|  |
| --- |
| **Stimulus: *Intrusion Testing*** |
|  |

Digital Technologies project: folio

**Test 1: Phishing test**

Generatea script that tells a user their account has been compromised, and then asks the user to change their password. The script should ask the user to input their old password, then their new password, then retype their new password for confirmation (to appear legitimate). Don’t worry if the new passwords don’t match – the target here is the *old* (i.e. current) password.



|  |  |
| --- | --- |
| **Extension** | **Description** |
| **Target a persons name** | Allow the user using this test to specify a target username: |
| **Loop multiple targets, and produce a report** | Run this test within a *loop*, so that you have a few usernames and old (i.e. current) passwords, and then output a report summarising your findings as follows: |

**Test 2: Common password test**

Generatea script that takes three common passwords: "qwerty", "abc123", and "password", and tests all 3 against 3 user accounts: "joe", "zoe" and "lee". The test should respond with "correct" or "incorrect" password.

* It can be assumed that "joe" has the password "abc123"
* It can be assumed that "zoe" uses 1 of these common passwords
* It can be assumed that "lee" does not use these common passwords



|  |  |
| --- | --- |
| **Extension** | **Description** |
| **Less verbose code** | Use loops and "if" statements to make your code less verbose. Try to code this using a loop to test the common passwords, *within* a loop that tests each username.This extension challenge isn’t really about adding any features, but making sure your code is as efficient as possible. |
| **Scalable sample sizes** | Generate this script so that the user using it can add:* More usernames (for example "ben" or some others)
* More common passwords to test (for example "123456" or more)
 |

**Test 3: Brute force pin test**

Generatea script that loops through all pin numbers from 1000 to 9999 inclusive. The script should stop at a *random* number within this range, that is different on every test:



|  |  |
| --- | --- |
| **Extension** | **Description** |
| **Zeros at the start** | Include the digits 0000 (through to 9999) in the test range. For example, 0645 and 0090 are valid pin numbers, however 645 and 90 are not. |
| **Custom pins** | Create variables so that the user using this script can specify:* Number of digits in the pin (any number between 4 and 16 inclusive)
* Start and end range to test (for example, only 6 digit pins between 100000 to 200000)
 |

**Test 4: Avoiding detection and compromise test**

A [cyber honey trap](https://www.kaspersky.com.au/resource-center/threats/what-is-a-honeypot) (also known as a “*canary*”) is an unused physical or virtual service or device set up on a network that alerts whenever it detects activity. It can lure hackers into thinking it is real.

For this exercise, we will assume that canary accounts (which we want to avoid during intrusion testing) will only be set up with *less than $50 credit*, due to their inactive nature. If this is the case, we will flag these as canary accounts, and not count their funds as potentially compromised.

However, If the account contains $50 credit or more, we will add that total to the amount of funds that could be compromised in a cyber breach.

For this task:

* generate 5 accounts with random balances between 0 and $100
* flag the canary accounts (those with less than $50)
* sum the remaining amounts as potential funds compromised:

|  |  |
| --- | --- |
| **Extension** | **Description** |
| **Add options to customize the test** |  |
| **2c at a time** | There is an urban myth that due to cash rounding and unnoticeable trace amounts, withdrawing 2 cents from multiple accounts may reward a hacker a low risk profit.To prove or disprove this premise, for all available accounts (non-canary), withdraw 2 cents from each, showing the remaining amount, and total profit made. You may have to scale out to a large number of accounts to test: |

**Test 5: Network discovery**

A network map can be used by hackers to gain access to uncontrolled or vulnerable devices or ports on a system.

For this exercise, we will map the IP network addresses of all of the devices a test-case financial service has:

* 150 IP addresses set aside for end user devices (workstations, tablets, phones, either connected wired or wirelessly)
* 12 printers
* 10 switches
* 8 wireless access points
* 7 servers:
	1. domain controller (Active Directory) server
	2. file and print server (combined)
	3. data server
	4. web server
	5. mail server
	6. application server
	7. proxy server
* 1 router (with built in firewall)

The devices will be connected in the IP range of 192.168.0.0 >> 192.168.0.255 (so you only have to change the last number). Generate a script that lists all of the devices in this range. You can allocate the hardware to the IP pool as you wish:



|  |  |
| --- | --- |
| **Extension** | **Description** |
| **Add options to customize devices** | In future, the number of devices may change. Include options to customise this network map based on future expansion. Your code will have to be scalable and efficient to do this. |

**Bonus Test: Cheque writer**

This test is *purely for enjoyment* if you finish the previous tests. It is a test that is far more difficult than the previous tests, and it is not expected that you will finish this to a high or adequate functioning level.

