CS165 – Computer Security

Final Review

December 6, 2024

Final Structure

- Three sections
  - 14 multiple choice (2pts each)
    - Fill in the blank with specific choices
  - 6 short answer (5pts each)
    - Scenarios that you answer 1 or 2 questions
      - Don’t skip the second question accidentally…
    - Free form – 2-3 sentences
  - 4 “constructions” (42pts altogether)
    - Scenarios with problem solving
    - 3-4 sub-questions
- Should have plenty of time

Final Scope

- Focus from the “Fuzz Testing” lecture on
  - Will be a question on payload generation
    - From prep for midterm
  - Will be questions related to Project 2 and Project 3
    - Based on the projects and the lecture
      - Even tho no homework question
- Will include concepts from earlier – in particular, memory safety and memory errors

Homework (Part I)

- 1. Suppose an attacker has modified a program to insert malicious code at address 0x0804ff00 of the binary (when view via ‘objdump’). How can the attacker get that code run by modifying the binary file?
  - a) Launch a buffer overflow on the program to modify a return address to 0x0804ff00.
  - b) Modify the entrypoint in the program binary file to the value 0x0804ff00.
  - c) Use a return-oriented gadget to change the stack pointer to refer to the address 0x0804ff00.
  - d) Modify the first line of the binary file to call 0x0804ff00.
- 2. What functions may be invoked using an indirect call (i.e., function pointer) using coarse-grained control-flow integrity (CFI)?
  - a) All functions.
  - b) Only functions whose type signatures (i.e., return address, number of arguments, and argument types) match the type of the function pointer.
  - c) Only functions whose values may be assigned to function pointers.

Homework (Part I)

- 1. Suppose an attacker has modified a program to insert malicious code at address 0x0804ff00 of the binary (when view via ‘objdump’). How can the attacker get that code run by modifying the binary file?
  - a) Launch a buffer overflow on the program to modify a return address to 0x0804ff00.
  - b) Modify the entrypoint in the program binary file to the value 0x0804ff00.
  - c) Use a return-oriented gadget to change the stack pointer to refer to the address 0x0804ff00.
  - d) Modify the first line of the binary file to call 0x0804ff00.
- 2. What functions may be invoked using an indirect call (i.e., function pointer) using coarse-grained control-flow integrity (CFI)?
  - a) All functions.
  - b) Only functions whose type signatures (i.e., return address, number of arguments, and argument types) match the type of the function pointer.
  - c) Only functions whose values may be assigned to function pointers.

Homework (Part I)

- 3. A cross-site scripting (XSS) attack is possible because a __________ may mix code and data on a __________.
  - a) Web browser and web page
  - b) Web server and web page
  - c) Web browser and web server
  - d) Web server and web browser
- 4. Which function can be invoked when enforcing fine-grained control-flow integrity (CFI) using type signature matches for the function pointer `int (*int_fn)(char *, int)`?
  - a) `int *function1(char *x, int y)`
  - b) `int *function2(char x, int y)`
  - c) `int function3(char *x1, char *x2, int y)`
  - d) `int function4(char *x, int y)`

Homework (Part I)

- 3. A cross-site scripting (XSS) attack is possible because a __________ may mix code and data on a __________.
  - a) Web browser and web page
  - b) Web server and web page
  - c) Web browser and web server
  - d) Web server and web browser
- 4. Which function can be invoked when enforcing fine-grained control-flow integrity (CFI) using type signature matches for the function pointer `int (*int_fn)(char *, int)`?
  - a) `int *function1(char *x, int y)`
  - b) `int *function2(char x, int y)`
  - c) `int function3(char *x1, char *x2, int y)`
  - d) `int function4(char *x, int y)`

Homework (Part I)

- 5. Which domain satisfies the same origin policy for cookies from `http://example.com:80`?
  - a) `https://example.com:80`
  - b) `http://example.com/foo:80`
  - c) `http://example.com:8080`
  - d) `http://example_com.com:80`
- 6. Capabilities specify access rights from the perspective of __________.
  - a) subjects
  - b) objects
  - c) operations

Homework (Part I)

- 5. Which domain satisfies the same origin policy for cookies from `http://example.com:80`?
  - a) `https://example.com:80`
  - b) `http://example.com/foo:80`
  - c) `http://example.com:8080`
  - d) `http://example_com.com:80`
- 6. Capabilities specify access rights from the perspective of ___________.
  - a) subjects
  - b) objects
  - c) operations

Homework (Part I)

- 7. The target pathname of a symbolic link can be __________
  - a) Any file path.
  - b) Only file paths to filesystem resources (e.g., files and directories) whose access is authorized to the creator of the symbolic link.
  - c) Only file paths to a file system resource accessible to the “others” category in the UNIX permission system.
- 8. Why does the iptables firewall have “chains” of firewall rules? __________
  - a) To reduce the number of firewall rules necessary to be stored.
  - b) To reduce the number of rules that need to be checked to process a packet.
  - c) To increase the number of rules that may match a packet.

