Software Security

Final Exam Preparation

Note: This document contains the questions from the final exam on 09.06.2017. Additionally potential questions about Combinatorial Web Security Testing and Decentralized Trusted Computing are listed.

Be aware, there is no guarantee for the correctness of the answers!

Final Exam 09.06.2017

1) Briefly outline the underlying concept behind Proof-of-Work and how the difficulty can be adjusted? (2p)

The concept behind Proof-of-Work is a consensus algorithm allowing nodes in the network to collectively agree on a set of updates to the state of the blockchain. The weight of a single node in the consensus voting process is directly proportional to the computing power it brings in.

In Bitcoin:
The precise condition is that the double-SHA256 hash of every block, treated as a 256-bit number, must be less than a dynamically adjusted target. The purpose of this is to make block creation computationally "hard", thereby preventing Sybil attackers from remaking the entire blockchain in their favor. Because SHA256 is designed to be a completely unpredictable pseudorandom function, the only way to create a valid block is simply trial and error, repeatedly incrementing the nonce and seeing if the new hash matches.

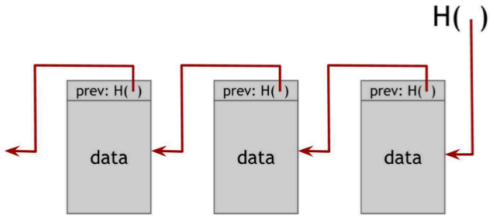
During mining a 32-bit value, the nonce (part of the block header) is changed via brute-force search in order to find a hash value matching the condition described above.

Difficulty is adjusted by number of leading zeros of SHA-256. By adding leading zeros to the target hash the target space is decreased making the PoW more difficult.

2) What is the benefit of chaining hash pointers to create a linked list and how can this type of data structure be called? (1p)

Blockchain: Blocks (elements of the linked list) contain the digest (hash) of their predecessor, so one can verify that previous elements have not changed.

- Confirms the integrity of all prior blocks
- Alter one block and all subsequent blocks must be altered as well!



3) Briefly outline the mining (and state transition) process in Ethereum that is followed by all nodes to create/validate a new block in the chain. (3p)

Each transaction changes the state of the system (blockchain is the archive and chain of custody for all state transitions). The changed state is computed by every node in the Ethereum network (might differ from node to node due to synchronization). Every node executes every transaction to verify it, but only one node (the miner) gets rewarded for this work. Only the state of the miner node (mined a valid block) is accepted by all others (honest) nodes after they have received and validated it. Honest nodes build up on the previously mined block, which they consider valid and try to find the next valid block on top of it.

The state transitions are recorded and synchronized via blocks.

4) What is the difference between externally controlled accounts and contract accounts in Ethereum? (1p)

Externally controlled account	Contract account
• has no associate code • has associated private/public key pair • controlled by its private key (wallet/client)	• has associate code • has no directly associated private/public key pair • is entirely controlled by its contract code

5) You are tasked with writing and ID3v2 tag editor. Because the specification contains length-value fields, you assume that this is a context-sensitive grammar. Your boss requests the following two features:

a. **Your program should reject files that do not contain valid ID3v2 tags.**
b. **Your program should parse ID3v2 tags in exactly the same way as the main competitor's product.**

Which of these tasks are feasible, which are not? Why? (2p)

ad a) Feasible. A so-called recognizer, which rejects non-conforming inputs and transforms conforming inputs into structured data could be generated using a parser generator that takes the specified grammar as input. By doing so input handling is separated from further processing.

ad b) Infeasible. Since our input language specification is stronger than deterministic context-free, namely context-sensitive the problem of establishing parser equivalence is undecidable.

