++ code, and the complete absence of compiler hardening flags creates a significant attack surface.

Affected Software: FastNetMon Community Edition <= 1.2.9
Vendor: FastNetMon LTD / Pavel Odintsov
Repository: https://github.com/pavel-odintsov/fastnetmon
Fix Available: No (as of disclosure date)
Researcher: Lorikeet Security (lorikeetsecurity.com)

Severity	Count	Categories
Critical	2	Heap buffer overflow (RCE), Stack buffer overflow (RCE)
High	11	OOB reads, integer overflows, command injection, auth bypass, symlink attack
Medium	3	Missing TLS verification, ExaBGP stack overflow, BGP parse confusion

Network Attack Surface

FastNetMon Community Edition binds the following network services by default: NetFlow/IPFIX UDP on 0.0.0.0:2055 (unauthenticated), sFlow UDP on 0.0.0.0:6343 (unauthenticated), gRPC API on 127.0.0.1:50052 (InsecureServerCredentials, no auth). Telemetry is POSTed to community-stats.fastnetmon.com every 3600 seconds without TLS certificate verification. FastNetMon does not run a BGP daemon; it uses GoBGP or ExaBGP as external BGP speakers.

Compiler Hardening

The default CMake build has zero security flags: no -fstack-protector, no -D_FORTIFY_SOURCE=2, no -fPIE/-pie, no RELRO. This makes all stack and heap overflows directly exploitable without bypass techniques.

Table of Contents

CVE-2026-48682

Out-of-Bounds Read via Unvalidated IPv4 IHL Field

HIGH CVSS: 7.5 (High)

Vulnerability Type:	CWE-125 (Out-of-bounds Read), CWE-20 (Improper Input Validation)
Attack Vector:	Remote (sFlow/NetFlow/PCAP packet capture)
Affected File:	src/simple_packet_parser_ng.cpp, lines 130-164
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The IPv4 packet parser in simple_packet_parser_ng.cpp validates that the packet buffer contains at least sizeof(ipv4_header_t) (20 bytes), but then advances the internal pointer by 4 * IHL (up to 60 bytes for IHL=15) without any subsequent bounds check. This creates a 40-byte out-of-bounds read window past the validated region. The vulnerability is reachable via any packet capture plugin including sFlow raw packet headers (UDP port 6343), NetFlow sampled packets, AF_PACKET, PCAP, or XDP.

Vulnerable Code

```
// Line 130: Only validates 20 bytes
if (local_pointer + sizeof(ipv4_header_t) > end_pointer) {
return parser_code_t::memory_violation;
}

// Line 164: Advances by up to 60 bytes - NO BOUNDS CHECK
local_pointer += 4 * ipv4_header->get_ihl();
```

Impact

Information disclosure via heap memory leakage and denial of service via segmentation fault. With IHL values 0-4, the pointer advances less than the header size, causing TCP/UDP header parsing to occur within the IP header data (type confusion). Remotely triggerable via unauthenticated sFlow UDP on port 6343.

Remediation

Add validation: if (ipv4_header->get_ihl() < 5) return parser_code_t::broken_packet; and if (local_pointer + 4 * ipv4_header->get_ihl() > end_pointer) return parser_code_t::memory_violation;

CVE-2026-48683

Out-of-Bounds Read in NetFlow v9 Data Flowset Processing

HIGH CVSS: 7.5 (High)

Vulnerability Type:	CWE-125 (Out-of-bounds Read)
Attack Vector:	Remote (NetFlow v9 UDP, default port 2055)
Affected File:	src/netflow_plugin/netflow_v9_collector.cpp, lines 1695-1702
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The NetFlow v9 data flowset processing loop iterates over flow records without per-iteration bounds checking against the packet boundary. The Options flowset branch at lines 1709-1719 correctly performs this check, confirming the omission is a bug rather than a design choice. An attacker sends a template packet followed by a data packet with an inflated flowset_length, causing the loop to read past the UDP receive buffer into adjacent heap memory.

