Example Exam 1

1. **Make a comparison between static and dynamic techniques for security analysis, highlighting pros and cons of each one of the two classes of techniques.**
   Static Analysis
   – Analyze system (documentation/code etc.)
   – No need to have a running/concrete system
   – Examples: auditing, formal verification
   Dynamic Analysis
   – Test a running instance of the system (testing)
   – Examples: penetration testing
2. **Explain the meaning of the following Proverif output**
   Query inj-event(eC(x_1,y_1,z)) ==> inj-event(bS(x_1,y_1,z)) is true.

   the represented query must be proven to be true (query → property). In this case we talk about correspondence property (order relationship that should bind trace events) between the two events eC and bS. This correspondence is injective, i.e each occurrence of event eC(x_1, y_1, z) corresponds to a distinct occurrence of event bS(x_1, y_1, z)

3. **In the context of the Common Criteria, what are Evaluation Assurance Levels? In what cases an EAL level is higher than another one?**
   It quantifies the assurance achieved. Each EAL requires a set of components. A higher EAL is obtained from the previous one by: including other components (other families) and replacing components with higher level assurance components (same family). An EAL is higher than another one when we have an higher risk situations.

4. **Describe what is software obfuscation, how does it work, and which risks it mitigates.**
   Obfuscation is a software protection technique whose purpose is to make the code more difficult to understand for a potential attacker, hence it is used to delay an attack. For this reason, obfuscation is a form of security-through-obscurity by performing "anti-static" analysis.
   Obfuscation can be done using several techniques based on the desired purpose and the risks they mitigate. Some examples:
   - make the control flow unintelligible → control flow flattening, branch functions, hide external calls
   - add bogus control flow → opaque predicates
   - data obfuscation → forms that hide constants and values, white-box cryptography to hide keys in code
   - manipulate functions to hide their signatures → split/merge

- avoid static reconstruction of the code and force dynamic analysis → just-in-time techniques, virtualization obfuscation, self-modifying code.

5. **Describe the "Data Execution Prevention" protection and indicate which kind of attacks it aims at countering.**
   Data Execution Prevention (DEP) is a security feature that monitors and protects certain pages or regions of memory, preventing them from executing (usually malicious) code. So starts from the idea that harmful programs can try to attack OS by attempting to run code from system memory locations reserved for OS and authorized programs. When DEP is enabled, all data regions are marked as non-executable by default. When DEP is disabled, vulnerabilities such as buffer overflow can leave your code exposed to an injection of malicious code within these regions that can then be executed.

6. **List at least 5 of the OWASP top-10 vulnerability classes**
   OWASP top-10 stands for Open Web Application Security Project: it's a periodic publication that identifies the top 10 most critical security risks for web applications and it's based on statistics about known vulnerabilities. The 2021 top-10 is:
   1) broken access control, e.g violation of least privilege principle
   2) cryptographic failures, e.g use of weak algorithms for confidentiality
   3) injection: hostile data are sent to an interpreter without validation or filtering. E.g: SQL injection, PHP injection, XSS (Cross-Site Scripting)
   4) insecure design: security is not properly considered in the development process (e.g no threat modeling)
   5) security misconfiguration: security features are not properly configured. E.g: XEE (XML external entities) attack due to weakly configured XML parser, which can lead to disclosure of sensitive data
   6) vulnerable and outdated components: third-part components used by web applications which run with the same privileges as the main application
   7) identification and authentication failures, e.g due to weak credentials
   8) software data integrity failures, which typically occurs during software updates and can lead to insecure deserialization, i.e when a web application accepts data in form of serialized objects from untrusted sources and it doesn't serialize them securely
   9) security logging and monitoring: insufficient logging or monitoring (unclear log messages or even absence of logging system)
   10) server-side request forgery SSRF: when a web application fetches a remote resource without validating the user-supplied URL → used to perform unintended operations on resources protected by firewalls or VPNs.

