SIM Global Education

UNIVERSITY OF WOLLONGONG AUSTRALIA

CSCI262 – System Security

Tutorial Set D

Malicious code or malware

14 October 2022

Part Zero

There is a Windows executable pinNeeded2.exe. Find the password. Use ptime.exe to help you.

Tutorial Set D – Part One

Have a look at RFC 2410: rfc2410.txt

Tutorial Set D – Part One

2. You should have a look at http://ha.ckers.org/xas.html. This is a collection of sneaky ways to insert statements into websites.

Tutorial Set D – Part One

3. An attacker would be extremely pleased to find a website that had implemented a bogosort algorithm for sorting provided data. Why?

You can compile and run the provided Bogosort.cpp using one of the following instructions to compile it on Banshee.

g++ Bogosort.cpp –o Bogo

CC Bogosort.cpp –library=stlport4 –o Bogo

You can run some tests to obtain some idea of the distribution of run lengths.

Tutorial Set D – Part One

4. Read F-Secure News from the Lab.pdf. Should we worried?

Tutorial Set D – Part One

5. Run Bomb.sh on Banshee. What is it doing?

CSCI262 – Systems Security

Tutorial Set D

Part Two

Tutorial Set D – Part Two

1. Look at the Wild List at http://www.wildlist.org. Compare the typical lifetimes of recent wild viruses, as opposed to those earlier in the history of the wild list.

Tutorial Set D – Part Two

2. What is a zero–day exploit?

Zero-day exploit refers to a known vulnerability that attackers is able to exploit before software vendor comes out with patches for the vulnerability; in other words, once a vulnerability is uncovered or known and before a vendor comes out with patches for the vulnerability, the attackers exploit the vulnerability and carry out the attack. The name zero-day is because in this kind of attack, the attackers is able to carry out the attack on the same day (0 day of lead time) a vulnerability is known.

Tutorial Set D – Part Two

3. What damage did the worm of 1988 cause?

Type: Worm **Year:** 1988 **Users affected:** Infected an estimated 6,000 university and military computers connected over the Internet. Became one of the first worm's to spread extensively "in the wild", and one of the first well-known programs exploiting buffer overrun vulnerabilities. **Infection and spread:** Exploited known vulnerabilities in Unix sendmail, Finger, rsh/rexec, and weak passwords. An unintended consequence of the code, was that a computer could be infected multiple times and each additional process would slow the machine, eventually making it unusable. **Hacker Motivation:** (Experiment) Morris was written by Cornell University graduate student, Robert Tappan Morris. He claims it was not written to cause damage but to gauge the size of the Internet. **Damage caused:** (Medium-Major) The U.S. Court of Appeals estimated the cost of removing the virus from each installation was in the range of $200 – 53,000. The U.S. General Accounting Office estimated that damages may have cost as much as $10 million. Harvard spokesman Clifford Stoll estimated the total economic impact was between $100,000 - $10,000,000. There were an estimated **60,000** computers attached to the Internet at the time, and the worm might have infected about **10 percent of them**. **Aftermath:** Robert Morris was tried and convicted of violating the 1986 Computer Fraud and Abuse Act. After appeals he was sentenced to three years probation, 400 hours of community services, and a fine of $10,000.

Tutorial Set D – Part Two

Robert Tappan Morris, the creator of Morris worm that infected an estimated 6,000 university and military computers connected over the Internet.

Source: http://en.wikipedia.org/wiki/Morris_worm

Tutorial Set D – Part Two

4. What would the effect of a worm spreading on the scale of the 1988 one likely be today?

I) Total Internet Human Usage By Country of Residence

1. **1790 million to 2230 million**: the estimated number of unique individuals who will use the Internet in 2012, all countries combined.
2. **282 million**: the estimated number of American residents who will use the Internet in 2012.
3. **61.5 million**: the estimated number of United Kingdom residents who will use the Internet in 2012.
4. **63.1 million**: the [illegible]nternet in 2012.
5. **124 million**: the [illegible]Internet in 2012.
6. **76.5 million**: the [illegible]nternet in 2012.
7. **34.1 million**: the estimated number of Canadian residents who will use the Internet in 2012.
8. **440.2 million**: the estimated number of Chinese residents who will use the Internet in 2012.