Vulnerable Code

```
// Data branch - NO bounds check
for (uint32_t i = 0; i < num_flowsets; i++) {
netflow9_flowset_to_store(pkt + offset, ...);
offset += field_template->total_length;
}
// Options branch - HAS bounds check
for (uint32_t i = 0; i < num_flowsets; i++) {
if (pkt + offset + field_template->total_length > packet_end) {
return false; // Correct!
}
...
}
```

Impact

Information disclosure (heap memory contents leaked via flow processing) and denial of service. Exploitable via unauthenticated UDP to the NetFlow collector port (default 2055, binds 0.0.0.0).

Remediation

Add the same bounds check from the Options branch to the Data branch: if (pkt + offset + field_template->total_length > packet_end) return false;

CVE-2026-48684

Out-of-Bounds Read in NetFlow v9 Options Template Parsing

HIGH CVSS: 7.5 (High)

Vulnerability Type:	CWE-125 (Out-of-bounds Read)
Attack Vector:	Remote (NetFlow v9 UDP, default port 2055)
Affected File:	src/netflow_plugin/netflow_v9_collector.cpp, lines 224-257
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The process_netflow_v9_options_template() function iterates based on attacker-controlled option_scope_length and option_length fields, reading netflow9_template_flowset_record_t structures (4 bytes each) without validating that each read position remains within the flowset boundary. Setting these length fields to values larger than the actual flowset data causes reads past the packet buffer.

Vulnerable Code

```
for (; scopes_offset < fast_ntoh(
options_nested_header->option_scope_length);) {
netflow9_template_flowset_record_t* tmplr =
(netflow9_template_flowset_record_t*)
(zone_address + scopes_offset); // OOB read
scopes_total_size += fast_ntoh(tmplr->length);
scopes_offset += sizeof(*tmplr);
}
```

Impact

Information disclosure and denial of service via unauthenticated UDP. Additionally, when scope/option lengths are not multiples of 4, misaligned reads of the 4-byte record structures occur, compounding the parsing error.

Remediation

Validate that zone_address + scopes_offset + sizeof(*tmplr) <= flowset_end before each dereference.

CVE-2026-48685

BGP Extended Length Attribute Parsing Reads 1 of 2 Length Bytes

MEDIUM CVSS: 6.5 (Medium)

Vulnerability Type:	CWE-130 (Improper Handling of Length Parameter Inconsistency)
Attack Vector:	Remote (via BGP peer through GoBGP)
Affected File:	src/bgp_protocol.hpp, line 173
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

When the BGP extended length flag is set (indicating a 2-byte length field per RFC 4271 Section 4.3), the parse_raw_bgp_attribute() function reads only a single byte for attribute_value_length. An attribute of 300 bytes (0x012C) is misread as 44 bytes (0x2C). The remaining 256 bytes are then misinterpreted as the start of subsequent attributes, causing cascading parse failures. FastNetMon does not implement a BGP listener; this is triggered when processing BGP data received from an external GoBGP daemon.

Vulnerable Code

```
if (attr_flags.extended_length_bit == 1) {
length_of_length_field = 2; // Correctly identifies 2-byte
}
// ...
attribute_value_length = value[2]; // BUG: Always reads 1 byte
// Should be: attribute_value_length = (value[2] << 8) | value[3];
```

Impact

Denial of service via parse confusion, potential out-of-bounds reads when orphaned bytes are interpreted as attribute headers. Triggerable by a malicious BGP peer sending crafted UPDATE messages through GoBGP.