7. **When Flawfinder runs on the following C code**

```
#define MAXLINE 1024
#define MAXWORD 512
int processFile(FILE *fp) {
   char line[MAXLINE];
   char word1[MAXWORD], word2[MAXWORD];
   while( fgets (line, MAXLINE, fp)!=NULL ) {
         if (sscanf(line, "%s %s", word1, word2) != 2) {
         return 1;
         } else
         process(word1,word2);
   }
    return 0;
 }
```

it produces the following output:

```
FINAL RESULTS:
test1.c:9:  [4] (buffer) sscanf:
         The scanf() family's %s operation, without a limit specification,
         permits buffer overflows (CWE-120, CWE-20). Specify a limit to %s, or
         use a different input function.
test1.c:5:  [2] (buffer) char:
         Statically-sized arrays can be improperly restricted, leading to
         potential overflows or other issues (CWE-119:CWE-120). Perform
         bounds checking, use functions that limit length, or ensure that the
         size is larger than the maximum possible length.
test1.c:6:  [2] (buffer) char:
         Statically-sized arrays can be improperly restricted, leading to
         potential overflows or other issues (CWE-119:CWE-120). Perform
         bounds checking, use functions that limit length, or ensure that the
         size is larger than the maximum possible length.
```

Considering the code and the Flawfinder output, answer the following questions:

1. **Does the code contain buffer overflows? If so, at what lines? If so, write a fixed version of the code.**

   The code contains buffer overflow at line 9 because one of the substrings could be longer than MAXWORD (e.g: line is composed by one string of 1000 chars and one of 23 chars). The problem is fixed if a limit in substring dimension is specified. Fix:

   sscanf(“%511s, %511s”, word1, word2);

2. **Are there any false positives in the warnings produced by Flawfinder? If so, at what lines?**

   There code contains one false positive at line 5 (declaration of line[MAXLINE]), because the string is filled in the fgets() function, which will never insert more than MAXLINE characters because the limit is specified.

3. **Are there any false negatives in the warnings produced by Flawfinder? If so, at what lines?**

   The code doesn't contain false negatives

Example Exam 2

1. **What is reachability analysis? How does it work? What are its limitations?**

   In the context of model checking of dynamic systems reachability analysis is a simplified form of state exploration. More precisely, reachability analysis is done through explicit model checking tools and it's used to verify the temporal logic property []P, i.e analyze all states and verify that P holds in each state.
   Limitations:

   - works well only if the number of states is finite, because this technique implies the exploration of all the states. Hence when dealing with infinite states models some abstractions are needed
   - for concurrent system there's the risk of state explosion, i.e the complexity grows exponentially with respect to the number of parallel components

2. **A company has performed a Vulnerability Assessment and identified several vulnerabilities. Propose a strategy to prioritize the correction of the vulnerabilities (hint: follow the NIST suggestion).**

3. **Describe the advantages and limitations of the fuzzers when used for security verification and testing.**
   The fuzzer is an automatic input generator used to trigger a bug in the program. The advantage is that fuzzying allows detecting bugs and improves security testing complements usual software testing procedures. On the other hand, it may not be able to give enough information to describe bugs,

requires significant resources and time, can only detect "simple" faults or threats and it does not work in detecting unwanted behaviors.

4. **What are security evaluation standards? What are the Common Criteria?** Security evaluation standards are in charge of process and product evaluation. In particular we use Common Criteria to compare between the results of independent security evaluation, providing a common standard reference for evaluating/certifying IT system security. Common Criteria born as a fusion of similar national standard.

5. **Here, f*or your convenience, you can find a definition of shared-key cryptography and signatures in the typed Proverif syntax:***

   (Shared-key cryptography *)*

   fun senc(bitstring, bitstring): bitstring.

   reduc forall x: bitstring, y: bitstring; sdec(senc(x,y),y) = x.

   (Signatures *)*

   fun ok():result.

   fun sign(bitstring, skey): bitstring.

   reduc forall m:bitstring, y:keymat; getmess(sign(m,sk(y))) = m.

   reduc forall m:bitstring, y:keymat; checksign(sign(m,sk(y)), pk(y)) = ok().

   Following the same approach and language, write a definition of the symbolic functions that represent the computation and the verification of Message Authentication Codes (MAC). A MAC is a code computed from a message and a symmetric key, with the purpose of enabling integrity verification. The integrity of a message with its MAC can be verified by using the same key that was used to generate the MAC.