10% of 2230 million = 223,000,000

Tutorial Set D – Part Two

5. In an ACL based access control system, there are three users: Alice, Bob, and Carol. The ACLs with respect to Files F1, F2, and F3 are as follows:

   F1: (Alice, r), (Bob, rw), (Carol, x)
   F2: (Alice, rx), (Carol, rwx)
   F3: (Bob, rwx)

   Assume that the virus can only spread when a user executes it. (Note, the spreading of the virus is when an infected file is executed, the virus becomes active and it is spread when a write action is done.)
   Consider each of the following independent scenarios:

   i. A virus has somehow infected F1 on Alice's account. Can the virus infect F2. Justify your answer. What are the impacts on Bob and Carol?
   ii. If a virus has infected F1 through Bob's account (independent from question i), what will be the impact on F2 and F3?
   iii. If a virus has infected F2 through Carol' account, what will happen to F1 and F3?

Tutorial Set D – Part Two

5. In an ACL based access control system, there are three users: Alice, Bob, and Carol. The ACLs with respect to Files F1, F2, and F3 are as follows:

   F1: (Alice, r), (Bob, rw), (Carol, x)
   F2: (Alice, rx), (Carol, rwx)
   F3: (Bob, rwx)

   i. A virus has somehow infected F1 on Alice's account. Can the virus infect F2. Justify your answer. What are the impacts on Bob and Carol?

Although Alice account is infected, the virus cannot infect F2 through Alice, because Alice cannot write to F2. Carol can execute F1 and as a result, Carol's account becomes infected. If Carol write F2, F2 will be infected through Carol (by "write"). Bob cannot execute F1 so Bob's account is not infected, so F3 is safe.

Tutorial Set D – Part Two

5. In an ACL based access control system, there are three users: Alice, Bob, and Carol. The ACLs with respect to Files F1, F2, and F3 are as follows:

F1: (Alice, r), (Bob, rw), (Carol, x)
F2: (Alice, rx), (Carol, rwx)
F3: (Bob, rwx)

ii. If a virus has infected F1 through Bob's account (independent from question 5i), what will be the impact on F2 and F3.

Carol has got the same problem as question 1, so F2 is infected if Carol write F2. In addition, since Bob's account is infected, F3 will also be infected, if Bob writes it. Hence files F2 and F3 may can be infected.

Tutorial Set D – Part Two

5. In an ACL based access control system, there are three users: Alice, Bob, and Carol. The ACLs with respect to Files F1, F2, and F3 are as follows:

   F1: (Alice, r), (Bob, rw), (Carol, x)
   F2: (Alice, rx), (Carol, rwx)
   F3: (Bob, rwx)

   iii. If a virus has infected F2 through Carol' account, what will happen to F1 and F3?

F1 cannot be infected through Alice, since she can read it only. F1 cannot be infected through Carol as well, since she cannot write to F1. F3 is safe because Bob cannot execute F2 and as a result, Bob's account is not infected. Carol can execute F2, and thus it is possible that Carol's account be infected. However, Carol has no access to F3 so F3 cannot be infected through Carol. So F1 and F3 are safe.

Tutorial Set D – Part Two

6. In general, do capabilities offer more or less protection against Trojan horses than do access control lists. Justify your answer.

Tutorial Set D – Part Two

Theoretically, both Capabilities List and Access Control Lists are equivalent. Thus, both would provide the same level of protection. In other words, Capabilities Lists do not provide more nor less protection on Trojan Horse than ACLs do.

Recall access control matrix. The rows are capabilities and the columns are ACL. As an example,

	F1	F2	F3
Alice	r	rw	x
Bob	rx		rwx
Carol	rwx		

We can see that all analyses given in question1 can still apply.

