IOActive Security Advisory

Title	Multiple Vulnerabilities in Fwknop
Severity	High
Discovered by	Fernando Arnaboldi
CVEs	CVE-2012-4434, CVE-2012-4435, CVE-2012-4436

Introduction

Fwknop stands for the "FireWall KNock OPerator", and implements an authorization scheme called Single Packet Authorization (SPA). This method of authorization is based around a default-drop packet filter and libpcap. A server might appear to have no open ports available, but it could still grant access to certain services if authorized fwknop packets are received. This service is commonly used on exposed systems which need to diminish the attack surface of their services.

Fwknop contains several vulnerabilities, of which the most critical could allow remote authenticated attackers to take advantage of flaws to execute code and/or to produce denials of service. In addition to that, certain local flaws could also be triggered to execute code.

1) Remote Stack Overflow in acc_check_port_access

Affected Products

The following versions of `fwknop`: `2.0.2, 2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2, 2.0.0-rc1`.

Impact

Denial of Service / Remote Code Execution

Severity

High

Technical Details

There is a buffer overflow in the `fwknop` server when processing an authenticated request to open up a port. This overflow occurs in the file `server/access.c`, when the function

`acc_check_port_access` assigns the variable '`buf`' with the contents of '`start`' without checking the length:

```
int acc_check_port_access(acc_stanza_t *acc, char *port_str)
{
    int             res     = 1;
    char            buf[32];
    char           *ndx, *start;

    acc_port_list_t *o_pl   = acc->oport_list;
    acc_port_list_t *r_pl   = acc->rport_list;
    acc_port_list_t *in_pl  = NULL;
    start = port_str;

    for(ndx = start; *ndx; ndx++){
        if(*ndx == ',') {
            strlcpy(buf, start, (ndx-start)+1);
            add_port_list_ent(&in_pl, buf);
            start = ndx+1;}}

    strlcpy(buf, start, (ndx-start)+1);
    …
```

To exploit this issue a modified version of the `fwknop` client was used against a default `fwknop` server running on a Linux version of Debian.

Remediation

Upgrading to Release `2.0.3` of `fwknop` removes this stack overflow vulnerability.

CVE

More information can be found about this vulnerability at the following CVE location:

- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4435

2) Remote Stack Overflow in parse_proto_and_port

Affected Products

The following versions of `fwknop`: `2.0.2, 2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2, 2.0.0-rc1`.

Impact

Denial of Service / Remote Code Execution

Technical Details

There is a buffer overflow in the `fwknop` server when processing an authenticated request to open up a port. This overflow occurs in the file `server/access.c`, when the size of the variable '`pstr`' is not checked before assigning the value into '`proto_str`' for the function `parse_proto_and_port`:

```
static int parse_proto_and_port(char *pstr, int *proto, int
*port)
{
    char    *ndx;
    char    proto_str[32];

    if((ndx = strchr(pstr, '/')) == NULL) {
        log_msg(LOG_ERR,"Parse error on access port entry: %s",
pstr);
        return(-1);}

    strlcpy(proto_str, pstr,  (ndx - pstr)+1);
    …
```

To exploit this issue a modified version of the `fwknop` client was used against a default `fwknop` server running on a Linux version of Debian.

Remediation

Upgrading to Release `2.0.3` of `fwknop` removes this stack overflow vulnerability.

CVE

More information can be found about this vulnerability at the following CVE location:

- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4435

3) Local Stack Overflow in run_last_args

Affected Products

The following versions of `fwknop`: `2.0.2, 2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2, 2.0.0-rc1.`

Impact

Potential Code Execution

Technical Details

There is a buffer overflow in `fwknop` when using the parameter `--last-cmd` using a history file with malicious content. This overflow occurs in the file `client/access.c`, in the function `run_last_args()` which contains a loop that reads all the previous parameters stored in the history file `.fwknop.run`. If the contents of that file contains more than 200 parameters, there will be buffer overflow in the `argv_new[]` array when trying to assign the value 210.

```
static void
run_last_args(fko_cli_options_t *options)
{
    FILE           *args_file_ptr = NULL;

    int             current_arg_ctr = 0;
    int             argc_new = 0;
    int             i = 0;

    char            args_save_file[MAX_PATH_LEN] = {0};
    char            args_str[MAX_LINE_LEN] = {0};
    char            arg_tmp[MAX_LINE_LEN]  = {0};
    char           *argv_new[200];  /* should be way more than
enough */


#ifdef WIN32
    /* Not sure what the right thing is here on Win32, just
return
     * for now.
    */
    return;
#endif

    if (get_save_file(args_save_file))
    {
        if ((args_file_ptr = fopen(args_save_file, "r")) == NULL)
        {
            fprintf(stderr, "Could not open args file: %s\n",
                args_save_file);
            exit(EXIT_FAILURE);
        }
```

```
        if ((fgets(args_str, MAX_LINE_LEN, args_file_ptr)) !=
NULL)
        {
            args_str[MAX_LINE_LEN-1] = '\0';
            if (options->verbose)
                printf("Executing: %s\n", args_str);
            for (i=0; i < (int)strlen(args_str); i++)
            {
                if (!isspace(args_str[i]))
                {
                    arg_tmp[current_arg_ctr] = args_str[i];
                    current_arg_ctr++;
                }
                else
                {
                    arg_tmp[current_arg_ctr] = '\0';
                    argv_new[argc_new] =
malloc(strlen(arg_tmp)+1);
                    if (argv_new[argc_new] == NULL)
                    {
                        fprintf(stderr, "malloc failure for cmd
line arg.\n");
                        exit(EXIT_FAILURE);
                    }
                    strlcpy(argv_new[argc_new], arg_tmp,
strlen(arg_tmp)+1);
                    current_arg_ctr = 0;
                    argc_new++;
                }
            }
        }
    …
```

