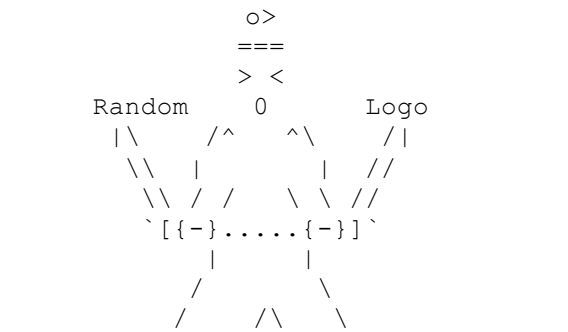

Penetration Testing Report

Performed By: the_hamburgler
Performed For: Ronald_mcdonald

Confidential Information Warning – This document is meant only for use by Company XYZ and should not be disclosed or interpreted out their authorized direction.

Contents

Executive Summary

<replace with – should be in the range of a couple sentences, though a short paragraph is fine. target audience is executives, goal is answer questions such as "where should I invest in next" and "what should I do with this report". So for example stating "in general high risk was found, most of that risk was around the topic or area of controlling access to internal network machines. There is still more to look at, future testing should continue to investigate this.". Do not use security terms, use simple language as possible. Keep in mind the reason of why you are writing this. >

e.g: More generic/App pentest (this is just a explanation/example, naturally delete this from the report)
Overall a low level of coverage was obtained and a significant amount of risk was enumerated. A low level of coverage describes the amount of testing in relativity to the total testing which could be performed, not specifically that there is more known risk. However, it was highlighted that there is risk which a motivated attacker would likely take advantage of to gain access to customer data. These issues appear to be occurrences of a larger risky pattern of behavior to be addressed. It is recommended further testing coverage be allowed to this area and this report be used to help aid the difference in that coverage.

e.g: more red-team or goal oriented pentest (this is just a explanation/example, naturally delete this from the report)
Or alternatively if this pentest has a specific "ask" by the client, such as please emulate an attacker to specifically break into the domain controller, then you would write to describe how successful or unsuccessful that was. So this would be describing first what the ask/goal was, then how easy or hard was to obtain. Then answering the same questions of "where should I invest in next" and "what should I do with this report"

Scope & Coverage

< replace with – this is just an example, naturally orient this towards the point and type of testing. Keep it mind its important ot let your clients know what was tested and what was not tested. Also what context and resources it took this time around so they know what they got for xyz resources.>

Testing was performed from 1/1/2021 – 1/21-2021 for a approximately 14 days of active testing of 1 tester. Environment IP range 192.168.1.1 – 192.168.1.10 was leverage and internal systems, network, and application running within that range were in scope.

Coverage obtained should be considered "Low". The low coverage is largely due to investing into enumerating the prerequisite knowledge, resources, and access needed for testing in the limited time. For most systems a base line of testing was performed such as observing file system permissions and looking for common misconfigurations. This should be considered looking for low hanging fruit that would likely deter a low motivated attacker if not found. A highly motivated attacker with ample time and reasoning could continue to look at deeper details to find additional areas to attack.

High Level Observations & Recommendations

<replace with – high level takeaways from the overall testing. This is different from the executive summary in what question is it answer. This is answer the question of "if I only have enough time or resources to focus on a single or a couple things to improve, what should it be?". Basically you don't have

to be as high level as the executive summary, and is your chance to get more details to a executive or project manager but keeping in mind its still the same audience. >

e.g:
Various individual issues were identified during testing, however despite each requiring a unique fix an underlying theme of improperly configuration of internal systems is apparent. It is recommended that the practice and process of creation of internal systems be invested into and revise to prevent repeat and reintroduction of similar issues. Additionally, risk also appeared to aggregate in web application designs during handling of data which the application does not control. This led to additional opportunities for attackers to steal customer data. It is recommended that a framework be introduce in which developer can leverage to handle such data across any future applications. In this way a secure and consistent behavior can be applied at scale for all applications.

Issues Found

01 – Customer Data and System access can be stolen on /loginpage.aspx (Reflected Cross Site Scripting)

Summary

- Technical Identification: Cross Site Scripting vulnerability in Login Page errors message Modal from URL parameter
- Problem: Customer data or system access can be stolen by unauthenticated attackers having a victim simply view a webpage or click a link
- Fix: Data which the application does not fully control must be protected against for malicious data before being leveraged in the UI code the application.