   (* MAC *)

   type mkey; #not needed

   fun ok():result.

   fun MAC(bitstring, mkey):bitstring. #just use the already used above symmetric key type “skey”

   reduc forall m:bitstring, k:mkey, y:bistring; checkMAC(MAC(m,k),k) = ok() # was missing k as parameter for check MAC, in order to check a mac you need the message, the MAC, and the sharedKey to recalculate the MAC on the message you received and compare

   Please, check if this solution is ok :)

6. **Consider the following code, for which has produced two warnings at lines 18 (execute Query, can be vulnerable to SQL injection) and 25 (println, could be vulnerable to XSS).**

```
@WebServlet(value=*/test2*)
public class Test2 extends HttpServlet{
        private static final long serialVersionUID =1L;
@Override
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, I0Exception{
        response. setContentType("text/html;charset=UTF-8");
        String user « request.getHeader("USER");
        String pass request. getHeader ("PASSWORD");
        user - java. net. URLDecoder. decode(user,"UTF-8");
        pass • java. net. URLDecoder. decode(password,"UTF-8");
        String sql:
        if (user. equals("adm") && pass.matcher("^[a-zA-Z0-9_]+$")){
                sql = "SELECT from USERS where USERNAME ='+user+'and
                PASSWORD='+ pass+'":
                try{
                        java.sql. Statement statement
                        statement. executeQuery( sql ):
                        response. setStatus(200):;
                        }catch (java. sql.SQLException e){
                response. setStatus(500);
                response. getwriter().printin("Error processing request. "):
        }else{
                response. setStatus(400):
                response.getWriter() .println('Bad request for user'+ user):
        } //end if
} //end doPost
}
```

For each one of the two warnings raised by , tell whether it is a true positive or a false positive, and explain why.

Exam 25/01/2021

1. **con del codice**

2. **domanda su proverif (non ricordo nel dettaglio cosa purtroppo)**

3. **linear sweep disassembler**

4. **Rop Vs ret2libc**

5. **Xss varie tipologie e cosa è**

6. **cosa è un transition system e quale è la sua formalizzazione**

Exam 08/02/2021

1. **6 pt.**

2. **Proverif 6 pt.**

3. **Anti-tampering 6 pt.**

4. **EPM (Enterprise Patch Management)5 pt.**

5. **Non-determinism 3 pt.**

6. **CCRA 4 pt.**

7. **0-day vulnerability 3 pt.**

Exam 02/07/2021

1. **Explain what kind of software analysis can be performed by means of the 'angr' tools. Describe the functions of the loader (CLE), explain what are the simulation states and how they are used by the Simulation Manager. Finally, explain how to configure the execution of the unwanted branches.**

2. **Can Proverif find false attacks? Explain why.**
   Yes, because if proverif ends without finding a proof, it starts to perform a state exploration in order to look for an attack, this may lead into a false attack.
   #wrong, <u>proverif can find attacks which are possible only in theory but not in practice, as it turns all conditions into horn clauses, it loses the order of some operations and as such it can report vulnerabilities that aren't replicable in the real world.</u>
   <u>Better explained:</u> Yes, proverif can find false attacks. Because the logic theory described by the Horn clauses over-approximates the behavior of the real protocol, false positives are possible. In fact the approximations have been

proved sound, but not complete. The set of traces that can be derived according to the Horn clause model is greater than the set of traces that are really possible

3. **Describe what is a buffer overflow vulnerability. Furthermore, explain how a buffer overflow is n function can be exploited to execute a remote shell.**

4. **Explain what is a control graph and what is a basic block.**

5. **What is responsible disclosure? Provide a precise definition and an example scenario of responsible disclosure.**

6. **Explain how SQL injection vulnerabilities works. How can it be fixed/prevented?**

7. **Describe what is a buffer overflow vulnerability. Furthermore, explain how a buffer overflow is n function can be exploited to execute a remote shell. When Flowfinder runs of the following code:**

```
#include <stdin.h>
#include <string.h>
#define MAXWORD 512
int process();
int processInput(char *input, int off) {
        char word[MAXWORD];
        int length = strlen(input);
        if (off < 0) {
                off = -off;
        }else if (length > MAXWORD) {
                return -1;
        }
        if (off >= MAXWORD) {
                strcpy(word, input);
        }else{
                strncpy(word, input, off);
                word[off] = '\0';
        }
        return process(word);
}
```

It produces the following output (4 hits):

1) **test2.c:20:[4] (buffer) strcpy:**
   Does not check for buffer overflows when copying to destination (CVE-120).