Remediation

When extended_length_bit == 1, read 2 bytes: attribute_value_length = (value[2] << 8) | value[3];

CVE-2026-48686

Stack Buffer Overflow via Unvalidated BGP NLRI Prefix Length

CRITICAL CVSS: 9.8 (Critical)

Vulnerability Type:	CWE-787 (Out-of-bounds Write), CWE-120 (Buffer Copy without Checking Size)
Attack Vector:	Remote (via BGP peer through GoBGP)
Affected File:	src/bgp_protocol.cpp, lines 93-111
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The decode_bgp_subnet_encoding_ipv4_raw() function reads prefix_bit_length from untrusted BGP packet data without validating it is <= 32 for IPv4. This value is passed to how_much_bytes_we_need_for_storing_certain_subnet_mask() which computes ceil(prefix_bit_length / 8), returning up to 32 for prefix_bit_length=255. The result is used as the memcpy length, copying into a 4-byte uint32_t stack variable (prefix_ipv4) - a 28-byte stack buffer overflow with attacker-controlled content.

Vulnerable Code

```
uint8_t prefix_bit_length = value[0]; // Attacker: 255
uint32_t prefix_byte_length =
how_much_bytes_we_need(prefix_bit_length); // = 32

uint32_t prefix_ipv4 = 0; // 4 bytes on stack
memcpy(&prefix_ipv4, value + 1, prefix_byte_length);
// Copies 32 bytes into 4-byte variable = 28-byte OVERFLOW
```

Impact

Remote code execution. The attacker controls 28 bytes of stack overflow data, sufficient to overwrite the saved return address. No stack protector (-fstack-protector) is enabled in the build. Additionally, convert_cidr_to_binary_netmask() computes (32 - cidr) which wraps to a huge value for cidr > 32, causing undefined behavior.

Remediation

Add validation: if (prefix_bit_length > 32) return false; before the memcpy call.

CVE-2026-48687

OS Command Injection in Juniper Plugin Logging Function

HIGH CVSS: 8.1 (High)

Vulnerability Type:	CWE-78 (OS Command Injection)
Attack Vector:	Indirect remote (via attack notification pipeline)
Affected File:	src/juniper_plugin/fastnetmon_juniper.php, lines 115-119
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The _log() function passes unsanitized message data directly to PHP exec() inside a shell command string. The message variable ($msg) is constructed from argv[1] ($IP_ATTACK), argv[2] ($DIRECTION_ATTACK), and argv[3] ($POWER_ATTACK), all of which are attacker-influenceable through the FastNetMon attack notification pipeline. Shell metacharacters (semicolons, backticks, $() substitution) in these values result in arbitrary command execution.

Vulnerable Code

```
function _log( $msg ) {
$FILE_LOG_TMP = "/tmp/fastnetmon_api_juniper.log";
exec( "echo `date` \"- [FASTNETMON] - " .
$msg . " \" >> " . $FILE_LOG_TMP );
}
```

Impact

Remote code execution as the FastNetMon process user. While the C++ core currently passes IP addresses via inet_ntoa() (safe for IPv4), the PHP script performs no input validation of its own and is vulnerable if invoked by any other mechanism or with IPv6/string inputs.

Remediation

Replace exec() with file_put_contents() for logging, or use escapeshellarg() on all interpolated values.

CVE-2026-48688

Out-of-Bounds Reads in BGP MP_REACH_NLRI IPv6 Decoder

HIGH CVSS: 7.5 (High)

Vulnerability Type:	CWE-125 (Out-of-bounds Read)
Attack Vector:	Remote (via BGP peer through GoBGP)
Affected File:	src/bgp_protocol.cpp, lines 155-209
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The decode_mp_reach_ipv6() function contains a TODO comment explicitly acknowledging missing safety checks: 'we should add sanity checks to avoid reads after attribute\'s memory block.' Multiple pointer dereferences use attacker-controlled offsets without bounds validation. The function casts raw pointers to bgp_mp_reach_short_header_t without size checks, uses attacker-controlled length_of_next_hop in a memcpy, computes prefix_length from multiple attacker-controlled offsets, and uses attacker-derived byte counts in a final memcpy.

