CVE-2026-48696: `sprintf` Into a 256-Byte Stack Buffer in FastNetMon's ExaBGP Action Handler

Lorikeet Security Team May 23, 2026 8 min read

In This Article

LORIKEET SECURITY

CVE	CVE-2026-48696
CVSS	**6.0 (Medium)**
CWE	CWE-120 (Buffer Copy without Checking Size), CWE-676 (Use of Potentially Dangerous Function)
AFFECTED	FastNetMon Community Edition <= 1.2.9
COMPONENT	src/actions/exabgp_action.cpp, function `exabgp_prefix_ban_manage()`, lines 21-31
ATTACK VECTOR	Local (configuration file)
DISCOVERED BY	Lorikeet Security

FastNetMon talks to ExaBGP — the Python-based BGP speaker that many operators use as a programmable BGP route injector — via a simple command-line protocol. The function `exabgp_prefix_ban_manage()` in `src/actions/exabgp_action.cpp` formats an ExaBGP announce-or-withdraw command into a fixed 256-byte stack buffer using `sprintf()` and then writes the result to ExaBGP's input pipe.

The format string is `"announce route %s next-hop %s community %s\n"`, which consumes about 40 bytes of fixed text, leaving roughly 216 bytes for the three variable-length string substitutions (prefix, next-hop, community list). The prefix and next-hop are bounded (an IPv4 prefix is at most 18 characters; an IPv6 next-hop is at most 39 characters), but the community list is unbounded. The `exabgp_community` configuration value is read from `fastnetmon.conf` with no length validation.

LORIKEET SECURITY

The vulnerable code

```
// src/actions/exabgp_action.cpp, lines 21-31
void exabgp_prefix_ban_manage(const std::string& prefix,
                              const std::string& next_hop,
                              const std::string& community) {
    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
    // ...
}
```

This is the canonical `sprintf`-into-a-fixed-buffer bug. Every "C programming pitfalls" article from the past 35 years lists it. The C standard library has provided `snprintf()` since C99 (1999) precisely to avoid this; using `sprintf()` in 2026 in a security-relevant code path is a code smell that compiler warning configurations should flag with `-Wformat-security` (which the default FastNetMon CMake build does not enable).

LORIKEET SECURITY

needs the ability to either modify the configuration file or coerce an operator into supplying a long community list. But unlike CVE-2026-48690 (which fires at startup), this bug doesn't fire until FastNetMon actually invokes ExaBGP — which happens during an active DDoS mitigation.

The implication: **the overflow can be timed**. An attacker who can briefly modify the configuration file just before a DDoS attack lands can choose precisely when the corrupted-stack code path executes. This is useful in scenarios where the attacker is coordinating a multi-stage attack: caus traffic, FastNetMon attempts to push an ExaBGP announcement, the announcement code overflows, the daemon crashes (or, with the right payload, the daemon's stack is hijacked).

The attack scenarios:

- **Malicious or compromised configuration management.** Same shape as CVE-2026-48690. An operator's Ansible playbook templates the community list from a database query; the query returns more entries than expected; the resulting config overflows on first BGP announcement
- **Tutorial transcription errors.** A blog post describes setting up granular community-based blackholing with 20+ communities. An operator copies it verbatim; the community list grows; the bug becomes live.
- **Operator who wants more communities.** An operator legitimately needs to attach many communities (e.g., multiple upstream providers each requiring their own community for blackhole signaling) and configures a long list. The configuration is valid; the daemon's code path is not.

The "Medium" CVSS rating reflects the local attack vector. In practice, the bug is more dangerous than that score suggests because the trigger is operator-supplied configuration that an operator might reasonably set in production.

LORIKEET SECURITY

Same exploitation story as CVE-2026-48696: stack overflow, no `-fstack-protector`, no `-fPIE`, attacker-controlled bytes overwriting the saved return address. The overflowed bytes here come from the community string, which is the operator-supplied configuration value — so the bytes are not fully attacker-chosen unless the attacker is also the operator. But for an attacker who can write the configuration file (because they have local shell access on the FastNetMon host), they choose the community string and therefore the overflow bytes.

The chain is: write malicious community string to `fastnetmon.conf` → wait for an attack (or generate one) → FastNetMon invokes `exabgp_prefix_ban_manage` with the malicious community → sprintf overflows the stack → saved return address now points to attacker-chosen address → code execution at function return.

How a fix should look

```
void exabgp_prefix_ban_manage(const std::string& prefix,
                              const std::string& next_hop,
                              const std::string& community) {
    // 1. Use std::string, no fixed buffer.
    std::string bgp_message = "announce route " + prefix
                            + " next-hop " + next_hop
                            + " community " + community + "\n";

    // 2. (Defensive) Reject communities that are obviously pathological.
    if (community.size() > 1024) {
        log_error("exabgp community list too long: %zu bytes", community.size());
        return;
    }
```

LORIKEET SECURITY

fixed-size `char[256]` in a 2026 C++ codebase except for narrowly performance-constrained inner loops; this is not one of those.