Homework (Part I)

- 7. The target pathname of a symbolic link can be __________
  - a) Any file path.
  - b) Only file paths to filesystem resources (e.g., files and directories) whose access is authorized to the creator of the symbolic link.
  - c) Only file paths to a file system resource accessible to the "others" category in the UNIX permission system.
- ~~8. Why does the iptables firewall have "chains" of firewall rules? __________~~
  - ~~a) To reduce the number of firewall rules necessary to be stored.~~
  - ~~b) To reduce the number of rules that need to be checked to process a packet.~~
  - ~~c) To increase the number of rules that may match a packet.~~

Homework (Part I)

- 9. Which of the following about fuzz testing is true?
  - a) A main goal in fuzz testing is to ensure that every bug is found because its testing must be complete.
  - b) A main goal in fuzz testing is to produce as many inputs values as possible for the same memory operation.
  - c) A main goal in fuzz testing is to generate inputs to execute more branches in the program.
- 10. In computer security, there's a well-known principle called the principle of the least privilege. The idea is that every subject (process, user, program) should have access to only the information and resources they absolutely need (no more should be allowed). What do you best think describes the motivation behind principle?
  - a) The reasoning behind the principle is to prevent an attacker from compromising a subject.
  - b) The reasoning behind the principle is to reduce the damage once a subject is compromised.
  - c) Both a) and b).

Homework (Part I)

- 9. Which of the following about fuzz testing is true?
  - a) A main goal in fuzz testing is to ensure that every bug is found because its testing must be complete.
  - b) A main goal in fuzz testing is to produce as many inputs values as possible for the same memory operation.
  - c) A main goal in fuzz testing is to generate inputs to execute more branches in the program.
- 10. In computer security, there's a well-known principle called the principle of the least privilege. The idea is that every subject (process, user, program) should have access to only the information and resources they absolutely need (no more should be allowed). What do you best think describes the motivation behind principle?
  - a) The reasoning behind the principle is to prevent an attacker from compromising a subject.
  - b) The reasoning behind the principle is to reduce the damage once a subject is compromised.
  - c) Both a) and b).

Homework (Part II)

- 1. What is a confused deputy? What is one way to prevent a confused deputy situation?

- 2. What attacks can a man-in-the-middle (MITM) launch against network communication? Why does the encryption of data thwart MITM attacks on TCP communications, even though the IP address and port are not encrypted?

Homework (Part II)

- 1. What is a confused deputy? What is one way to prevent a confused deputy situation?
  - A confused deputy is a computer program that is tricked by another program (lacking privileges of the victim) into misusing its authority on the system. It is a specific type of privilege escalation.
  - The server can make all clients provide capabilities (demonstrating that they have access) to all objects used in processing client requests to avoid confused deputy problems. [Accept other answers regarding restrictions on open/openat]
- ~~2. What attacks can a man-in-the-middle (MITM) launch against network communication? Why does the encryption of data thwart MITM attacks on TCP communications, even though the IP address and port are not encrypted?~~
  - ~~A MITM can eavesdrop, drop, modify, and insert packets. Pretty much anything.~~
  - ~~Encryption of packet data protects communication, because even though MITMs can modify header information arbitrarily to launch attacks, they cannot produce packet data that will decrypt and correspond to a meaningful communication with a non-trivial probability. Also, there is an integrity check of packet data, which will likely fail.~~

Homework (Part II)

- 3. If a worm attack can find a subject to attack in 3 time units and takes 1 time unit to compromise the subject to propagate the worm, how many subjects will be controlled by the worm in 64 time units, assuming no subjects are attacked multiple times?

- 4. What does an attacker need to be authorized for in order to launch a file squatting and a link traversal attack? How can a programmer ensure that they open the expected object?

