Security
28 May 2014

- This exam consists of three parts: (1) eight short questions, every one counts for 5 points, (2) four true/false questions, every one counts for 2.5 points, and (3) five problems, every one counts for 12.5 points. In order to get the maximum amount of points,
  - answer the Short Questions, and
  - answer the True or False Questions, and
  - pick and solve **four out of five** Problems from the third part. Please specify below which ones I should grade. If you forget to do so, I will grade the first four problems, i.e., Problems 1, 2, 3, 4.
- Mark every page with name and student number.
- Use of books, additional course material, and calculator is prohibited.
- Do not use pencil or red ink. Give your answers on the exam paper (if needed, you may request additional paper.)
- Answer in English.

Please tick below the four problems from the third part you have chosen to solve:

Problems to be graded	**Problems**
	Problem 1: Cryptography
	Problem 2: Access Control
	Problem 3: Exploits
	Problem 4: Vulnerabilities
	Problem 5: Firewalls

---

1 Short Questions (40 points: 8 × 5 points)

1. Is a TCP connection secure against eavesdropping? Why or why not?

2. Banks use an Electronic Clearing House (ECH) network to process checks. Suppose that Bank A may send a file to Bank B, with each record in the file representing one check deposited in Bank A that is to be withdrawn from Bank B. Assume that each record contains only two account numbers and the amount of money to be transferred. The ECH file format specifies an integrity checksum for each record, but not for the entire file. How can this be exploited by an insider at Bank A who does not have the ability to write valid checksums, but can modify a file? (Hint: assume that checksums are long enough to render all birthday-paradox based attacks impractical.)

3. How does salt slow down an offline dictionary attacks? Specifically, how many hash computations are needed to run a dictionary attack with and without salt? Assume d passwords in the dictionary, and k-bit long salt.

4. Assume that you want to slow down an offline dictionary attack even further. What can you do? **Hint:** the goal is to add a time factor that is not related to the complexity of the password.

5. What is the principle of least privilege? Why is it important?

6. What is a polymorphic virus? What is a metamorphic virus? Which one is harder to block?

7. Which is generally safer (from a security point of view), a firewall with a *default deny* policy or a firewall with a *default allow* policy? Why?

8. How many lecturers gave at least one lecture during the course?

2 True or False (10 points: 4 × 2.5 points)

Circle TRUE or FALSE. Optionally you can add one line to justify your answer.

9. TRUE or FALSE: A host-based intrusion detection system (HIDS) is harder to evade than a network intrusion detection system (NIDS) because a HIDS has access to application-layer semantics.

10. TRUE or FALSE: No intrusion detection system is capable of detecting novel attacks.

11. TRUE or FALSE: Diffie-Hellman is secure against passive eavesdroppers who cannot modify packets or send forged packets.

12. TRUE or FALSE: Diffie-Hellman is secure against man-in-the-middle attacks.

3 Problems (50 points: 4 × 12.5 points)

13. Problem 1: Cryptography (12.5 points)

Let's consider the following statement:

If `SA` is the secret key of A, `PB` is the public key of B, and `E(K, X)` denotes encryption of `X` with the key `K`, then we can safely say that for practical purposes

```
E(SA, E(PB, M)) = E(PB, E(SA, M)).
```

Note that the equality sign should be read as an equality in the mathematical sense, i.e., stating that the message on the left-hand-side of the equation is *exactly the same* as the message on the right-hand-side of the equation.

a) (2.5 points) Is this statement true or false?

b) (10 points) If the statement is true, explain why. If it is false, give a complete counterexample. A complete counterexample should (1) specify which public-key encryption algorithm you chose, (2) provide example concrete keys and (3) an example concrete message. If this algorithm involves some mathematical computations, (4) please perform them. Finally, if this algorithm makes some assumptions about the keys and/or messages, make sure to (5) state them, and (6) show that your example satisfies them.

14. **Problem 2: Access Control (12.5 points)**

One way to express or analyze access control policies is using logical if-then rules, often written backward in then-if form. For example, the rule below says that a `User` has a specific `Permission` to access a `Resource`, if the `User` is a member of a `Group` that is given that `Permission` for the `Resource`.

```
canAccess(User, Permission, Resource) <-
       member(User, Group),
       canAccess(Group, Permission, Resource)
```

A single rule such as this can apply to all permissions in the system (such as read permission, write permission, and execute permission) and all resources (such as all files). A comma `,` on the right-hand-side of a rule means `and`. Names such as `Group` and `User` that begin in upper-case are variables and can be replaced by any value.