To exploit this issue a modified version of the `fwknop` client was used against a default `fwknop` server running on a Linux version of Debian.

Remediation

Upgrading to Release `2.0.3` of `fwknop` removes this stack overflow vulnerability.

CVE

More information can be found about this vulnerability at the following CVE location:

- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4436

4) Local Stack Overflows when Reading access.conf

Affected Products

The following versions of `fwknop`: `2.0.2` (other versions were not checked).

Impact

Potential Code Execution

Technical Details

When the `fwknop` server reads the `access.conf` file, there are no controls when using long parameters on `SOURCE`, `OPEN_PORTS` and `RESTRICT_PORTS`, and it's possible to crash the server and cause different buffer overflows.

Remediation

Upgrading to Release 2.0.3 of `fwknop` removes this stack overflow vulnerability.

5) Fwknop Global Configuration Readable

Affected Products

The following versions of `fwknop`: `2.0.2 (partially affected), 2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2, 2.0.0-rc1`

Impact

Passwords confidentiality might be compromised.

Technical Details

The file `access.conf` stores plain text passwords used by the `fwknop` server. When the source tar archives are uncompressed, the default permissions of `server/access.conf` are set to `664` for versions `2.0.2, 2.0.1, 2.0.0-rc3` and `2.0.0-rc2`; permissions are set to `644` for versions `2.0.0, 2.0.0-rc5, 2.0.0-rc4` and `2.0.0-rc1`. This means that any user with access to the `server/` directory will be able to access this information.

Afterwards, when the `fwknop` server is installed using `make install`, the file `access.conf` is moved to `/usr/local/etc/fwknop`. The permissions are set to `644` in versions `2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2` and `2.0.0-rc1` which means that read access is still possible after installation. Version `2.0.2` uses permissions `600`, which do not allow read access to everyone.

Remediation

Upgrading to Release `2.0.3` of `fwknop` removes this vulnerability.

6) Fwknop Configuration File Permissions are Set to the User's Umask Value

Affected Products

The following versions of `fwknop`: `2.0.2, 2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2, 2.0.0-rc1`

Impact

An attacker might be able to modify the contents of the configuration file.

Technical Details

The `fwknop` client stores its configuration in `~/.fwknoprc` when being executed for the first time. The conventional policy is to make configuration files only writable for the owner (i.e., `~/.bashrc`). However the permissions for `~/.fwknoprc` are set using the `umask` value of the logged user. Since the creation of this file relies on the `umask` value, this file might be created with world writable permissions. If this happens, an attacker would be able to modify the contents of this file and alter the configuration parameters.

This behavior was observed in `client/config_init.c`:

```
static int
create_fwknoprc(const char *rcfile)
{
    FILE    *rc;

    fprintf(stderr, "Creating initial rc file: %s.\n", rcfile);

    if ((rc = fopen(rcfile, "w")) == NULL)
    {
        fprintf(stderr, "Unable to create rc file: %s: %s\n",
            rcfile, strerror(errno));
        return(-1);
    }
    …
```

Remediation

Upgrading to Release `2.0.3` of `fwknop` removes this vulnerability.

7) Fwknop History File Permissions are Set to the User's Umask Value

Affected Products

The following versions of `fwknop`: `2.0.2, 2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2, 2.0.0-rc1`

Impact

The last `fwknop` command might be read and modified by an attacker with local access to the file `~/.fwknop.run`. Since the file contents might be used with the fwknop parameter `--last-cmd`, the original user could inadvertently execute unexpected content that could affect either the local host or a remote server. If this issue is used in conjuntion with the vulnerability #3 exposed in this document, risk is even higher.