Homework (Part II)

- 3. If a worm attack can find a subject to attack in 3 time units and takes 1 time unit to compromise the subject to propagate the worm, how many subjects will be controlled by the worm in 64 time units, assuming no subjects are attacked multiple times?
  - 2^(64/3+1) = 2^16
- 4. What does an attacker need to be authorized for in order to launch a file squatting and a link traversal attack? How can a programmer ensure that they open the expected object?
  - An adversary only needs write permission to any directory used in resolving a pathname presented by a victim.
  - The best way to prevent these attacks today is to use “openat” from the current working directory of the program. The kernel maintains the current working directory, so that is trustworthy as a validated “dirfd.” Then, any “path” can be restricted using O_NOFOLLOW (no symlinks) and O_EXCL and O_CREAT (no prior file on creation).

Homework (Part II)

- 5. What fundamental problem enables adversaries to launch SQL injection attacks? How does a programmer prevent this problem from manifesting in their web applications?

- 6. How does a Trojan horse enable an adversary to violate confidentiality? Why does multi-level security prevent such leaks?

Homework (Part II)

- 5. What fundamental problem enables adversaries to launch SQL injection attacks? How does a programmer prevent this problem from manifesting in their web applications?
  - SQL injection attacks mix code and data in SQL queries, which when optimized for execution, change the expected code based on the data provided by an adversary. For example, an adversary can insert comment symbols in the SQL language as data to remove SQL code from the query in the optimization step.
  - The goal to prevent SQL injection attacks is to use a ”prepared statement”, which is optimized first and then apply any untrusted data in the query later, as a parameters.
- 6. How does a Trojan horse enable an adversary to violate confidentiality? Why does multi-level security prevent such leaks?
  - A Trojan horse pretends to be a legitimate application, so it may be used to process secret information. Once the Trojan horse has access to secrets, it will try to leak the secret data by writing it to some resource that can be accessed by the adversary.
  - Multi-level security is an information flow policy that ensures that a process that can access “secret” data cannot “write down” to “public” resources, as no information flows from “secret” processes to “public” resources are allowed.

Homework (Part III)

- Suppose that the filesystem has the following permissions:

Subject	Object	Operations
`Subject-1`	`/dir1`	`RW`
`Subject-1`	`/dir2`	`R`
`Subject-1`	`/dir3`	`RW`
`Subject-2`	`/dir1`	`R`
`Subject-2`	`/dir2`	`RW`
`Subject-2`	`/dir3`	`RW`

- 1. Draw the information flow graph that results from these permission assignments.
- 2. Which directories have permission assignments that allow Subject-2 to attack Subject-1?
- 3. Identify a directory in the example that may be used by Subject-2 to successfully launch a file squatting attack against Subject-1. Specify an example attack and why this attack could be successful.

Homework (Part III)

- Suppose that the filesystem has the following permissions:

Subject	Object	Operations
Subject-1	/dir1	RW
Subject-1	/dir2	R
Subject-1	/dir3	RW
Subject-2	/dir1	R
Subject-2	/dir2	RW
Subject-2	/dir3	RW

- 1. Draw the information flow graph that results from these permission assignments.

Subject-1 --> /dir1 (write)	/dir1 --> Subject-1 (read)	/dir2 --> Subject-1 (read)
Subject-1 --> /dir3 (write)	/dir3 --> Subject-1 (read)	
/dir1 --> Subject-2 (read)	Subject-2 --> /dir2 (write)	/dir2 --> Subject-2 (read)
Subject-2 --> /dir3 (write)	/dir3 --> Subject-2 (read)	

Homework (Part III)

- Suppose that the filesystem has the following permissions:

Subject	Object	Operations
`Subject-1`	`/dir1`	`RW`
`Subject-1`	`/dir2`	`R`
`Subject-1`	`/dir3`	`RW`
`Subject-2`	`/dir1`	`R`
`Subject-2`	`/dir2`	`RW`
`Subject-2`	`/dir3`	`RW`

- 2. Which directories have permission assignments that allow Subject-2 to attack Subject-1?