Consider using strcpy_s, strncpy or strlcpy.

2) **test2.c:9:[2] (buffer) char:**
   Statically sized arrays can be improperly restricted, leading potential overflows or other issues (CVE-110; CVE-120).
   Perform bands checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.
3) **test2.c:11: (buffer)**
   Does not handle strings that are not '\0' terminated; if given one it may perform an over-read (CVE-126).
4) **test2.c:22:[1] (buffer) strncpy:**
   Easily used incorrectly; doesn't always are not '\0' terminated a check for invalid pointers (CWE-120).

Consider the code and the Flowfinder output, answer

1) **Does the code contain buffer overflows? If so, at what lines? If so, write a fixed version of the code.**

2) **Are there any false positives in the warnings produced by Flowfinder? If so, at what lines?**

Exam 30/08/2021

1. **Same origin policy Basile**

2. **Format String attack Basile 6 punti**

3. **Strong secrecy property dì Proverif. Definizione formale**

4. **Temporal logic formula, fare esempio. 3 Punti**
   it is a sort of extension of classic logic programming where time is considered as a variable that leads to changes in the system

5. **Security Assessment vs Security Certification.**
   in order to perform a security assessment of a system it is required to verify if there are any vulnerabilities and their related exploits. a mix of different techniques can be used like vulnerability assessment and penetration testing. for security certification we need to certify the security of a system through evidence that we reached security requirements following some metrics for security assurance

6. **Esercizio**

7. **Cos'è Taint Analysis**

Exam 31/01/2022

1) exercise on a java code. It is a TP or FP? If it is a TP provide a possible solution to solve it.

2) Proverif exercise.

```
type key.
type host.
type nonce.

free c:channel.
free s:bitsting [private].

fun encrypt(bitstring;key):bitstring.
reduc forall x: bitstring, y: key; decrypt (encrypt(x,y),y)=x.

let pA(A:host, B:host, kS: key)=
        new Na: nonce;
        out (c, A,B,Na);
        in(c, m:bitstring);
        let (=Na,=A,=B, k:key)=decrypt (m,kS) in; 0
        out (c,encrypt(s,k));

let pB(B:host, kS:key) =
        in(c,x:bitstring);
        let (xA: host, xNa:nonce) = x;
        new k:key;
        out(c, encrypt(xNa,xA,B,k), kS);
        in (c, x2:bitstring);
        let sec:bitstring = decrypt (x2,k) in
        0.
```

-What kind of encryption is represented by functions encrypt and decrypt? What are the properties of this kind of encryption in the symbolic model?
(To check but are also words during the correction)

We use the same key for encryption and decryption, in common between client and server.
In symbolic model we don’t model attacks such as brute force attack guessing the secret key
Affected by MITM(Man in the middle) attack

- Describe the behavior of the protocol specified by these processes definitions with your own words
The process pA starts sending to process pB its identity, the identity of pB and a nonce.
The process pB takes the parameters and check if the identity B is equal to be sure that process A wants talk with it.
Process pB creates a new key k and then encrypts this key, the nonce, its identity and the identity of the process A with kS.
pB process sends all the pA process that decrypt with the kS keys that the 2 process have in common and verify if the nonce and the identity.
Finally the pA process encrypt the secret with the key sent by the process pB.
This is done to send a shared key k for the process that will be used to encrypts the secret.

3) What is model checking and how does it works? In this context, what is a counter example?

4) In the context of Common Criteria, what is an Evaluation Schema.
(not 100% sure) An evaluation schema is a predefined framework that every nation defines to regulate the criteria of evaluation and its corresponding values, such as target of evaluation, protection profiles, security targets, etc. So it’s an administrative framework that works as a guide to make the evaluation of the security certification of a system.

(Dalla correzione 3/3)
An evaluation schema is a report to perform some test. Only the Authorized Nations of the CCRA have an Evaluation Scheme to evaluate and certificate, but to do this the Nation must add the CC and the methodology, like CEM. The Consuming Nations does not have an Evaluation Scheme and trust the evaluation scheme done by the Authorized Nation.

