

Audit Report: FreeRADIUS

August 23, 2005

Package: freeradius
Version: 1.0.2-5.5
Product: SL 9.3
Date: 2005-08-23
Reason: most wide-spread open-source RADIUS server
Scope: - authentication code
- module code (exec, unix, ldap, sql, realm, expr)
- checkrad.pl
- radlog() function
Reviewer: Thomas Biege <thomas@suse.de>
Review-Version: 0.1
Bug-ID: 104195
References: none

1 Introduction

none

2 Audit Environment

- SUSE LINUX 9.3 x86
- freeradius-1.0.2-5.5
- pam-0.78-8
- mysql-4.1.10a-3.2
- server: idoru.ashpool.org
- client: spiral.ashpool.org
- database server: idoru.ashpool.org

3 Methodology

The RADIUS server was setup on host *Idoru* with authentication *Local*, *System*, *PAM*, and *SQL*. To be able to test a wide-range of common pitfalls in programming of such a complex software a "fuzzer" was written and tested successively against different kind of RADIUS extensions and configurations (Local, System, PAM, SQL).

Additionally some parts of the code were inspected line-by-line.

4 Fuzzer Results

The fuzzer sends about 5 million packets which include various common attacks in Perl, SQL, C, HTML, and LDAP. It does also special fuzzing of the User-Name AVP (attribute value pair) and of the realm support.

5 Source-Code Review

5.1 File: `exec.c`

- `radius_exec_program`: line 201: The number of filedescriptors closed may not be large enough. Better use:

```
if((maxopenfd = sysconf(_SC_OPEN_MAX)) < 0)
maxopenfd = OPEN_MAX_LINUX;
```

- `radius_exec_program`: line 117-233: This is a minor issue but it may be worth correcting. The strings used as command-line option for the code to execute should be sanitized. For example they should not include '-' or '-' to add unwanted command-line options This is mostly stopped because the code is executed after the authentication and so at least for User-Name it has to be defined by the admin.
- `radius_exec_program`: line 212-231: The maximum size (MAX_ENVP) of envp is not tested in the for-loop while stuffing the AVPs in the environment which may cause **writing out-of-bounds**.

5.2 File: `rlm_ldap.c`

- `ldap_escape_func`: The function `ldap_escape_func()` should filter much more LDAP-specific characters like: || (pipe), (, =, cn=, etc. This will avoid cross-site-scripting attacks as they happen in SQL etc..
- Will an unavailable LDAP server block the RADIUS daemon?

5.3 File: sql.c

- Ok.

5.4 File: rlm_sql.c

- `sql_escape_func`: The function `sql_escape_func()` does convert all input up to an upper limit. By this a **username may get truncated**. This leads to a subtle problem. Let's assume user John Doe wants to get access to the account of John Doe Dorian. Both users are authenticated by using SQL. John Doe logs into the system using the username `johndloedorian`. The RADIUS server looks up this string in its user file, sees `Auth-Type := SQL` and calls its `rlm_sql` module. Then `sql_escape_func()` truncates this string to `johndoe`, receives the password from the database and compares it to the password given by John Doe. This password is John Doe's original password. And due to the truncation John Doe's information was received, the password matches and John Doe gets access to John Doe Dorian's account.¹ (This needs to be tested more carefully by the auditor due to long code paths that a username travels through which all have their own limits and may possibly avoid this attack.)
- `rlm_sql_authorize`: line 747: Here assumptions are made that the string should be long enough to hold the full escaped username. It is stated:

```
/* sqlusername holds the sql escaped username. The original
 * username is at most MAX_STRING_LEN chars long and
 * *sql_escape_string doubles its length in the worst case.
 * Throw in an extra 10 to account for trailing NULs and to
 * have a safety margin. */
```

But reading the `sql_set_user()` function it shows up that no escaping will be done at all. This doesn't hurt but was just a wrong assumption. On the other side `sql_escape_func()` doubles the size in the worst case (MIME-encoding for non-printable chars) but this does not affect `sqlusername`. So, no problem either... just a note.

5.5 File: xlat.c

- `valuepair2str`: Ok.
- `decode_attribute`: **Possible write beyond buffer bounds in `attrname[256]`**. The buffer is 256 bytes in size but the length is not checked while writing to it in a while-loop.

¹This can also happen due to the limit of the query-string.

- radius_xlat: line 725: DNS responses are not escaped and may include SQL commands.

5.6 File: sql_unixodbc.c.c

- sql_error: line 362-363: **One byte buffer overflow:**

```
result = (char*)rad_malloc(strlen(state)+1+strlen(error));  
sprintf(result, "%s %s", state, error);
```

Two extra bytes (blank + trailing zero) have to be allocated for result.

5.7 File: rlm_exec.c

- Ok.

5.8 File: rlm_unix.c

- Ok.

5.9 File: rlm_realm.c

- check_for_realm: line 209: Better add a trailing zero to vp->strvalue
- check_for_realm: line 210: If the trailing zero is missing strlen(vp->strvalue) will return a wrong value.

5.10 File: rlm_expr.c

- Ok.

5.11 File: session.c

- rad_check_ts: line 202: The number of filedescriptors closed may not be large enough. Better use:

```
if((maxopenfd = sysconf(_SC_OPEN_MAX)) < 0)  
maxopenfd = OPEN_MAX_LINUX;
```

- radius_exec_program: line 224: This is a minor issue but it may be worth correcting. The strings used as command-line option for the code to execute should be sanitized. For example they should not include '-' or '-' to add unwanted command-line options. Unlikely.

5.12 File: checkrad.pl

- Ok.

5.13 File: log.c

- vradlog: line 133: The `vsnprintf()` is dangerous if `len` becomes `sizeof(buffer)`. This results in a **buffer overflow** due to an **integer wrap** of the second argument. Better check this case before. Very unlikely because `len` is not under the control of the attacker.
- vradlog: line 135: Checking the buffer bounds does not make sense **after the overflow already happened**. Check earlier.
- vradlog: line 150: The `strcat(buffer, "\n")` may overwrite the last and trailing zero if the buffer is full. Additionally a `'\0'` has to be added manually after the `'\n'` by the programmer. **Off-by-One**

5.14 File: auth.c

- rad_auth_log: line 131,143,145: Add a trailing zero.
- rad_authenticate: line 678: Is it guaranteed that `auth_item->strvalue` has a trailing zero? ²

5.15 File: radius.c

- rad_recv: Ok.

6 Summary

FreeRADIUS tries hardly to force the rules of the RADIUS RFC to avoid processing malformed packages. The original code seems to be well written but attention should be brought to the various modules written by different programmers.

7 The RADIUS Fuzzer

The tool is available via CVS by using the following parameters: `"CVS.RSH=ssh"`, `"CVSROOT=thomas@wotan.suse.de:/suse/thomas/Projekte/repository/"` and later at <http://www.suse.de/thomas/>.

²Quickly checking `pairread()` from `lib/valuepair.c` `getthing()` from `lib/token.c` shows that it is not guaranteed.

8 Status

Sebastian Krahmer will do some additional source-code auditing.

The full `src/modules` tree needs an audit and the decoding functions in `radius.c` need more attention. The usage of SSL/TLS (EAP-TLS, LDAP) needs to be verified with an real-life attack on the SSL-communication.

The truncation attack mentioned for the SQL module needs to be verified further.

References

[1] Bugzilla:

https://bugzilla.novell.com/show_bug.cgi?id=104195