Vulnerable Code

```
// TODO: we should add sanity checks to avoid
// reads after attribute's memory block

memcpy(&next_hop_ipv6.subnet_address, ptr,
bgp_mp_ext_header->length_of_next_hop);
// ^ Attacker-controlled length, no bounds check

uint8_t* prefix_length = ptr + sizeof(header)
+ length_of_next_hop + sizeof(uint8_t);
// ^ Multiple attacker-controlled offsets, no check
```

Impact

Information disclosure via heap memory leakage and denial of service. A malicious BGP peer can craft MP_REACH_NLRI attributes to read arbitrary memory past the attribute boundary.

Remediation

Add bounds validation before every pointer dereference, checking that offset + read_size <= attribute_end.

CVE-2026-48689

Off-by-One Heap Buffer Overflow in dynamic_binary_buffer_t

CRITICAL CVSS: 9.8 (Critical)

Vulnerability Type:	CWE-193 (Off-By-One Error), CWE-122 (Heap-based Buffer Overflow)
Attack Vector:	Remote (NetFlow/sFlow/IPFIX/BGP processing)
Affected File:	src/dynamic_binary_buffer.hpp, lines 101, 110, 121, 149, 160
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The dynamic_binary_buffer_t class, used throughout FastNetMon for binary protocol serialization, contains an identical off-by-one error in five methods. The bounds check uses '> maximum_internal_storage_size + 1' instead of the correct '> maximum_internal_storage_size', allowing exactly 1 byte to be written past the heap-allocated buffer. The append_byte() method at line 87 uses the correct check, confirming inconsistency. The developer left a 'TODO: Why +1?' comment at line 100, acknowledging the issue.

Vulnerable Code

```
// Line 110 - BUGGY check (5 methods)
if (internal_data_shift + length >
maximum_internal_storage_size + 1) // BUG: +1

// Line 87 - CORRECT check (append_byte)
if (internal_data_shift > maximum_internal_storage_size - 1)

// TODO: Why +1? (line 100)
```

Impact

Deterministic 1-byte heap write primitive. On glibc systems, this corrupts adjacent heap chunk metadata, a well-known primitive for achieving arbitrary code execution. Triggerable via any protocol path that uses the buffer class (NetFlow templates, BGP messages, Flow Spec NLRI). Compounded by absence of compiler hardening flags.

Remediation

Replace '> maximum_internal_storage_size + 1' with '> maximum_internal_storage_size' in all five methods.

CVE-2026-48690

Integer Overflow in Packet Capture Buffer Allocation

HIGH CVSS: 7.0 (High)

Vulnerability Type:	CWE-190 (Integer Overflow), CWE-122 (Heap-based Buffer Overflow)
Attack Vector:	Local (configuration file)
Affected File:	src/packet_storage.hpp, lines 23-25
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The allocate_buffer() function computes memory_size_in_bytes using 32-bit unsigned arithmetic: buffer_size_in_packets * (max_captured_packet_size + sizeof(fastnetmon_pcap_pkthdr_t)) + sizeof(fastnetmon_pcap_file_header_t). With max_captured_packet_size=1500 and sizeof(pkthdr)~16, each slot uses ~1516 bytes. When buffer_size_in_packets >= 2,832,542, the multiplication exceeds UINT32_MAX and wraps, resulting in a small allocation. Subsequent write_packet() calls overflow the heap buffer.

Vulnerable Code

```
bool allocate_buffer(unsigned int buffer_size_in_packets) {
unsigned int memory_size_in_bytes =
buffer_size_in_packets *
(max_captured_packet_size +
sizeof(fastnetmon_pcap_pkthdr_t))
+ sizeof(fastnetmon_pcap_file_header_t);
memory_pointer = (unsigned char*)malloc(
memory_size_in_bytes); // Tiny allocation
```

Impact

Heap buffer overflow via malicious configuration. The buffer_size_in_packets value comes from ban_details_records_count in the config file, parsed with atoi() with no overflow or range checking.