5) SAST, DAST and IAST what are their definitions? Explain pros and cons.
Sast:
- (Static Application Security Testing)

- White-box static analysis of source code

Pro and cons:

- Find more vulnerabilities
- More false positives
- Used in all development stages
- Can provide info on the cause of vulnerability (code line)
- Coverage of libraries is an issue
- Each tool applies to only some languages/frameworks
- May be time-consuming

Dast:

- (Dynamic Application Security Testing)
- Black-box dynamic analysis (vulnerability scan)

Pro and cons:

- Find less vulnerabilities
- Less false positives
- Used in the last development stages
- Cannot provide info on the cause of vulnerability (black-box)
- Independent of language/technology used to develop application

Iast:

- (Dynamic Application Security Testing)
- Black-box dynamic analysis (vulnerability scan)
- Recent concept that tries to combine static and dynamic analyses
    - Motivation: complexity of applications (especially web apps), made of many third-party components
    - Main ideas
        - Agent that runs inside a running application (code instrumentation)
        - Agent has access to code and all execution details, but only the code that is triggered is analyzed
        - Agent can run continuously (even during operation)
        - Reports are live (continuously generated/updated)

  Pro and cons:

    - Find many vulnerabilities (like SAST) but with few false positives
    - Very fast and scalable (good for dev-sec-ops)
    - Not yet available for any language/framework
    - Not yet widely known and adopted

6) [???= idk what is correctly written] You find a program, running on it root, which is only accessible from localhost on a specific port. By decompiling the code, you notice that the input is managed by a function named and reported below:
[code (illeggibile)]
You also disassembled:

- **get_input_from_socket which takes up to 1024 bytes from this socket and copies??? in the memory location pointed by the var1 pointer**
- **printf??_on_socket which replicates?? on the socket the output of the printf (i.e. using the sprintf?? functions)**

Given the code shown before:

1. **Which kind of attack looks very promising, as it ?? not depend on array protection enabled on the program server?**
2. **Which security property is easy to endanger??? with this attack?**
3. **What other attack is possible whose success depends on the absence of protections that may have been enabled by who compiled the application?**
4. **Imagine these protections are not active on the running program, what do you have to send to the server to try a privilege escalation??? ? Describe the structure of the payload to send (presume it has been compiled for 64-bit architecture). List an informal syntax but very precise on the number of byte (you can use *python-like???* or any other syntax provided this behavior its unambiguous??? or explain it)**
5. **Which protections may render you attack impossible?**

Exercise in C code that performs a gets on a variable var1 (char of size 128).

-what kind of attack is possible?
-??? (I don't remember)
-??? (maybe ask for a second possible attack if there aren't some other security properties)
-explain how to perform this second attack, how to do the payload (I'm not sure lol)
-how could you avoid this attack?

7)Software Attestation and Code Guards, definitions, pros and cons. When it is better to use one over the other?

software attestation verify that a program running on another system is behaving as expected. -> pro: achieves execution correctness. cons: it's difficult to have evidences of execution correctness.
code guards are pieces of code injected into the application in order to perform checking, for example the hash of bytes in memory or hash of executed instructions for unconditional code. -> if it fails prevents the correct execution of the rest of the application.
cons: the correct values are somewhere in the code so they are vulnerable to attacks that change the execution environment.

Exam 14/02/2022

1. **When Flawfinder runs on the following C code**

```
void function (char *name, char* value){
        char buffer[100];
        if (strlen(name)>94)
                return -1;
        strcpy(buffer,name);
        strcat(buffer,":%s\n");
        print(buffer,value);
}
```

   it produces the following output (5 hits):

```
FINAL RESULTS:
test1.c:12: [4] (format) printf:
If format strings can be influenced by an attacker, they can be exploited
(CWE-134). Use a constant for the format specification.
test1.c:6: [2] (buffer) char:
Statically-sized arrays can be improperly restricted, leading to potential
overflows or other issues (CWE-119:CWE-120). Perform bounds checking,
use functions that limit length, or ensure that the size is larger than the
maximum possible length.
test1.c:11: [2] (buffer) strcat:
Does not check for buffer overflows when concatenating to destination
(CWE-120). Consider using strcat s. strncat, or stricat (warning, strncat
is easily misused). Risk is low because the source is a constant string.
test1.c 8: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may
perform an over-read (it could cause a crash if unprotected) (CWE-126).
test1.c:10: [1] (buffer) strncpy:
Easily used incorrectly doesn't always \0-terminate or check for invalid
pointers (CWE-120).
```

   Consider the code and the Flawfinder output, and focus attention only on the top 2 hits (the ones at lines 12 and 6).