a) (3 points) In order to decide whether a user can access a file, for example, rules are combined with facts. For example, consider the facts

```
member(alice, admins)
member(bob, users)
canAccess(admins, Permission, Resource)
```

Explain which user(s) these facts (combined with the rule above) allow to read file `\etc\shadow`.

b) (3 points) We can associate access lists with files using facts and rules based on formulas of the form

```
file(FileID, ReadList, WriteList, ExecList, Owner),
```

where `ReadList`, `WriteList`, and `ExecList` are the groups allowed to read, write, and execute the file (respectively), and `Owner` is the owner of the file. For example:

```
file(f1234, read1234, write1234, exec1234, alice)
member(bob, read1234)
member(carol, write1234)
member(carol, exec1234)
```

Write a single rule that allows the `WriteList` group to write that file (when combined with other rules given in this problem). Your rule should imply `canAccess(carol, write, f1234)`, for example, but should not mention `carol` or `f1234`.

```
canAccess(WriteList, write, FileID) <-

    file(FileID, ..........., ..........., ..........., ...........)
```

c) (6.5 points) We can write a rule expressing that a process created by executing a file has the same permissions as the owner of the file. For simplicity, assume that when a process created by executing a file tries to perform an action, the system determines whether `canAccess(FileID, Permission, Resource)` is granted by the system. Under this assumption, we can use the following rule:

```
canAccess(FileID, Permission, Resource) <-
    canAccess(Owner, Permission, Resource),
    file(FileID, R, W, X, Owner)
```

Something is very wrong with this, however, if writing to a file does not change the owner of the file. Explain how the access control system given by the rules and facts stated in this problem (including all parts above) allows carol to execute any sequence of commands that an administrator is allowed to execute. Assume rules as in part (b) so that every member of the readlist of a file can read it, and similarly for write and execute.

15. **Problem 3: Exploits (12.5 points)**

Note: this question is different from the second challenge.

You were given credentials of a regular user on a certain classified machine. Note that this account does ***not*** give you root access. The machine contains a secret file, `/home/classified/top-secret`. Below is a screenshot illustrating some basic checks.

```
$ whoami
makai
$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 6.0.9 (squeeze)
Release:        6.0.9
Codename:       squeeze
$ uname -a
Linux server 2.6.32-5-686 #1 SMP Mon Sep 23 23:00:18 UTC 2013 i686 GNU/Linux
$ ls -l /home/classified
total 4
-rw------- 1 root root 13 May 27 19:06 top-secret
$
```

a) (5 points) Having explored this machine, you can see that there is an `nginx` process running:

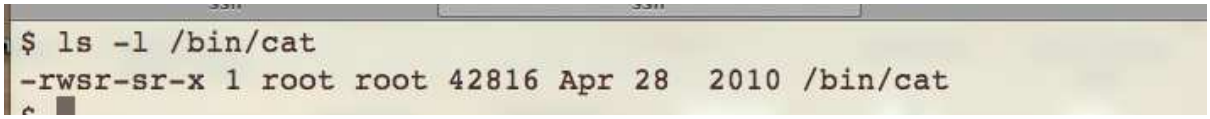
```
$ ps aux | grep nginx
root     10358  0.0  0.1   2972   456 ?        Ss   Apr30   0:00 nginx:
master process /usr/local/nginx/sbin/nginx
nobody   22737  0.0  0.3   3148   772 ?        S    May13   0:00 nginx:
worker process
makai    32197  0.0  0.2   3276   700 pts/2    S+   18:50   0:00 grep ng
inx
$ /usr/local/nginx/sbin/nginx -V
nginx version: nginx/1.4.0
built by gcc 4.4.5 (Debian 4.4.5-8)
configure arguments:
```

As you know from Challenge 2, there is a CVE entry (CVE-2013-2028) describing a problem with the ***worker process*** of this webserver: "The `ngx_http_parse_chunked` function in `http/ngx_http_parse.c` in `nginx 1.3.9` through `1.4.0` allows remote attackers to cause a denial of service (crash) and execute arbitrary code via a chunked Transfer-Encoding request with a large chunk size, which triggers an integer signedness error and a stack-based buffer overflow.". It's worth noting that the metasploit framework has already a module which one can use to send a malicious request exploiting this vulnerability, and get a shell.

Can you exploit this vulnerability to read the secret file? If so, why? How would you proceed? If

not, what prevents you from succeeding? Justify your answer.

b) (7.5 points) Having explored the machine further, you can observe the following:



Can you use the `cat` command to read the secret file? If so, why? How would you proceed? If not, what prevents you from succeeding? Justify your answer.