Interesting read about language security[1]

6) Write a grammar for a simple datetime format in any BNF variant (BNF, EBNF, ABNF). It should contain at least the following fields: Year, month, day, hour, minute. The start symbol should be called "datetime". You are free to validate the content of these fields afterwards (for instance, the value of 'minutes' field produced by the parser does not have to be restricted to [0,59], but it must be a number. (3p)

```
datetime     = [ day-name "," ] date time
day-name     = "Mon" / "Tue" / "Wed" / "Thu" / "Fri" / "Sat" / "Sun"
date         = day month year
day          = 1*2DIGIT
month        = "Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun" / "Jul" /
               "Aug" / "Sep" / "Oct" / "Nov" / "Dec"
year         = 2*4DIGIT
time         = hour ":" minute [ ":" second ] zone
hour         = 2DIGIT
minute       = 2DIGIT
second       = 2DIGIT
zone         = ( "+" / "-" ) 4DIGIT
```

7) Give the definition of a test oracle and provide one for testing the authentication functionality of a website, where the oracle has the form of a requirement. (1p)

Def. Test Oracle: A mechanism for determining whether a software system or system under test (SUT) has passed or failed a test case.

Test Oracle for testing the authentication functionality of a website:
Successful login of a user (requirement)

8) Explain what is security testing and penetration testing. How do they differ? (2p)

Security Testing: A process to determine that a software system protects data and maintains functionality as intended

Penetration Testing: The process of simulating attacks on a network and its software systems, at the request of the owner or senior management

Difference:

- Penetration testing occurs once software is complete and installed in its operational environment

[1] http://langsec.org/bof-handout.pdf

- By contrast, security testing can be applied before the software is complete

9) Explain how black-box testing and white-box testing are applied to penetration testing. Which are their respective goals? (2p)

Black-box Penetration Testing:

- refers to methodology where a hacker has no knowledge of the system being attacked
- **Goal:** simulate external hacking

White-box Penetration Testing:

- hacker has full knowledge of the system being attacked
- **Goal:** simulate attack of a malicious insider

10) Assume you are given a combinatorial attack grammar for XSS having k types and g derivation rules per type to form an attack vector. Which of the following is more cost effective in terms of combinatorial testing. Adding more types or more derivation rules per type in the grammar? Justify your answer. (Hint: For an SUT with x variables and y possible values per variable, the number of test cases in combinatorial testing is proportional to $y^t \log x$) (3p)

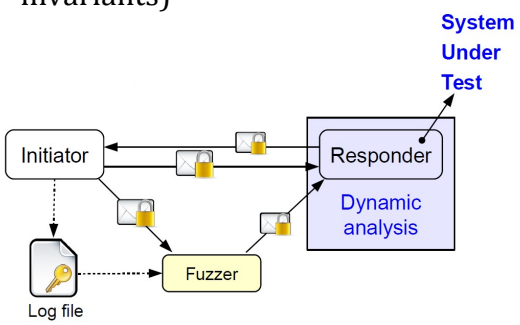
$$g^t \log k$$

Adding more types k in the grammar increases in logarithmic scale the number of vectors.
Adding more derivation rules per type will result in a polynomial growth.
=> It is more cost effective to add more types in the grammar since a logarithmic growth is smaller than a polynomial.
$O(\log n) < O(n^c)$; for $c > 0$

11) Describe two testing methods for TLS/SSL implementations. (2p)

Fuzz Testing:

- Collect well-formed inputs
- Mutate the inputs (fuzz operators)
- Execute the inputs and check for failures (e.g. memory errors, broken invariants)



Model-/Combinatorial-/Differential Testing:

- Model X.509 certificates
  - Identify and model parameters and parameter values
  - Model the structure
- Create a testsuite testing all t-way interactions using combinatorial approach
- Use differential testing approach: send input certificates to N implementations of the same protocol and record their classification

12) You are given the following (sample) configuration options for a TLS Cipher Suite:

Sample Test Suite for TLS Cipher Suite Registry		
Key Exchange Algorithm	**Encryption Algorithm**	**MAC**
RSA	3DES	MD5
RSA	AES	SHA256
ECDHE	3DES	SHA256
ECDHE	AES	MD5

Which testing method is most likely to have been employed to generate the test suite? Justify your answer. (3p)

Combinatorial testing for 2-way interactions was used to generate the test suite. All possible combinations of 2-way interactions are covered - take two arbitrary columns of the testsuite and you can see that all combinations: {00, 01, 10, 11} are covered.

0 0 0
0 1 1
1 0 1
1 1 0

Combinatorial Web Security Testing

What is Software Testing?