Assuming that "value" is tainted while "name" is not,
1. is the hit at line 12 a real security vulnerability (TP) or not (FP)? Explain why.
FP
2. is the hit at line 6 a real security vulnerability (TP) or not (FP)? Explain why.'
FP
Assume now that "name" is tainted while value is not
3. is the hit at line 12 a real security vulnerability (TP) or not (FP)? Explain why.
TP
4. is the hitit line 6 a real security vulnerability (TP) or not (FP) Explain why
FP

2. **What is the meaning of the security properties defined in the script? How could the process definitions be fixed to make the second property true? (write the fixed version of the process)**

```
free channel (Be public channel)
free bitstring [...] (shared secret)

("Cryptography hash")
query attacker(s)

event send(bitstring)
event receive(bitstring)
query x bitstring.event(receive(x))==> event(send(x))

let p(A) =
        new ds bitstring
        event send(da)
        out(c ,(da, hash(da,s))
        0

let p(B) =
        in (c, y bitstring)
        let ( db bitstring,  hashb bitstring) == y in
        let ch=hash(db,s) in
        if ch==hashb then ok()
        event receive (db)
        0

process
```

```
{
!pA | !pB
}
```

3. **What is security assurance in the context of security assessment and certification standards? Give a precise definition**
   What is the main techniques that can be used to achieve assurance? Make at least 3 concrete examples of such techniques (of different types)

   Confidence that an entity meets its security requirements.
   Techniques:
   - **Use of security mechanisms**
   - **Use of a development methodology**
   - **Use of security assessment techniques**

4. **What does the percentage on the X represent? How can it be computed? What does the percentage on the Y represent? How can it be computed? Explain why the tools usually have scores that lay in the left upper area of the diagram (the white one)**

5. **After having initialized the project *p* and execute the following commands:**
   es = p.factory.entry_state()à
   sm = p.factory.simulation_manager(es)
   sm.explore (find=lambda s : b"Correct result:" in s.posix.dumps(1), avoid=0x0804871e)
   The output of the tool is:
   <SimulationManager with 3 active, 1 found, 10 avoid>
   First, discuss what tool has been used and what kind of analysis has been performed.
   Then, identify the component in charge of doing this analysis, and provide (informal yet precise) details on how this analysis is been conducted.
   Finally, explain the meaning of the output produced by the tool.

   The tool used: angr
   The analysis performed: conclic analysis
   The component of the tool actually used: simEngine
   More details on how the analysis is performed: the Simulation Manager starts from the entry state (starter point of execution of a program), attach a simulation manager. The simulation manager start the execution of symbolic

block and stop itself when it must transform the block into concrete or in case of branches.
Explain the meaning of:
- **active:** When there are branch, they start in parallel (that's why there are 3 active).
- **found:** the one found by the matching of this condition (it found in the stdoutput "Correct result").
- **avoid:** the branches discharged because entered in this address of memory

6. **Explain what is the corpus minimization used by the fuzzers (e.g. AFL), the purpose, and the results it permits to achieve.**
   Moreover, illustrate one more technique that can be used for improving the performance of fuzzers, shortly indicating purposes and results.
   Finally, discuss why and how fuzzers can be used by a company developing software for vulnerability assessment, what they help to discover, and who benefits from using them.
   Corpus minimization: useful to start from the seeds of a fuzzer of type mutational
   Purpose: remove duplicate
   Result: a corpus smaller / improve the performance because the mutation are easier

   Technique name and short description: use dictionary or models to make it more accurate the coverage/parallelization
   Purpose: a technique to speed up the fuzzer
   Result:

   Fuzzers for VA, why:
   Fuzzers for VA, how:
   what discovers:
   who benefits:

Exam 02/05/2022

1. **Consider the following excerpt of Java code on which has reported one warning at line 36: 6 punti**
   @RestController
   @ RequestHapping (value = '/testshopping Cart")
   public class Test8{

```
        @RequestMapping (method = RequestMethod.POST, value =
        "/shoppingCart/(userId)/products,produces = "application/json")
public String addProductToCart(PathVariable("userId") String userid,
@Request Paran String productId) throws SQLException
try {
        Integer.parseInt(productId);
        if (checkProduct (productId))
                return Cartservice.addProduct (userId, productId);
        else
                return null;
} catch (Number FormatException e) {
        return null;
}
}

private boolean checkProduct (String productId) throws SOLException {
        Connection connection - Cartservice.getConnection();
        Statement stmt;
        stat = connection.createstatement();
        ResultSet rs - stmt.executeQuery("select from products where id=' " +
        productId + " ' ");
        if (rs.next())
                return true;
        else
                return false;
}
```

1. What kind of security warning can be expected at this line?

SQL injection because the productId (which is tainted) may be inserted in the executeQuery without checks

2. Tell if it is a true positive (TP) of a false positive (FP), and explain why

It is a FP because the try catch of parseInt will give an exception if the string is not a parsable integer.

2. **Proverif script contains the following two queries:**

   query x host, y host; inj-event(endB(x,y)) => inj-event(beginB(x,y))

   query x hosl, y. host; event(endB(x,y)) => event(`(x,y)).

   When Proverif runs, it finds that the first query is false and the second one is true.

   a. Explain the meaning of these results

   The first one means that the injectivity correspondence of the events is false. The second one means that the correspondence of the events is true.

b. According to these results, if in a run of the system we have n occurrences of event endB(A,B), how many occurrences of begin(A,B) may we have in the same run?
Since the injectivity correspondence is false, we may have that n events of endB can correspond to 1<=x<=n occurrences of event beginB.

3. **What goal can an attacker achieve by exploiting XSS vulnerability? Make a concrete example 5 punti**
   By exploiting XSS vulnerability it is possible to avoid the SOP (Same-Origin-Policy) and/or sandboxes of browsers. In this way, an attacker can inject malicious code at the victim's browser and, e.g., perform session hijacking, user impersonification, stealing cookies, etc..
   This is possible because even if SOP exists, some tags (eg., img, script) can follow links of other origins.

   There are three types of XSS vulnerability: reflected XSS, stored XSS, DOM XSS.

   An example is the one of the Reflected XSS: suppose to have a vulnerable website which accepts some parameters in the URL (e.g. www.example.com/?name=example) without checking them and shows them in the page to the user (e.g., "Hello example!"). An attacker could create a malicious URL by putting inside the parameter a script code (e.g., www.example.com/?name=<script>...</script>) and could send it to the victim (e.g., using email). If the victim clicks on the link, then the browser will send back the page populated with the parameters (e.g., Hello <script>..</script>!), and so the victim will execute the malicious code.

4. **Explain what is Theorem proving and how it works. In this context, provide a formal definition of what a proof is. 5 punti**
   Explain what is Theorem proving and how it works:

R1: Theorem Proving is a formal verification technique used to prove theorems. It receives as input the theory T, which is a formal system composed of a formal language (e.g., descriptive logic) + a deductive apparatus that is a set of axioms and inference rules, but receives also a formula f that is the one to be proved. The theorem prover answers the question "|- f ?" which means "given the theory T, is the formula f a theorem?" and it can answer Yes by giving a proof, otherwise it can't say anything (because maybe a proof has not been found but the formula is a theorem). Furthermore, it requires also human assistance.
A technique for theorem proving is the logic programming, mainly used with horn clauses.

R2: The theorem proving is a formal verification technique. It works in this way: it receives as input a T theory, a wff f, human assistance. The idea is to verify that the formula f is true for each interpretation. The theorem proving doesn't work with state exploration, but it works with a deductive apparatus. The theorem proving gives a proof in case the formula is true for each interpretation. Instead, in the contrary case it cannot give us any information because is not known is the formula is true or not.