Tutorial Set D – Part Two

7. A computer system uses the BLP policies for access control. How would a virus spread if:
   a) the virus were placed on the system at system low (the compartment that all other compartments dominate)?
   b) the virus were placed on the system at system high (the compartment dominates all other compartments)?

Tutorial Set D – Part Two

Recall the principle of “write up” and “read down” (S-property and *-property).

a) If a virus is placed on the lowest security level, then the virus can write up to higher levels and infect the associated files.

b) If a virus is placed on the highest security level, then the virus cannot spread to any lower level. This is because you can only read from a lower level.

Tutorial Set D – Part Two

- Look at http://mtc.sri.com/Conficker/addendemC/. This is probably something more to read in detail some other time though.

Tutorial Set D – Part Two

8. A system provides protection using the flow distance scheme. There are four users (A, B, C, D) in the system. They can assess flow distance (fd) less than 4, 3, 2, and 1, respectively. B creates process P which has been unfortunately infected by a virus. Initially, fd(P) = 0. Answer the following questions:

   i. Can P infect B's files? Justify your answer.
   ii. Can P infect D's files? Justify your answer.
   iii. Can A access P? How? If yes, after access, can other users access it.

Tutorial Set D – Part Two

A user can only access a file which has fd less than the permission.

An fd for an object or a subject/process increases only when it is shared.

i. Since B can access files whose fd is less than 3, so he can access P which then infects B's files.

ii. When D shares P, fd(P) increases by 1, i.e., fd(P) = 1. P cannot infect D's files, since D can only access files whose fd is 0. As a result, P cannot infect D's files.

Tutorial Set D – Part Two

iii. A can access P, since he can access files whose fd is less than 4. After access, fd(P) increases by 1.

- Since P has been shared by B, fd(P) will not further increase (because B has already shared P), i.e., B can still access it (since B can access files whose fd is less than 3).
- When C accesses P, fd(P) increases by 1, i.e., fd(P) = 2, so C cannot access it and P will not infect C's files.
- When D tries to access P, since D has shared it in (ii), i.e., fd(P) = 2, D cannot access it and P will not infect D's files.

CSCI262 – Systems Security

Tutorial Set D

Part Three

Tutorial Set D – Part Three

1. How can you test randomness?

   Execute the random function, collect the output (results) and determine the statistical property of the results.

Tutorial Set D – Part Three

2. Run some tests on the shell function $RANDOM to test its randomness.

```
#!/bin/bin/bash
function pword(){
   cat /dev/urandom | tr -cd '[:graph:]' | head -c ${1:-10}

}
```

Save the above code as pw1.sh
Type the following commands:
>source pw1.sh
>pword 5
>pword

Tutorial Set D – Part Three

- When pword is executed with an argument (5), it generates random password with 5 characters. And when pword is called without argument, it generates 10 characters length password.
- ${1:-10} indicates if $1 is unset or null , 10 will be returned. Otherwise, value of $1 will be substituted.

Tutorial Set D – Part Three

3. Perform similar tests of the random function calls in C++ and/or in Java.

Tutorial Set D – Part Three

4. Sketch a program, in whatever language you like, to illustrate the idea of race conditions.

   A race condition is a situation in which two or more threads or processes are reading or writing some shared data, and the final result depends on the timing of how the threads are scheduled. Race conditions can lead to unpredictable results and subtle program bugs. A thread can prevent this from happening by locking an object. When an object is locked by one thread and another thread tries to call a synchronized method on the same object, the second thread will block until the object is unlocked.

