COMP4108 Assignment Review

William Findlay

December 17, 2019

Contents

1 Assignment 1

1.1 John the Ripper

Default Settings.

- single crack mode
- then default word list and rules
- attempt incremental after

Modes.

- single crack
  - try GECOS/Full Name fields and home directory names as candidates with mangling rules
- word list
  - try all words in a list, with mangling rules
- incremental
  - try every possibility one after another

Mangling Rules.

- use `--rules=`
- extra
  - extra mangling
- single
  - make single substitutions
- jumbo
  - combines all options

Wordlists.

- sensible default wordlists
- specify new word lists using `--wordlist=`

1.2 Guessing Decryption Password

- use John to generate candidate passwords
- attempt them all on the file

1.3 Online Attack

- use John to generate candidate passwords
- check response from server
- wait for rate limiting before trying again

2 Assignment 2

2.1 Files

`/etc/group`.

- information about group names and IDs

`/etc/passwd`.

- information about UIDs, home directories, shells, etc.

`/etc/shadow`.

- contains username and password hash pairs

2.2 `find`

- `find /path/to/dir [options]`
- `-type` filters file type
- `-name` filters filename
- `-group` filters by group of owner
- `-user` filters by username of owner
- `-perm` filters by permissions
- `-exec "the_command {};"` runs a command on each found file, `{}`

2.3 `chmod`

- `chmod ugo+rwx file`
- `chmod ugo-rwx file`
- `chmod 0777 file`
- `chmod 7777 file`
- etc.

2.4 `chown`

- `chown user:group file`
- etc.

2.5 `getfacl`

- `user::rwx`
  - owner

- `group::rwx`
  - group member
- `other::rwx`
  - all other users by default
- `mask::r-x`
  - all other permissions will no longer have `w`
- `user:the_user:rwx`
- `group:the_group:rwx`

2.6 `setfacl`

- `setfacl -m user:the_user:rwx`
  - set `rwx` permission for `the_user`
- `setfacl -x user:the_user`
  - remove entry for `the_user`

2.7 TOCTOU

- `access` check before opening file for writing
- if we can change the file between access check and open, we can get a race condition
  - use symlink to do this
- lower niceness makes a process go faster, higher niceness makes it go slower
  - make the checking program high niceness, swapping program low niceness

3 Assignment 3

3.1 LKM Compilation

- use the Makefile, has a `make -c` command that runs Makefiles in the kernel sources
- run `insert.sh` to insert the module
  - parameters to `insmod` allow us to pass parameters to the module when it's inserted

3.2 Rootkit Setup

- get syscall table address from system boot map and pass it in
- syscall table is protected, we need to change memory protection to `rw` to make changes
  - then remember to change it back

3.3 Syscall Hooks

- save pointer to old function
- write new function, perform preprocessing and/or postprocessing
  - make sure to call function at old function pointer in our new function
- find offset in table with syscall number

- overwrite function pointer at that offset with our new function

3.4 Walking `dents`

- use `d_reclen` for the size of each entry
- walk forward by `d_reclen` bytes by casting buffer to `char*` and using pointer arithmetic
- to sanitize
  - use `memmove` rather than `memcpy` because `memmove` allows safe modification while walking
  - compare magic prefix with `strncmp` using length of prefix as `n`

4 Assignment 4

4.1 SSH Tunnels

- `ssh -L localport:host:hostport hostname`
- `ssh -R hostport:host:localport hostname`

4.2 Command Execution

- escape command using `;` then add our own command
- harder level filters `;` and `&&`, but we can use `&` to run as background process and escape

4.3 CSRF

- exploits session cookie that keeps us logged in
  - we can use this to authorize our phony request
- make a fake page that redirects to the password form submission and set parameters with `?field=value`
- harder version requires same origin policy
  - we get around this by writing a webserver to host our attack page from localhost

4.4 Redirection

- just set `?page=/path/to/secret`

4.5 XSS Stuff

- this is all easy enough

4.6 Where's the BeEF

Stored XSS Attack.

- increase input length by changing HTML properties

    - they should be checking length server-side but aren't
- app doesn't sanitize `<script>` tags, so we source our malicious JS that way

BeEF Stuff.

- this is straightforward

5 Assignment 5

5.1 `nmap`

- `nmap -p 1- -sS localhost`
    - `-p 1-` scan all ports
    - `-sS localhost` TCP SYN scan on localhost
    - need to be root to do SYN scan because it uses raw sockets
- other scans
    - `-sA` TCP ACK scan
        - check packets filtered by firewall
    - `-sF` TCP FIN scan
        - tells us if port is closed or open
        - `RST` means closed
    - `-sT` TCP connect() scan
        - slower and crappier and less stealthy version of SYN scan
        - used when we don't have root
        - need to wait for response each time
    - `-sP` ping discovery for subnet `192.168.*` for example
    - `-O` operating system info for IP address

5.2 `tcpdump`

- uses BPF, so we need to be root
- choose an interface with `-i`
    - `any` matches all
    - `lo` matches loopback = localhost

5.3 `iptables`

- `--list` shows us rule chains
- `-F` then `iptables-save` to flush filters
    - `iptables-save` command to commit
- `-m` uses a module
    - `conntrack --ctstate` tracks connection state (stateful filtering)
- rules need results
    - `ACCEPT`
    - `REJECT`

- `DROP`