In Wild Critical Buffer Overflow Vulnerability in Solaris Can Allow Remote Takeover — CVE-2020-14871

fireeye.com/blog/threat-research/2020/11/critical-buffer-overflow-vulnerability-in-solaris-can-allow-remote-takeover.html

Breadcrumb

Threat Research

Jacob Thompson

Nov 04, 2020

5 mins read

Threat Research

FireEye Mandiant has been investigating compromised Oracle Solaris machines in customer environments. During our investigations, we discovered an exploit tool on a customer's system and analyzed it to see how it was attacking their Solaris environment. The FLARE team's Offensive Task Force analyzed the exploit to determine how it worked, reproduced the vulnerability on different versions of Solaris, and then reported it to Oracle. In this blog post we present a description of the vulnerability, offer a quick way to test whether a system may be vulnerable, and suggest mitigations and workarounds. Mandiant experts from the FLARE team will provide more information on this vulnerability and how it was used by UNC1945 during a Nov. 12 webinar. Register today and start preparing questions, because we will be fielding them from the audience at the end of the session.

Vulnerability Discovery

The security vulnerability occurs in the Pluggable Authentication Modules (PAM) library. PAM enables a Solaris application to authenticate users while allowing the system administrator to configure authentication parameters (e.g., password complexity and expiration) in one location that is consistently enforced by all applications.

The actual vulnerability is a classic stack-based buffer overflow located in the PAM parse_user_name function. An abbreviated version of this function is shown in Figure 1.

```
static int
parse_user_name(char *user_input, char **ret_username)
{
        register char *ptr;
        register int index = 0;
        char username[PAM_MAX_RESP_SIZE];
    /* ... */

        ptr = user_input;
    /* ... */
         /*
         * username will be the first string we get from user_input
         * - we skip leading whitespaces and ignore trailing whitespaces
         */
        while (*ptr != '\0') {
            if ((*ptr == ' ') || (*ptr == '\t'))
                    break;
            else {
                    username[index] = *ptr;
                    index++;
                    ptr++;
            }
        }
         /* ret_username will be freed in pam_get_user(). */
        if ((*ret_username = malloc(index + 1)) == NULL)
            return (PAM_BUF_ERR);
        (void) strcpy(*ret_username, username);
        return (PAM_SUCCESS);
}
```

Figure 1: The parse_user_name function has a stack-based buffer overflow vulnerability

The vulnerability arises whenever a username longer than PAM_MAX_RESP_SIZE (512 bytes) is passed to parse_user_name. The vulnerability has likely existed for decades, and one possible reason is that it is only exploitable if an application does not already limit usernames to a smaller length before passing them to PAM. One situation where network-facing software does not always limit the username length arises in the SSH server, and this is the exploit vector used by the tool that we discovered.

SSH Keyboard-Interactive authentication is a “passthrough” authentication mechanism where the SSH protocol relays prompts and responses between the server’s PAM libraries and the client. It was designed to support custom forms of authentication such as two-factor without modifying the SSH protocol. By manipulating SSH client settings to force Keyboard-Interactive authentication to prompt for the username rather than sending it through normal means, an attacker can also pass unlimited input to the PAM parse_user_name function.

Proof of Concept Exploit

In order to quickly test different versions of Solaris to see if they may be vulnerable, we developed a proof of concept exploit to trigger the overflow and crash the SSH server. The standard OpenSSH client offers all the options needed to trigger the vulnerability (Figure 2).

Figure 2: A server can be quickly tested to see if it is vulnerable over SSH

The indication that the server is vulnerable is that the SSH client prints “Authentication failed;” a non-vulnerable PAM library causes the SSH server to repeatedly prompt for a username if it receives one that is too long. The overflow in the PAM library also causes the SSH server to crash, as shown in Figure 3. The operating system writes a crash dump to /core if the SSH server crashes with no debugger attached. In fact, if a /core file exists on a Solaris machine and the file command reports that it is from sshd, those are indicators consistent with this vulnerability having been exploited.

Figure 3: The SSH server crashes in the parse_user_name function

Vulnerable Operating Systems

- Solaris 9 (some releases)
- Solaris 10 (all releases)
- Solaris 11.0

  While the parse_user_name function remains vulnerable in unpatched Solaris 11.1 and later, unrelated changes to the PAM library truncate the username before the vulnerable function receives it, rendering the issue non-exploitable via SSH. If the parse_user_name function were reachable in another context, then the vulnerability could become exploitable.
- Illumos (OpenIndiana 2020.04)

Mitigations and Workaround

A patch from Oracle for Solaris 10 and 11 is described in the October 2020 Critical Patch Update.

Because Solaris 9 is no longer supported, Oracle has not released a patch. For Solaris 9, as well as Solaris 10 or 11 systems where patching is inconvenient, we recommend editing the /etc/ssh/sshd_config file to add the lines ChallengeResponseAuthentication no and KbdInteractiveAuthentication no and restart the SSH server. While this removes the opportunity to exploit the vulnerability using SSH Keyboard-Interactive authentication, there may be other ways to attack the parse_user_name function and we recommend using this workaround only as a stopgap until Solaris 9 systems can be upgraded, or the October patch can be accessed and installed for supported Solaris versions.

Acknowledgements

Jeffrey Martin of Rapid7 contributed to the testing of this vulnerability.