- Testing is an important but expensive part of the software development process
- The problem is to design a test plan for a software system

Provide the definition of a test plan. Give such an example for a security application.

Test Plan: is a logical collection of (abstract) test cases

Example:
- Assume you have some test cases that test the authentication functionality of a website
- Then you can group all those test cases under an authentication test plan

What is the definition of a Test Case and a Test Suite? Give an example Test Case for testing a security application.

Test Case: A sequence of actions under which a tester will test the correct behavior of a software system (or one of its features)

Example:
Purpose: Test the authentication functionality of a website
Test Case: (Login=dsimos,password=qwerty,protocol=ssh)

Test Suite: A collection of test cases (i.e. a test plan) that are intended to be used to test a software system

What is the difference between Test Coverage and Fault Coverage?

Test Coverage: refers to measuring how much a software system has been exercised by tests

Fault Coverage: refers to the percentage of some type of fault that can be detected during the test of a (software) system

What is a Fault/Bug? Explain how a vulnerability relates. Give an example for a vulnerability.

Fault/Bug: An implementation-level software problem that causes it to produce an error

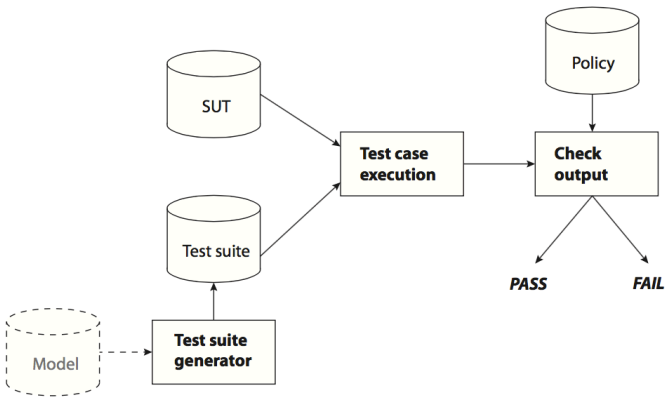
Vulnerability: is a bug that an attacker can exploit
Examples: Input validation errors:
- SQL Injection (SQL-I)

- Cross-sit scripting (XSS)

What Testing Methods do you know? Explain them.

- **Fuzz testing:** Generate random, invalid or unexpected data to the input modules of the SUT
- **Combinatorial testing:** Test (possible) discrete combinations of input modules of the SUT
- **Model-based testing:** A set of techniques and tools to automate the creation and/or the execution of test cases based on a model (usually represented as UML diagram) that describes the behaviour of the SUT

What are the components of a testing framework?



Sketch the combinatorial design process for a test plan.

1. Model the input space; the model is expressed in terms of variables and variable values
2. The model is input to a combinatorial design procedure to generate a combinatorial object which is simply an array of symbols
3. Every column of the generated array is used to output a test case for testing the software system

Explain the parameters of a Covering Array CA(t,k,g) for Test Plans.

- The strength t refers to the number of interactions we want to check
- The rows k of the CA give the different variables of the model
- The number |S|=g gives the number of different variable values, e.g. S={0,1,2}

Test coverage is ensured from the property that in every combination of t rows, each tuple with symbols from S appears at least once.

Assume you want to test the authentication functionality of a website with input modules login, password and protocol and possible outputs per module (root,jdoe), (123,321!) and (SSH,TLS). Give the (general) definition

of a test case and provide two examples of test cases for the previous scenario.

Test Case: A sequence of actions under which a tester will test the correct behavior of a software system (or one of its features)

Examples: (root,321!,TLS) and (jdoe,123,SSH)

What is the difference between training applications and intentionally vulnerable applications for penetration testing? Give some examples of such applications for XSS.

Training applications are designed for learning which guide the user to specific, intentional vulnerabilities while intentionally vulnerable applications have a wide variety of intentional security vulnerabilities, but are designed to look and work like a real application. Some examples of such applications are:

- Webgoat, DVWA (training)
- Gruyere, Bodgelt (intentionally vulnerable)

Compare exploitation rate versus field coverage when testing for XSS.

Exploitation rate refers to how many of the generated XSS vectors exploit a vulnerability in one field parameter of a web application, while field coverage is intended to measure how many of the field parameters of a web application are vulnerable to XSS with a specified attack vector.