Remediation

Use 64-bit arithmetic (size_t) or add an explicit upper bound: if (buffer_size_in_packets > 1000000) return false;

CVE-2026-48691

Integer Overflow in BGP AS_PATH Attribute Length

HIGH CVSS: 7.5 (High)

Vulnerability Type:	CWE-190 (Integer Overflow), CWE-122 (Heap-based Buffer Overflow)
Attack Vector:	Remote (via BGP peer through GoBGP or gRPC API)
Affected File:	src/bgp_protocol.hpp, lines 600-621
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The AS_PATH attribute length is stored in a uint8_t field. When the actual length exceeds 255 bytes (more than 63 ASNs in the path), the value silently truncates. With 64 ASNs: 2 + 64*4 = 258, which wraps to 2 in uint8_t. The dynamic_binary_buffer is then allocated based on this truncated length, and all ASNs are appended, overflowing the buffer. The path_segment_length field at line 621 has the same uint8_t truncation issue with more than 255 ASNs.

Vulnerable Code

```
bgp_attribute_as_path.attribute_length =
sizeof(bgp_as_path_segment_element_t) +
this->as_path_asns.size() * sizeof(uint32_t);
// uint8_t: 2 + 64*4 = 258 wraps to 2

// Buffer allocated based on truncated length
as_path_as_binary_array.set_maximum_buffer_size(
as_path_attribute_full_length); // Too small!
// Then all ASNs appended -> heap overflow
```

Impact

Heap buffer overflow when constructing BGP UPDATE messages with large AS_PATH. Triggerable via the unauthenticated gRPC API or when processing routes from BGP peers with long AS paths.

Remediation

Use uint16_t with the extended length flag, or validate as_path_asns.size() <= 63 before encoding.

CVE-2026-48692

Unauthenticated gRPC API Allows Ban/Unban and Script Execution

HIGH CVSS: 8.1 (High)

Vulnerability Type:	CWE-306 (Missing Authentication for Critical Function)
Attack Vector:	Local (network-adjacent if bind address changed to 0.0.0.0)
Affected File:	src/fastnetmon.cpp, line 477; src/api.cpp (all methods)
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The gRPC API server is initialized with grpc::InsecureServerCredentials() and exposes critical network operations without any authentication. The source code comment on line 476 explicitly states: 'Listen on the given address without any authentication mechanism.' All RPC methods (ExecuteBan, ExecuteUnBan, GetBanlist, GetTotalTrafficCounters) have zero credential checks. Default bind is 127.0.0.1:50052, but this is a configurable option.

Vulnerable Code

```
std::string server_address("127.0.0.1:50052");
ServerBuilder builder;
// Listen on the given address without any
// authentication mechanism.
builder.AddListeningPort(server_address,
grpc::InsecureServerCredentials());
```

Impact

Any local process can blackhole arbitrary IPs via BGP (ExecuteBan), remove active DDoS mitigations (ExecuteUnBan), trigger shell script execution via the notify script pipeline (popen()), and enumerate monitored networks. If the bind address is changed to 0.0.0.0, all of the above becomes remotely exploitable. No RBAC separates monitoring from admin operations.

Remediation

Implement mutual TLS (mTLS) with grpc::SslServerCredentials and client certificate validation. Add an authentication interceptor and role-based access control.

CVE-2026-48693

Symlink Attack via Predictable /tmp Statistics File Path

HIGH CVSS: 7.0 (High)

Vulnerability Type:	CWE-59 (Improper Link Resolution), CWE-377 (Insecure Temporary File)
Attack Vector:	Local
Affected File:	src/fastnetmon.cpp line 159; src/fastnetmon_logic.cpp lines 2184-2196
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

FastNetMon writes statistics to the predictable path /tmp/fastnetmon.dat (and /tmp/fastnetmon_ipv6.dat) without symlink protection, using std::ofstream with std::ios::trunc which follows symlinks. The daemon runs with umask(0), making all created files world-writable. Additional bugs compound this: the chmod call uses the wrong variable name (always cli_stats_file_path instead of the file_path parameter), and the permissions include S_IROTH despite the code comment claiming '660'.