/dir2 and /dir3 – these can both be written by Subject-2 and can be read by Subject-1

Homework (Part III)

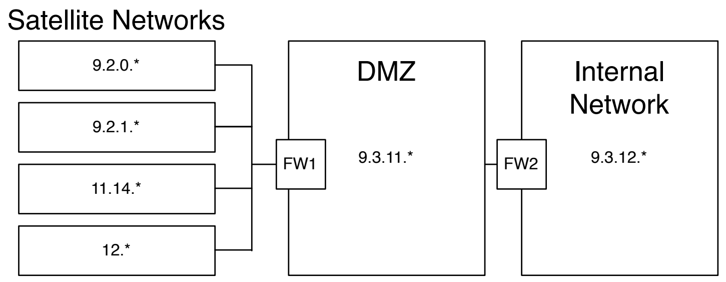
- Suppose that the filesystem has the following permissions:

Subject	Object	Operations
`Subject-1`	`/dir1`	`RW`
`Subject-1`	`/dir2`	`R`
`Subject-1`	`/dir3`	`RW`
`Subject-2`	`/dir1`	`R`
`Subject-2`	`/dir2`	`RW`
`Subject-2`	`/dir3`	`RW`

- 3. Identify a directory in the example that may be used by Subject-2 to successfully launch a file squatting attack against Subject-1. Specify an example attack and why this attack could be successful.
  - To create a file, Subject-1 (the victim) must have write permission to the directory. Thus, /dir3 is the mostly likely target of a file squat attack.
  - However, Subject-2 could create a file in /dir2 and grant Subject-1 write access to that file. Subject-1 would have to make the mistake of trying to create a file in a directory in which it only has read permission (i.e., normally cannot create a file).

Homework (Part IV)

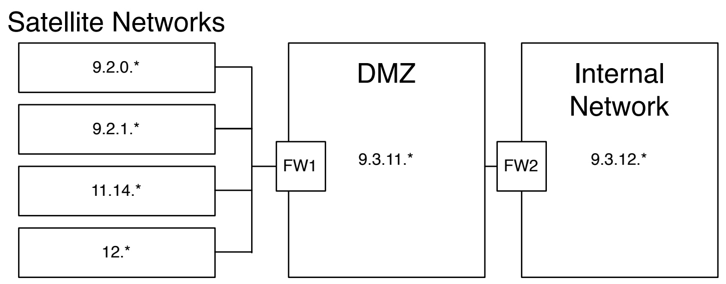
- Given the firewall configure shown, answer the following questions.



- 1. Suppose that hosts in the 9.3.12.* network want to connect to web servers in the 11.14.* network. What rules are necessary in FW1 to allow such communication?
- 2. Suppose that hosts in the 9.3.12.* network want to connect to web servers in the 11.14.* network. What rules are necessary in FW2 to allow such communication?
- 3. Suppose we want to allow external clients to access a web server at 9.3.11.19. What firewall rules are necessary in FW1 and FW2?

Homework (Part IV)

- Given the firewall configure shown, answer the following questions.

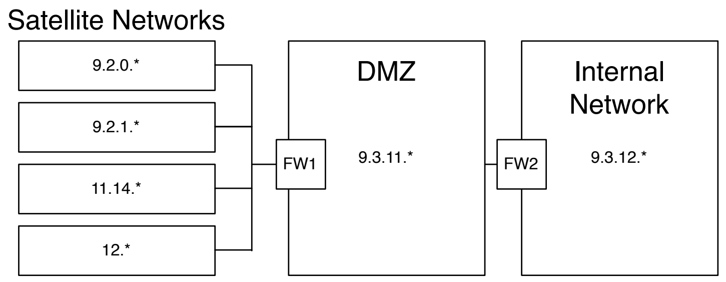


- 1. Suppose that hosts in the 9.3.12.* network want to connect to web servers in the 11.14.* network. What rules are necessary in FW1 to allow such communication?

src IP	src port	dest IP	dest port	action
9.3.12.*	*	11.14.*	80	allow
11.14.*	80	9.3.12.*	*	allow

Homework (Part IV)

- Given the firewall configure shown, answer the following questions.

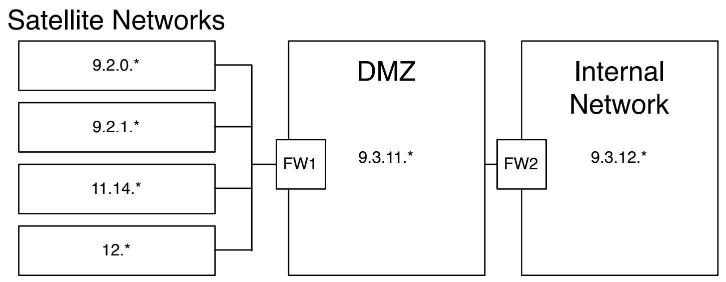


- 2. Suppose that hosts in the 9.3.12.* network want to connect to web servers in the 11.14.* network. What rules are necessary in FW2 to allow such communication?

src IP	src port	dest IP	dest port	action
9.3.12.*	*	11.14.*	80	allow
11.14.*	80	9.3.12.*	*	allow

Homework (Part IV)

- Given the firewall configure shown, answer the following questions.