***You do not have to complete this test*.**

Any attempts to this test, like all tests, will be evaluated as per the criteria for this assignment.

In the previous test, you discovered that the print server was vulnerable.

It turns out this print server prints bank cheques, where dollar value amounts are printed as words on bank cheques.

To do this, you must write a script that inputs an amount in currency format (e.g. 20.00), and then prepares this number as a written sentence that can be written on a cheque (e.g. TWENTY DOLLARS AND ZERO CENTS).

A completed sample of this test is available [here](https://digisoln.com/otherports/chequewriter).

Generate a script as best you can that can replicate this functionality.

END OF ASSESSEMENT

|  |
| --- |
| **Submission Requirements** |
| * Complete each test in a new file.
 |
| * Submit all commented source code files digitally.
 |
| **Important Notes** |
| * Visually simulate any necessary tasks that cannot be coded (such as load wait times).
 |
| * A visual simulation could be as simple as an on-screen prompt, such as “Now downloading ... (waits 5 seconds) ... Download complete.”
 |
| * Use comments in code to explain understanding of programming structures, as well as pointing out refinements and on-going testing of code.
 |
| * Keep backups of your files. Save every 10-15 minutes of work.
 |
| * Testing, refinements or recommendations should be neatly commented in your source code files.
 |
| **Getting Started**  |
| * Look at the examples from class to get an idea how to tackle these challenges.
 |
| * Look through resources from the website for ideas.
 |
| * Brainstorm some ideas with your friends or teacher if you can’t figure out how to start.
 |
| **Authentication Strategies** |
| * Acknowledge any and all code snippets, tutorials, advice, information or help given.
 |
| * Students may be asked to explain their solution, or parts there-of, to determine authenticity.
 |
| * Please do not share your solutions but rather help students with their own line of work – **you may not necessarily be right or efficient**.
 |

#### Appendix A: QCAA Years 9 and 10 Digital Technologies standard elaborations (contextualised)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **A** | **B** | **C** | **D** | **E** |
| **Processes and production skills** | *Generating and designing; producing and implementing* | **purposeful** design and **proficient** implementation of modular programs | **effective** design and **effective** implementation of modular programs | design and implementation of modular programs | **partial** design and implementation of modular programs | **fragmented** design and implementation of modular programs |

*This will be marked digitally via the submission platform.*

#### Appendix B: Australian Curriculum content descriptions

This assessment instrument is used to allow students to formally demonstrate the following Australian Curriculum Digital Technologies Years 9 and 10 Content Descriptions:

|  |
| --- |
| **Explicitly measured** |
| P&PS | Implement modular programs, applying selected algorithms and data structures including using an object-oriented programming language |
| **Implicit to the task** (not formally measured) |
| K&U | Investigate the role of hardware and software in managing, controlling, and securing the movement of and access to data in networked digital systems |
| K&U | Analyse simple compression of data and how content data are separated from presentation |
| P&PS | Develop techniques for acquiring, storing, and validating quantitative and qualitative data from a range of sources, considering privacy and security requirements |
| P&PS | Analyse and visualise data to create information and address complex problems, and model processes, entities and their relationships using structured data |
| P&PS | Define and decompose real-world problems precisely, taking into account functional and non-functional requirements and including interviewing stakeholders to identify needs |
| P&PS | Design the user experience of a digital system by evaluating alternative designs against criteria including functionality, accessibility, usability, and aesthetics |
| P&PS | Design algorithms represented diagrammatically and in structured English and validate algorithms and programs through tracing and test cases |
| P&PS | Evaluate critically how student solutions and existing information systems and policies, take account of future risks and sustainability, and provide opportunities for innovation and enterprise |
| P&PS | Create interactive solutions for sharing ideas and information online, taking into account safety, social contexts, and legal responsibilities |
| P&PS | Plan and manage projects using an iterative and collaborative approach, identifying risks, and considering safety and sustainability |

**Key**:

K&U: Knowledge and Understanding

P&PS: Processes and Production Skills