Provide a formal definition of what a proof is:

R1: A proof is a set of well-formed formulas P1…Pi that for each Pi there is an axiom that say that the wff is true, or there is an indirect inference rule.

R2: A proof is a set of well-formed formulas P1…Pn such that each formula Pn is an axiom or a direct consequence of an inference rule.

5. **Describe what is an Enterprise Patch Management system and which risks it helps to mitigate. Moreover, present the additional risks the use of an Enterprise Patch Management system adds and how severe they are.**

   What Is an Enterprise Patch Management system:

   R1: The EPM system is the process for identifying, acquiring and applying patches in a company. It is based on three advices: prioritize, timing, testing.
   Prioritize because it is important to understand the vulnerabilities that affect our system and prioritize based on their severity (patch first the higher ones).
   Timing because the higher is the risk, the faster should be the application of the patch.
   Testing because it is important to test patches before applying them.
   The aim is to reduce the time spent for patch application, helping companies to use their resources to address other security problems.

   R2: An Enterprise Patch Management system is a tool that is in charge the distribute updates of softwares to different machines. It is useful to save time and human resources. It has different features, like deciding
   when, to who, and what patch to apply based on different approaches, it may be Agent-Based, Agentless or by scanning targets to look if there are vulnerabilities.

   The risks it mitigates:

   It is used expecially in companies, where there are usually many machines that are wanted to keep secure, it may cause confusion and time loss if done manually, meanwhile with this approach it is easier to maintain secure the assets by checking systems for vulnerabilities and patching them when needed.

   Additional risks and their severity:

R1: The problem of using an EPM system is that with a patch an attacker could perform reverse engineering and find where the problem was, in order to attack the unpatched assets.
A patch could cost a lot because when a patch is released, the system could need a reboot or maybe be down for a while.
A way to mitigate these risks is to use EPM tools (but they add other risks!).

R2: Although it is safier to use this approach than not, it has some risk to take in consideration. One example is an attacker that has control over the patch that will be sent from the EPMS, the severity of this risk is very high because it may disrupt the entire system and causing a very big loss for a company

Other version:
what Is an Enterprise Patch Management system:
It is a process to identifying, acquiring, installing, verifying patch for programs or systems. The key concepts are: prioritization, timing. The problem related to patch management are security risks and costs. It is based on a phased approach:
SUBSETS: small groups are patched first
STANDARD first
DIFFICULT ones later.
Resort to manual activity when automated tools are not available
The risks it mitigates:
Mitigate risks related to security, fixing possible vulnerabilities that could affect assets and resources
Additional risks and their severity:
Bugs or flaws in the patching tools could lead to privilege escalation

6. **Describe what is a recursive disassembler, list its features and discuss its limitations. 5 punti**

R1: A recursive traversal disassembler follows the real execution flow in order to reconstruct the assembler code. Since it follows the execution steps, it gets rid of data embedded into code sections and also of data bytes since they are not executed. This is the most used type of disassembler.
The problems are:
- Since the disassembler follows the execution, only the executed parts of code are read. This creates a CFG that is not complete. One possible solution is to start again the analysis from the unreached code, in order to get many CFG that then need to be built into one bigger CFG manually.
- Indirect jumps do not allow to understand statically where the execution will jump (so CFG may appear disconnected)

R2: A recursive traversal disassembler is a type of disassembler that disassembles that opcodes following the execution flow. The first step is a depth analysis in order to understand the execution flow. The features of this disassembler are that it has no problems with data embedded in the code sections, and that it skips data bytes since they are not executable. The limitations are that if there is unreachable code, it will not decode it, so it will not be a complete decode. Another problem is that sometimes the understanding of the execution flow is hard, in particular in case of indirect jumps that make the CFG hard to build.

R3: A recursive disassembler is the one that considers the real execution flow, not as the linear sweep that considers the opcodes just in order. This kind obviously is better because when something happens that makes the execution disaligned with the order of the opcodes (for example overlapping instructions) you don't have incorrect results. The problem is sometimes the execution could skip some branches because they could be executed or not and this depends on the real data, so in this case you can identify this branches that are not taken and perform a disassemble on that one. But this means that you will have different CFGs and you need to manually merge them (waste of time).