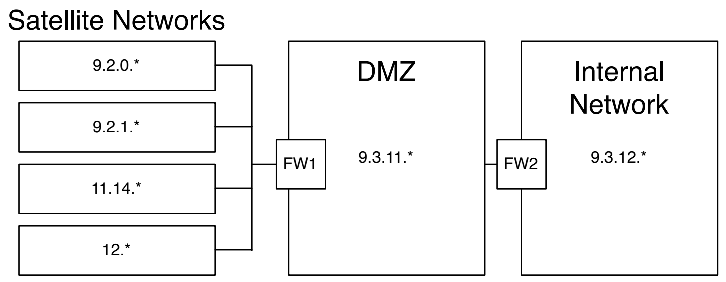
- 3. Suppose we want to allow external clients to access a web server at 9.3.11.19. What firewall rules are necessary in FW1 and FW2?

FW1:

src IP	src port	dest IP	dest port.	action
*	*	9.3.11.19	80	allow
9.3.11.19	80	*	*	allow

Homework (Part IV)

- Given the firewall configure shown, answer the following questions.



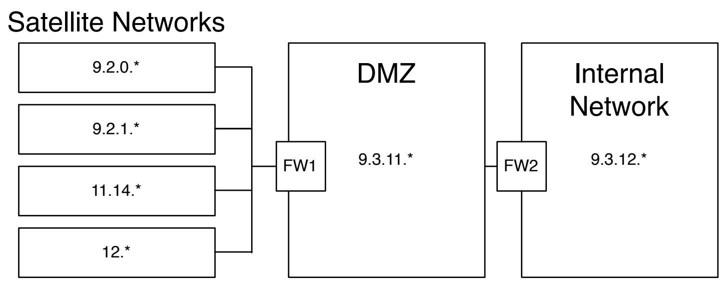
- 3. Suppose we want to allow external clients to access a web server at 9.3.11.19. What firewall rules are necessary in FW1 and FW2?

FW2:

src IP	src port	dest IP	dest port	action
(none)				

Homework (Part IV)

- Given the firewall configure shown, answer the following questions.

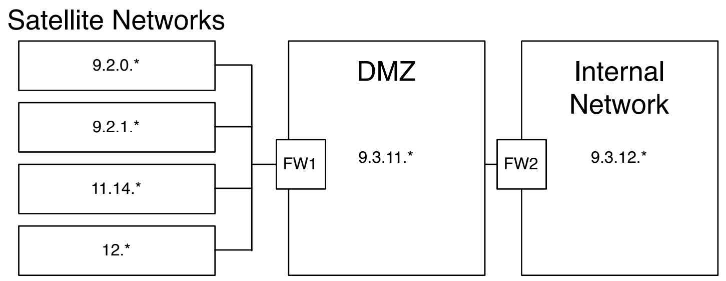


- 4. What is a problem with having the following rules in your firewall?

src_ip	src_port	dest_ip	dest_port	action
9.3.12.*	*	12.4.*	80	deny
9.3.12.8	*	12.4.11.7	80	allow

Homework (Part IV)

- Given the firewall configure shown, answer the following questions.



- 4. What is a problem with having the following rules in your firewall?

src_ip	src_port	dest_ip	dest_port	action
9.3.12.*	*	12.4.*	80	deny
9.3.12.8	*	12.4.11.7	80	allow

- Ans: The first rule will block the packet before the second rule is run - shadowing

Homework (Part V)

- Suppose that website `bank.com` has the following files and access rights.

Subject Label	Object Label	Authorized Operations
`httpd_process`	`config`	`RW`
`httpd_process`	`content`	`RW`
`httpd_process`	`user_data`	`RW`
`httpd_process`	`code`	`RX`
`web_appl`	`config`	`--`
`web_appl`	`content`	`R`
`web_appl`	`user_data`	`RW`
`web_appl`	`code`	`RX`

- 1. Suppose the adversary knows that the web application uses the name of an argument "file" at web address "https://bank.com/app" to modify a file with value assigned to "input". Describe how an adversary could use cross-site request forgery (e.g., produce a XSRF request) to cause `bank.com` to modify a file labeled as "user_data".