Technical Details

The `fwknop` client stores the last command executed in `~/.fwknop.run` when using the function `save_args()`. The conventional policy is to make history files only readable/writable for the owner (i.e., `~/.bash_history`). However, the permissions for `~/.fwknop.run` are set using the `umask` value of the logged user. Since the creation of this file relies on the `umask` value, this file might be created with world writable permissions. If this happens and an attacker is able to modify the contents of this file, the real owner might inadvertently execute dangerous content when using the parameter '`--last-cmd`'.

This behavior was observed in `client/fwknop.c`:

```
static void save_args(int argc, char **argv)
{
    char args_save_file[MAX_PATH_LEN];
    char args_str[MAX_LINE_LEN] = "";
    FILE *args_file_ptr = NULL;
    int i = 0, args_str_len = 0;

    …

    if (get_save_file(args_save_file)) {
        if ((args_file_ptr = fopen(args_save_file, "w")) == NULL)
{
            …
        }
    …
```

Remediation

Upgrading to Release `2.0.3` of `fwknop` removes this vulnerability.

8) Fwknop Configuration File Ownership and Permissions are not Verified

Affected Products

The following versions of `fwknop`: `2.0.2, 2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2, 2.0.0-rc1`

Impact

A user could inadvertently use parameters that could affect the client execution.

Technical Details

The `fwknop` client regularly reads the configuration file `~/.fwknoprc`. If an attacker is able to create this file before the real user executes the program for the first time, an attacker can later alter the default parameters used. If this happens, the real user might use unwanted paremeters.

This behavior was observed in `client/config_init.c`:

```
/* Process (create if necessary) the users ~/.fwknoprc file.
*/
static void process_rc(fko_cli_options_t *options)
{
    FILE    *rc;
    int     line_num = 0;
    int     rcf_offset;
    char    line[MAX_LINE_LEN];
    char    rcfile[MAX_PATH_LEN];
    char    curr_stanza[MAX_LINE_LEN] = {0};
    char    var[MAX_LINE_LEN]  = {0};
    char    val[MAX_LINE_LEN]  = {0};

    char    *ndx, *emark, *homedir;

#ifdef WIN32
    homedir = getenv("USERPROFILE");
#else
    homedir = getenv("HOME");
#endif
```

```
    if(homedir == NULL)
    {
        fprintf(stderr, "Warning: Unable to determine HOME
directory.\n"
            " No .fwknoprc file processed.\n");
        return;
    }

    memset(rcfile, 0x0, MAX_PATH_LEN);

    strlcpy(rcfile, homedir, MAX_PATH_LEN);

    rcf_offset = strlen(rcfile);

    /* Sanity check the path to .fwknoprc.
     * The preceeding path plus the path separator and
'.fwknoprc' = 11
     * cannot exceed MAX_PATH_LEN.
    */
    if(rcf_offset > (MAX_PATH_LEN - 11))
    {
        fprintf(stderr, "Warning: Path to .fwknoprc file is too
long.\n"
            " No .fwknoprc file processed.\n");
        return;
    }

    rcfile[rcf_offset] = PATH_SEP;
    strlcat(rcfile, ".fwknoprc", MAX_PATH_LEN);

    /* Open the rc file for reading, if it does not exist, then
create
     * an initial .fwknoprc file with defaults and go on.
    */
    if ((rc = fopen(rcfile, "r")) == NULL)
…
```

Remediation

Upgrading to Release `2.0.3` of `fwknop` removes this vulnerability.

9) Fwknop History File Ownership and Permissions are not Verified

Affected Products

The following versions of `fwknop`: `2.0.2, 2.0.1, 2.0.0, 2.0.0-rc5, 2.0.0-rc4, 2.0.0-rc3, 2.0.0-rc2, 2.0.0-rc1`

Impact

A user could inadvertently execute unexpected content that could affect either the local host or a remote server.

Technical Details

The `fwknop` client stores the last command executed in `~/.fwknop.run`. If an attacker is able to create this file before the real user executes the program, the attacker can later alter the contents of this file. If this happens, the real user might inadvertently execute dangerous content when using the parameter `--last-cmd`. If this issue is used in conjunction with issue #3, the risk is even higher.

This behavior was observed in `client/fwknop.c`:

```
static void save_args(int argc, char **argv)
{
    char args_save_file[MAX_PATH_LEN];
    char args_str[MAX_LINE_LEN] = "";
    FILE *args_file_ptr = NULL;
    int i = 0, args_str_len = 0;

    …

    if (get_save_file(args_save_file)) {
        if ((args_file_ptr = fopen(args_save_file, "w")) == NULL)
{
            …
        }
    …
```

Remediation

Upgrading to Release `2.0.3` of `fwknop` removes this vulnerability.