Vulnerable Code

```
// Predictable path
std::string cli_stats_file_path = "/tmp/fastnetmon.dat";

// Follows symlinks, truncates target
screen_data_file.open(file_path.c_str(),
std::ios::trunc);

// BUG: Wrong variable + wrong permissions
chmod(cli_stats_file_path.c_str(),
S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH);

umask(0); // World-writable files
```

Impact

Local arbitrary file overwrite as root. An attacker creates a symlink from /tmp/fastnetmon.dat to a target file (e.g., /etc/cron.d/backdoor). FastNetMon truncates and writes to the target through the symlink. Combined with umask(0), created files are world-writable.

Remediation

Move stats files to /var/lib/fastnetmon/ with restrictive permissions. Use O_NOFOLLOW flag. Set umask(0077) instead of umask(0).

CVE-2026-48694

Router Configuration Injection via Juniper NETCONF Plugin

HIGH CVSS: 8.1 (High)

Vulnerability Type:	CWE-77 (Command Injection)
Attack Vector:	Indirect remote (via attack notification pipeline)
Affected File:	src/juniper_plugin/fastnetmon_juniper.php, lines 69 and 90
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The $IP_ATTACK variable is directly interpolated into Juniper router configuration commands sent via NETCONF without any sanitization or validation. At line 69 (ban): load_set_configuration("set routing-options static route {$IP_ATTACK} community 65535:666 discard"). At line 90 (unban): load_set_configuration("delete routing-options static route {$IP_ATTACK}/32"). If $IP_ATTACK contains newline characters followed by Junos CLI commands, arbitrary router configuration changes can be injected.

Vulnerable Code

```
// Line 69 (ban action):
$conn->load_set_configuration(
"set routing-options static route " .
"{$IP_ATTACK} community 65535:666 discard");

// No validation of $IP_ATTACK format
```

Impact

Full Juniper router compromise. An attacker who can influence the IP address string can inject arbitrary configuration: backdoor user accounts, disable firewall filters, modify routing tables, enable SNMP with community strings, or hijack BGP sessions.

Remediation

Validate $IP_ATTACK matches a strict IP address regex (/^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$/) before interpolation into configuration commands.

CVE-2026-48695

OS Command Injection in MikroTik Plugin Logging Function

HIGH CVSS: 8.1 (High)

Vulnerability Type:	CWE-78 (OS Command Injection)
Attack Vector:	Indirect remote (via attack notification pipeline)
Affected File:	src/mikrotik_plugin/fastnetmon_mikrotik.php, lines 105-109
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

Identical vulnerability to CVE-2026-48687. The MikroTik plugin's _log() function uses the same exec("echo ... $msg ...") pattern with unsanitized attack data passed through argv[]. Additionally, this file contains hardcoded router credentials at lines 31-33 (username: 'api', password: 'api123'), which compound the security impact.

Vulnerable Code

```
function _log( $msg ) {
$FILE_LOG_TMP =
"/tmp/fastnetmon_api_mikrotik.log";
exec( "echo `date` \"- [FASTNETMON] - "
. $msg . " \" >> " . $FILE_LOG_TMP );
}

// Also: hardcoded credentials at lines 31-33
$user = "api"; $pass = "api123";
```

Impact

Remote code execution as the FastNetMon process user, identical to CVE-2026-48687. The hardcoded credentials provide additional lateral movement to the MikroTik router.

Remediation

Replace exec() with file_put_contents() for logging, use escapeshellarg() on all interpolated values, and externalize credentials to a configuration file.