Tutorial Set D – Part Three

- Race condition in Java is a type of concurrency bug or issue which is introduced in your program because parallel execution of your program by multiple threads at same time, Since Java is a multi-threaded programming language hence risk of Race condition is higher in Java which demands clear understanding of what causes a race condition and how to avoid that. Anyway Race conditions are just one of hazards or risk presented by use of multi-threading in Java just like deadlock in Java. Race conditions occurs when two thread operate on same object without proper synchronization and there operation interleaves on each other. Classical example of Race condition is incrementing a counter since increment is not an atomic operation and can be further divided into three steps like read, update and write. If two threads tries to increment count at a same time and if they read the same value because of interleaving of read operation of one thread to update operation of another thread, one count will be lost when one thread overwrite increment done by other thread.

http://www.java-forums.org/reviews-advertising/55999-race-conditions-java-2-examples.html
10 August 2013

Tutorial Set D – Part Four

1. What is Fuzzing?

   A software testing technique, developed by Professor Barton Miller at the University of Wisconsin Madison in 1989, that use randomly generated data as inputs to a program. The range of inputs that may be explored is very large. They include direct textual or graphic input to a program, random network requests directed at a Web or other distributed service, or random parameters values passed to standard library or system functions.

   The intent is to determine whether the program or function correctly handles all such abnormal inputs or whether it crashes or otherwise fails to respond appropriately. (Stallings and Brown, pp. 402 – 403)

Stallings and Brown, Computer Security: Principles and Practice, 2008, Pearson Education, Inc. Upper Saddle River, New Jersey

Tutorial Set D – Part Four

2. What limitation does fuzzing have?

The limitation of fuzzing is that it is unlikely to locate a bug that is triggered by a small number of very specific input values. (Stallings and Brown, pp. 402 – 403)

Tutorial Set D – Part Four

3. Should fuzzing be considered a “security concept” or a “software engineering” concept? Explain.

   In my opinion, it is a “software engineering” concept. Fuzzing has a characteristic of software testing, and its main intention is to determine whether a program or function correctly handles abnormal inputs. However, due to its ability to identify a widely range of potential weaknesses in a programs, fuzzing is used by attackers (hackers) to identify bugs in commonly deployed software. As a result, security experts and systems developers are also use this technique to locate and correct the errors (bugs).

Tutorial Set D – Part Four

4. The idea of JavaScript is that it is downloaded to your computer and then run. As such, the source is visible to the user of the machine it runs on. Is it possible to stop a user running the JavaScript usefully once they have it on their machine? Can you, for example, require a password to access the script? Explain your answer carefully.

Tutorial Set D – Part Four

5. What is an injection attack? Give an example.

   (Stallings and Brown, 2008) describes injection attack as a wide variety of program flaws related to invalid handling of input data.

   Examples: SQL injection attack, Web CGI injection attack, PHP code injection attack, and Cross-site scripting attack.

Stallings and Brown, Computer Security: Principles and Practice, 2008, Pearson Education, Inc. Upper Saddle River, New Jersey, pp. 394 – 400.

Tutorial Set D – Part Four

6. What is the difference between software reliability and software security?

Software reliability is concerned with the accidental failure of a program as a result of some theoretically random, unanticipated input, system interaction, or use of incorrect code.

Software security is concerned with the vulnerabilities of a program that allows an attacker to exploit it.

Tutorial Set D – Part Four

7. In one of the reading files the idea of ”Penetrate and Patch” is described as a ”dumb idea”. What is it and why might it not be considered an appropriate approach to security?

From the article “The Six Dumbest Ideas in Computer Security”, the editor discuss the issue with an example. The editor explained that the problem with “Penetrate and Patch” approach to enhance security is “dumb” because the approach instead of making the code/implementation/system better by design, it only makes it toughened by trial and error. In addition, the fixes always ended with the addition of new code; this potentially introduces new security vulnerability as well.

Tutorial Set D – Part Four

Have a look through Koobface_Jul2009.pdf and Koobface_Inside.pdf

a) What is Koobface?
b) What did it do?
c) Look at the way in which different technologies have been applied.
d) Is Koobface still a problem?

Reference

- William Stallings and Lawrie Brown, Computer Security: Principles and Practice, Pearson Education, 2012
- http://www.wildlist.org
- http://en.wikipedia.org/wiki/Morris_worm
- https://en.wikipedia.org/wiki/Koobface