Homework (Part V)

- Suppose that website `bank.com` has the following files and access rights.

Subject Label	Object Label	Authorized Operations
`httpd_process`	`config`	`RW`
`httpd_process`	`content`	`RW`
`httpd_process`	`user_data`	`RW`
`httpd_process`	`code`	`RX`
`web_appl`	`config`	`--`
`web_appl`	`content`	`R`
`web_appl`	`user_data`	`RW`
`web_appl`	`code`	`RX`

- 1. Suppose the adversary knows that the web application uses the name of an argument "file" at web address "https://bank.com/app" to modify a file with value assigned to "input". Describe how an adversary could use cross-site request forgery (e.g., produce a XSRF request) to cause `bank.com` to modify a file labeled as "user_data".

Suppose you know that "path" refers to a file labelled "user_data" then send the request: https://bank.com/app.html?file="adv-chosen-path"&input="adv-data"

Homework (Part V)

- Suppose that website `bank.com` has the following files and access rights.

Subject Label	Object Label	Authorized Operations
`httpd_process`	`config`	`RW`
`httpd_process`	`content`	`RW`
`httpd_process`	`user_data`	`RW`
`httpd_process`	`code`	`RX`
`web_appl`	`config`	`--`
`web_appl`	`content`	`R`
`web_appl`	`user_data`	`RW`
`web_appl`	`code`	`RX`

- 2. Suppose the web server's configuration file is labeled "config" and the web application runs as the label "httpd_process." Suppose the web application fails to sanitize the client input used to construct a file name by the web application. How can an adversary exploit that flaw to access the "config" file instead?

Homework (Part V)

- Suppose that website `bank.com` has the following files and access rights.

Subject Label	Object Label	Authorized Operations
`httpd_process`	`config`	`RW`
`httpd_process`	`content`	`RW`
`httpd_process`	`user_data`	`RW`
`httpd_process`	`code`	`RX`
`web_appl`	`config`	`--`
`web_appl`	`content`	`R`
`web_appl`	`user_data`	`RW`
`web_appl`	`code`	`RX`

- 2. Suppose the web server's configuration file is labeled "config" and the web application runs as the label "httpd_process." Suppose the web application fails to sanitize the client input used to construct a file name by the web application. How can an adversary exploit that flaw to access the "config" file instead?

If the web application runs as "httpd_process," it already has the permissions to modify files labelled "config," so the adversary just needs to provide a path to such a file.

Homework (Part V)

- Suppose that website `bank.com` has the following files and access rights.

Subject Label	Object Label	Authorized Operations
`httpd_process`	`config`	`RW`
`httpd_process`	`content`	`RW`
`httpd_process`	`user_data`	`RW`
`httpd_process`	`code`	`RX`
`web_appl`	`config`	`--`
`web_appl`	`content`	`R`
`web_appl`	`user_data`	`RW`
`web_appl`	`code`	`RX`

- 3. If the web server process (running as “httpd_process”) launches the web application under the label “web_appl” instead is an attack on “config” files possible via the flaw above in the web application? Explain.

Homework (Part V)

- Suppose that website `bank.com` has the following files and access rights.

Subject Label	Object Label	Authorized Operations
`httpd_process`	`config`	`RW`
`httpd_process`	`content`	`RW`
`httpd_process`	`user_data`	`RW`
`httpd_process`	`code`	`RX`
`web_appl`	`config`	`--`
`web_appl`	`content`	`R`
`web_appl`	`user_data`	`RW`
`web_appl`	`code`	`RX`

- 3. If the web server process (running as "httpd_process") launches the web application under the label "web_appl" instead is an attack on "config" files possible via the flaw above in the web application? Explain.

No, while the adversary can direct the web application to open the "config" as before, the MAC policy will deny access to the "config" file.