Assume you are given a combinatorial attack grammar for XSS having k types and g derivation rules per type to form an attack vector. Which of the following is more cost effective in terms of combinatorial testing. Adding more types or more derivation rules per type in the grammar? Justify your answer.

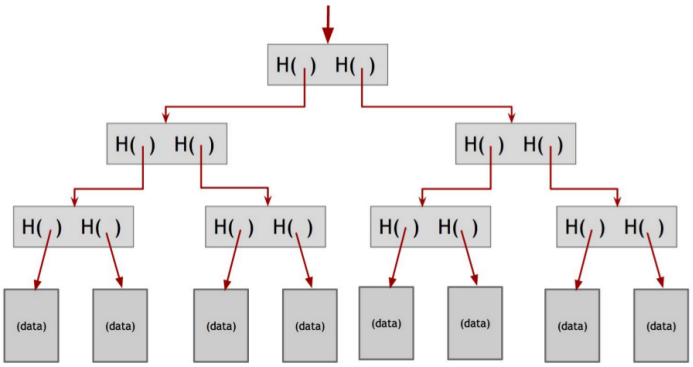
Adding more types in the grammar increases in logarithmic scale the number of vectors, while adding more derivation rules per type will result in an exponential growth. This is due to the fact that the number of test cases for combinatorial testing is proportional to $g^t \log k$, and increases logarithmically with the number of types but exponentially with interaction strength for the number of derivation rules per type.

Note: This is the sample answer from the slides. In my opinion varying g yields in a polynomial growth (not exponential), since the exponent (the interactions) stays constant.

Decentralized Trusted Computing

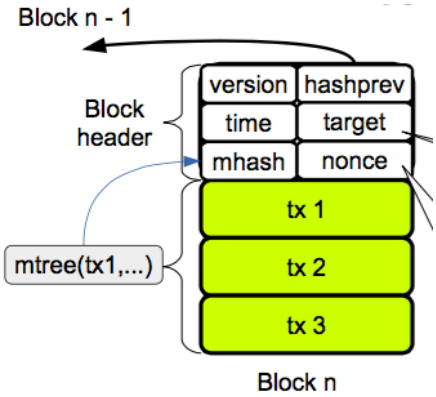
What is a Merkle Tree? Explain its structure and give an example application.

A data structure (tree), where data blocks are grouped in pairs and the hash of each of these blocks is stored in a parent node - all the way up the tree until we reach the root node.



- allows efficient and secure verification of the contents
- demonstrating that a leaf node is part of the given Merkle Tree (hash tree) requires processing an amount of data proportional to the **logarithm** of the number of nodes of the tree. Contrast to hash list where amount is **proportional** to number of nodes!
- Example: Used in a block of the blockchain to store the transactions.

Sketch the structure of a Bitcoin Block.



- mhash: Merkle tree root hash
- mtree: merkle tree of transactions that are included in the block
- hashprev: double-sha256 hash of previous block's header
- target: defines the required number of zeros at the beginning of PoW hash
- nonce: 32 bit value can be changed to find PoW via brute-force search

Outline how Bitcoin Mining works.

Mining nodes attempt to find the doubled SHA256 of the current block header which meets the current difficulty requirement. Therefore the nonce field (a 32-bit value) contained in the block header is modified until a valid hash value is

found (brute force). If a miner finds such a hash value a new block is appended to the blockchain and the miner is rewarded (currently 12.5 BTC).

Outline how a Bitcoin Transaction works.

Bitcoin transactions are based on UTXO (Unspent Transaction Outputs). If Alice wants to send Bitcoins to Bob, block(s) containing transaction(s) with an UTXO to Alice are searched within the block chain.

Alice creates a new transaction and uses the UTXO from previous transaction(s) as input(s) for the new transaction. Therefore the previous output needs to be unlocked to Alice. A signature created with Alice private key is used to unlock the received funds.

- Alice references the output of a previous transaction and signs the new transaction with her secret key.
- Bob's Bitcoin address is used as the output for the new transaction
- The new transaction is sent to "all" Bitcoin users and eventually included in a new block