CVE-2026-48696

Stack Buffer Overflow in ExaBGP Action Handler

MEDIUM CVSS: 6.0 (Medium)

Vulnerability Type:	CWE-120 (Buffer Copy without Checking Size), CWE-676 (Dangerous Function)
Attack Vector:	Local (configuration file)
Affected File:	src/actions/exabgp_action.cpp, lines 21-31
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The exabgp_prefix_ban_manage() function uses sprintf() to format a BGP announce/withdraw command into a fixed 256-byte stack buffer (bgp_message[256]). The format string 'announce route %s next-hop %s community %s\n' consumes ~40 bytes of fixed text, leaving ~216 bytes for three variable-length strings. The exabgp_community config value is read from the configuration file with no length validation. A community string with 30+ communities overflows the buffer.

Vulnerable Code

```
char bgp_message[256];
sprintf(bgp_message,
"announce route %s next-hop %s community %s\n",
prefix.c_str(), next_hop.c_str(),
community.c_str()); // OVERFLOW
// 30 communities: ~330 bytes > 256
```

Impact

Stack buffer overflow via malicious configuration. With 30 communities (each ~11 characters), the formatted string is ~370 bytes, causing a 114-byte overflow. No -fstack-protector in the build makes this directly exploitable for code execution.

Remediation

Replace sprintf with snprintf(bgp_message, sizeof(bgp_message), ...) or use std::string formatting.

CVE-2026-48697

Missing TLS Certificate Verification on HTTPS Connections

HIGH CVSS: 7.4 (High)

Vulnerability Type:	CWE-295 (Improper Certificate Validation)
Attack Vector:	Remote (network-adjacent, MITM)
Affected File:	src/fast_library.cpp, lines 1639-1670
Affected Software:	FastNetMon Community Edition <= 1.2.9
Discovered By:	Lorikeet Security (lorikeetsecurity.com)

Description

The execute_web_request_secure() function creates a Boost.Asio SSL context, calls set_default_verify_paths() to load CA certificates, and sets SNI via SSL_set_tlsext_host_name(), but never calls set_verify_mode(boost::asio::ssl::verify_peer). Without this call, OpenSSL accepts any certificate including self-signed ones, making all outbound HTTPS connections vulnerable to man-in-the-middle attacks. This affects telemetry sent every 3600 seconds to community-stats.fastnetmon.com.

Vulnerable Code

```
boost::asio::ssl::context ctx{
boost::asio::ssl::context::tls_client};
ctx.set_default_verify_paths(); // Loads CAs

// MISSING:
// ctx.set_verify_mode(
// boost::asio::ssl::verify_peer);

SSL_set_tlsext_host_name(
stream.native_handle(), host.c_str());
// Sets SNI but never verifies certificate
```

Impact

All outbound HTTPS connections accept any certificate. Telemetry data (CPU model, kernel version, traffic stats, software version, enabled features) sent every hour can be intercepted or redirected. An attacker on the network path can perform a full man-in-the-middle attack.

Remediation

Add ctx.set_verify_mode(boost::asio::ssl::verify_peer); and ctx.set_verify_callback(boost::asio::ssl::rfc2818_verification(host)); before the TLS handshake.

Disclosure Timeline

Date	Event
2026-04-25	Vulnerabilities discovered during comprehensive source code audit
2026-04-25	CVE IDs requested from MITRE
2026-04-25	Vendor notified at pavel.odintsov@gmail.com per SECURITY.md
2026-05-XX	16 CVE IDs assigned by MITRE (CVE-2026-48682 through CVE-2026-48697)
TBD	Vendor response
TBD	Fix release
2026-05-23	Responsible disclosure report published

About Lorikeet Security

Lorikeet Security is a security research organization specializing in vulnerability discovery and responsible disclosure for open-source infrastructure software. This disclosure is published in good faith to improve the security of open-source network infrastructure. All vulnerabilities were identified through source code review. No systems were exploited or accessed without authorization.

Contact: lorikeetsecurity.com