Homework (Part VI)

- **VI. Design a static analysis to detect buffer overflows in the following program. Need to identify the abstract values for the variable *len* used to access the array *buffer*. Then, need to determine the operations that set/modify the value of *len* and use *len*. Then, need a rule to detect when an access is out-of-bounds (OOB). Is there an OOB access via your static analysis if `SIZE = 10`? Is there really an OOB access allowed in the code in that case?**

```
int main( int argc, char *argv[] ).{

  char buffer[SIZE];

  int len = 0;

  if ( argc == 1 )  len += 7;

  else               len += 2;

  if ( argc > 3 )   len += 5;

  else              len -= 2;

  if ( argc < 10 )  len += 2;

   else              len -= 5;

 snprintf( buffer, len, “%s”, argv[1] );
```

Homework (Part VI)

- **VI. Design a static analysis to detect buffer overflows in the following program. Need to identify the abstract values for the variable *len* used to access the array *buffer*. Then, need to determine the operations that set/modify the value of *len* and use *len*. Then, need a rule to detect when an access is out-of-bounds (OOB). Is there an OOB access via your static analysis if `SIZE = 10`? Is there really an OOB access allowed in the code in that case?**

```
int main( int argc, char *argv[] ).{

  char buffer[SIZE];

  int len = 0;

  if ( argc == 1 )  len += 7;

  else               len += 2;

  if ( argc > 3 )   len += 5;

  else              len -= 2;

  if ( argc < 10 )   len += 2;

   else              len -= 5;

 snprintf( buffer, len, "%s", argv[1] );
```

Values: Len is integer value (concrete)

Ops: Set via += and used in snprintf

Rule: If (use(len) && len > SIZE)
error

Homework (Part VI)

- **VI. Design a static analysis to detect buffer overflows in the following program. Need to identify the abstract values for the variable *len* used to access the array *buffer*. Then, need to determine the operations that set/modify the value of *len* and use *len*. Then, need a rule to detect when an access is out-of-bounds (OOB). Is there an OOB access via your static analysis if `SIZE = 10`? Is there really an OOB access allowed in the code in that case?**

```
int main( int argc, char *argv[] ).{

   char buffer[SIZE];

   int len = 0;

   if ( argc == 1 )   len += 7;

   else                len += 2;

   if ( argc > 3 )    len += 5;

   else               len -= 2;

   if ( argc < 10 )   len += 2;

    else              len -= 5;

  snprintf( buffer, len, "%s", argv[1] );
```

Is there an OOB access via your static analysis if SIZE = 10?

Answer: Yes, because static analysis does not consider the results of Conditionals by default (expensive)

E.g., assume all "if"s are true

Homework (Part VI)

- **VI. Design a static analysis to detect buffer overflows in the following program. Need to identify the abstract values for the variable *len* used to access the array *buffer*. Then, need to determine the operations that set/modify the value of *len* and use *len*. Then, need a rule to detect when an access is out-of-bounds (OOB). Is there an OOB access via your static analysis if `SIZE = 10`? Is there really an OOB access allowed in the code in that case?**

```
int main( int argc, char *argv[] ).{

  char buffer[SIZE];

  int len = 0;

  if ( argc == 1 )   len += 7;

  else                len += 2;

  if ( argc > 3 )    len += 5;

  else               len -= 2;

  if ( argc < 10 )    len += 2;

   else               len -= 5;

 snprintf( buffer, len, "%s", argv[1] );
```

Is there really an OOB access allowed in the code in that case?

Answer: Appears not. Static analysis over-approximates program executions

All "if"s cannot be true.

Memory Error Defenses

- We have discussed some
  - Canaries
  - Address Space Layout Randomization
  - Data Execution Protection (No Execute)
- How do these defenses work? Review

Memory Error Defenses

- We have discussed some
  - Canaries
  - Address Space Layout Randomization
  - Data Execution Protection (No Execute)
- These defenses do not prevent ROP attacks
  - Why not?

Memory Error Defenses

- We have discussed some
  - Canaries
  - Address Space Layout Randomization
  - Data Execution Protection (No Execute)
- These defenses do not prevent ROP attacks
  - Why not?
    - Bypass canaries and ASLR
      - Disclose canary values on stack
      - Disclose stack pointer values (EBP)
    - DEP/NX does not prevent execution of code memory

Conclusions

- Structure of exam
  - Multiple choice – fill in blank
  - Short answer – Conceptual questions
    - May be more than one question – be sure to answer all
  - Constructions – Problem solving
    - Multiple sub-parts
- Time management – should not be an issue
- Topics – Covered in these slides
  - And mentioned.
- Readings – good to know more – different angle

Questions