(Just in case: `cat` is a Unix command that can be used to display the contents of a file on the screen. To read the contents of a file, just enter `cat filename`.)

16. **Problem 4: Vulnerabilities (12.5 points)**

Consider the following code snippet:

```
/* *to is formally a pointer to an array. You can think of it as a way for the program
 * to pass a buffer to this function. Thus this buffer is allocated in main(), and only
 * passed to copy_buf.
 */
int copy_buf(char *to, int pos, char *from, int len) {
    int i;
    for (i = 0; i < len; i++) }
        to[pos] = from[i];
        pos++;
    }
    return pos;
}

int display_file(char *filename) {
    /* open the given file, and send the contents over the network_socket */
}

int main(int argc, char *argv[]) {
    char filename[256];
    char request[32];

    copy_buf(filename, 0, "\home\data.txt\0", 15);
    read(network_socket, request, 64);
    display_file(filename);

    /* process the request */
    return 0;
}
```

Assume that you have a control over the `network_socket`. Further, assume that the compiler placed the `request` buffer just underneath the `filename` buffer, i.e., the `request` buffer starts at a lower address in memory.

a) (5 points) Identify a buffer overflow vulnerability, and sketch how you could exploit it to execute your own code. Assume that no protection mechanisms have been implemented.

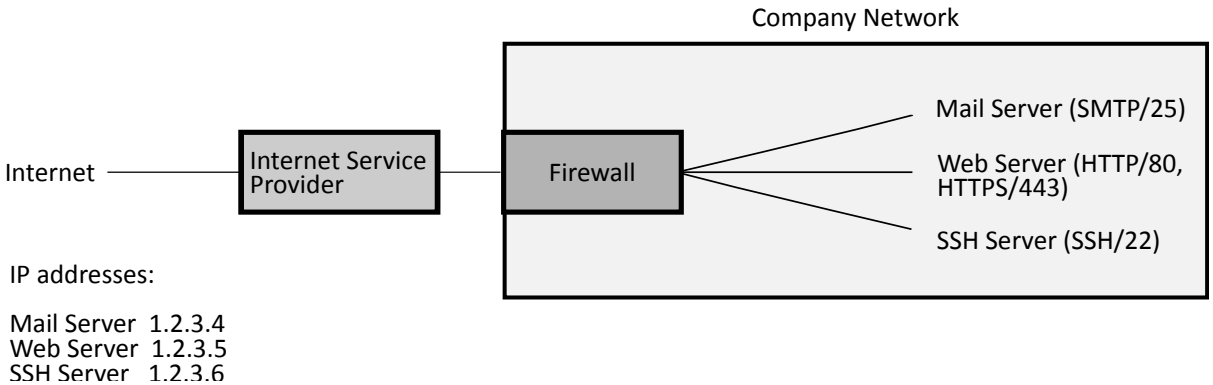
b) (5 points) Assume that during compilation, the compiler protected the return address of each function: it placed a random canary value below the old frame pointer address (in principle, it also means that the canary is located below the return address), and above the allocation space for local variables. On a function exit, a small code snippet checks if this value has been modified. If so, the program crashes.

Could you still exploit the vulnerability in such a way that the program wouldn't crash, but you would manage to leak a security critical file, e.g., `/etc/passwd`? Justify your answer.

c) (2.5 points) Rewrite the program so that it is no longer vulnerable to a buffer overflow attack.

17. **Problem 5: Firewalls (12.5 points)**

The following diagram shows the architecture for a company network and its connection to the internet. The company is installing a packet filter firewall (i.e., a stateless firewall).



Here is the proposed security policy for the firewall:

1. By default, block all inbound connections.
2. Allow all inbound TCP connections to SMTP on mail server.
3. Allow all inbound TCP connections to HTTP and HTTPS on web server.
4. Allow all inbound TCP connections to SSH on SSH server.
5. Allow all outbound connections.

Example rules:

```
allow * *:*/in -> *:*/out
drop * *:* -> *:*
```

Hint: remember that firewall rules are always checked and applied in the specified order, starting from the top one.

a) (6.5 points) Using syntax from the lecture (examples above), write the firewall ruleset for the company's firewall.

b) (3 points) Hackers target the company's network with repeated requests for large images on the company's webserver. The hackers machines are on the 9.8.7.x subnet. How could you change your firewall ruleset to block these attacks? (If you add a new rule, don't forget to say at which location in the ruleset from part (a) it should be located.)

c) (3 points) Employees start downloading lots of movie trailers from a website at 27.5.20.14:80. How could you change your firewall rules to stop employees from accessing the website? (If you add a new rule, don't forget to say at which location in the ruleset from part (a) it should be located.)