If you must use a fixed-size buffer, use `snprintf()` with a size argument and check the return value to detect truncation:

```
char bgp_message[1024];
int n = snprintf(bgp_message, sizeof(bgp_message),
                 "announce route %s next-hop %s community %s\n",
                 prefix.c_str(), next_hop.c_str(), community.c_str());
if (n < 0 || (size_t)n >= sizeof(bgp_message)) {
    log_error("ExaBGP message would be truncated: %d >= %zu",
              n, sizeof(bgp_message));
    return;
}
```

Either form is structurally safe. The original code is not. There is no defensible reason to retain it.

Compensating controls

- **Audit your `exabgp_community` configuration value.** Count the communities you've configured. If you have more than 15 (each community is ~11 characters), you are approaching the overflow threshold. Most operators only need 1–5 communities for blackhole signaling; long lists usually indicate misconfiguration or a misunderstood requirement.
- **Lock down `/etc/fastnetmon.conf` permissions.** Same advice as CVE-2026-48690.

LORIKEET SECURITY

- **Enable** `-Wformat-security` in your CMakeLists.txt. The compiler will flag this `sprintf` call as a security issue at build time. Worth fixing once at the project level so all future calls get checked.
- **If you don't use ExaBGP, disable the action.** FastNetMon supports multiple BGP-speaker backends (ExaBGP, GoBGP). If you use GoBGP, the ExaBGP code path is dead code; configure FastNetMon to use only GoBGP and the buggy function is never reached.

The pattern: don't write to fixed-size buffers, ever

Three CVEs in this disclosure series (this one, CVE-2026-48686, and CVE-2026-48690) come from the same root cause: a fixed-size buffer paired with a length value that wasn't checked against the buffer's actual capacity. Every one of them is preventable by the same set of habits:

1. **Use bounded standard-library APIs.** `snprintf` instead of `sprintf`. `strlcpy` instead of `strcpy`. `std::string` instead of `char[256]`. `std::vector<T>` instead of `T arr[N]`.
2. **Check destination size at every copy.** If you must copy, the copy length must be the minimum of (source bytes available, destination space remaining). The destination space remaining is always knowable; use it.
3. **Enable compiler warnings that catch these patterns at build time.** `-Wformat-security` `-Wstringop-overflow`, `-Walloca-larger-than`. These cost nothing and catch real bugs.
4. **Enable FORTIFY at build time.** `-D_FORTIFY_SOURCE=2` or `=3` instruments many of the

These are not advanced techniques. They are baseline. A 2026 codebase that ships any of the three bugs in this CVE series under default build flags is operating at a 1995 hygiene level.

Disclosure timeline

Date	Event
2026-04-25	Vulnerability identified during Lorikeet Security source code audit of FastNetMon Community Edition 1.2.9
2026-04-25	CVE ID requested from MITRE
2026-04-25	Vendor (Pavel Odintsov / FastNetMon LTD) notified at the contact published in `SECURITY.md`
2026-05-22	CVE-2026-48696 assigned by MITRE
TBD	Vendor response
TBD	Fix release
2026-05-23	Lorikeet Security publishes responsible disclosure report

LORIKEET SECURITY

19 views

Share

Lorikeet Security Team

PENETRATION TESTING & CYBERSECURITY CONSULTING

Lorikeet Security helps modern engineering teams ship safer software. Our work spans web applications, APIs, cloud infrastructure, and AI-generated codebases — and everything we publish here comes from patterns we see in real client engagements.

Practical security, written by practitioners.

LORIKEET SECURITY

You Might Also Like

Vulnerability Research

CVE-2026-22769: A Hard-Coded Tomcat Password Gave UNC6201 Root on Dell RecoverPoint for Two Years

13 min read

Vulnerability Research

CVE-2026-20127: A One-Byte Bug Cracked the Cisco SD-WAN Control Plane. Here Is Exactly How.

12 min read

Vulnerability Research

ESXicape: VM Escape Attacks, VSOCKpuppet, and Why Hypervisor Security Is Under Siege

16 min read

← PREVIOUS

CVE-2026-48697: FastNetMon Sets Up TLS But Never Asks It to Verify Anything

NEXT →

LORIKEET SECURITY

Penetration Testing

Managed Services

Company

Free Tools

Partners

Industries

Locations

Compliance Testing

Knowledge Base
Portal guides, FAQs & how-to articles

Developer Docs
API reference, webhooks & integrations

Stay Updated

LORIKEET SECURITY

+1 (929) 577-3213
New York, NY

+1 (888) 652-6479
Toll-Free Nationwide

Cookie Settings

Terms of Service

Privacy Policy