Relevant Documentation

- Product Version Affected: /loginpage.aspx (version: some_made_up.2.12.2)
- Cross Site Scripting vulnerability: https://owasp.org/www-community/attacks/xss/

Impact & Risk

- Example Attack:
    - An unauthenticated attacker can trivially discover this issue by analyzing the login page of the application. The attacker would modify a URL/link of the login page with malicious data. A user could be tricked into visiting the link directly such as a phishing email or indirectly such as being forcefully redirect from any webpage on the internet. Upon visiting the malicious data from the URL executes as code in the UI of the login page to steal the user's data or access.
- Impact:
    - CVSS Score: CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H
    - Risk and Reasoning: This issue is considered a HIGH risk due to the full impact towards the confidentiality, integrity, and availability of data. While the complexity of the attacking a specific user is somewhat high the impact allows complete control of the victims account.

Recommendations

- Cause: Any data which can be touched by a user, passes through their browser, or can be influenced by a user should be considered "untrusted". Untrusted data is being used and inserted into the code of the UI to populate messages (e.g: "failed to login <insert_username_here>"). However, that data being untrusted, can contain "code". The web browser which displays this to the victim has no way to differentiate the legitimate code of the application from code introduced by the malicious data. As such, it executes and has any capabilities that the application code itself has. Keep in mind the context in this case is all UI code, HTML and JavaScript.
- Fix: User input should be "HTML entity -encoded" at any point where it is used to directly or indirectly populate the UI. All HTML meta-characters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc). This should be performed on the "output" of data, as the data is inserted back into the UI code to be displayed. This should be done for the error messaging as well as in all UI components that reflect back untrusted data.
- Considerations & Learnings
    - It would be beneficial to prevent future occurrence of this vulnerable behavior to adopt development practices and mindsets which identifies data which is not in 100% control by the application. By constantly viewing data through this lens, through this point of view, it can help to naturally lead to more careful thought on how it is used.
    - Refer to https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html for additional details on Cross Stie Scripting.

Reproduction Steps

- Location:
    - endpoint: 192.168.1.1, application: web application forumz_that_i_made_up
    - In Application: /Login.aspx & "special_url_parameter" of URL
    - In Code: testing_something.py – line 332
- How to Tell Reproduction is Working: upon proper recreation of the noted URL and visiting it will result in a message box saying "xss". This is for visual confirmation, it could silently steal data.
- Steps:
    - Visit http://192.168.1.1/login.aspx
    - You will be redirected to http://192.168.1.1/login.aspx?special_url_parameter=12321_abc
    - Copy this URL, in "special_url_parameter" there will be a number underscore then more characters, delete the characters after the underscore and replace it with "<script>alert('xss')</script>"
    - It should resemble *special_url_parameter=12321_<script>alert('xss')</script>* however the number before the underscore will by unique
    - Visit the URL to simulate giving someone a phishing email with a link, clicking on the link should produce a alert box saying "xss"

02 – Low Privileged Users Steal access to High Privileged on Internal Systems from File Permissions

Summary

- Technical Identification: <*replace this with technical details identifying the issue, however one sentence if possible. Audience is developers*.>
- Problem: <*replace this with the high level "why should I care" – the impact and bad behavior happening, don't use security terms, however one sentence if possible. Audience is project managers and those not actively fixing this issue.>*
- Fix: <*replace this with the high level "behavior" that should be happening – allows a general idea of what kind of skill set is needed to work on this, however one sentence if possible. Audience is project managers and those not actively fixing this issue.>*

Relevant Documentation

- Product Version Affected: <add this if relevant>
- Windows Access Controls Considerations: <add details like this as relevent>

Impact & Risk

- Example Attack:
    - <replace with Details Here - concrete example if possible, else theoretical scenario. Target audience is none security people trying to understand the scenario, so likelihood and impact. However it is important to explain in as simple terms as possible and be as realistic as possible>.
- Impact:
    - CVSS Score: <replace with link to CVSS score to help client understand how the industry might view this risk>
    - Risk and Reasoning: <explain the reasoning behind why this might be perceived as high risk or low risk. Its important to give reasoning and justifications, not to prove why a score is right but to give a client the details necessary to do their own risk management prioritization appropriate for their company>

Recommendations

- Cause: <replace with - explain the actual root cause of the issue. Not many do this and say "the devs should know their own code", however the point of view of the "security expert" understanding what components are at play is valuable. For example a non security expert might end up fixing a symptom of a issue vs a root cause. Without the security expert explaining the root cause, if it requires some security knowledge how is a developer supposed to fix a root cause. You want to be explaining the insecure "behavior" and the components attributed to that. >
- Fix: <replace with – what should actually be implemented. Explain the "behavior" we want to see happen to prevent this issue and future issue from happening. Be as specific as possible. >
- Considerations & Learnings
    - <Replace with – add any extra details that are important to know such as how to stop introducing this issue in future application built>

Reproduction Steps

- Location:
    - <replace with where you found the issue, try to be as specific as possible such as to be clear for someone with no context trying to find what you are talking about>
- How to Tell Reproduction is Working: <replace with – we want those whom are not security experts to understand if the issue is fixed vs not reproducing properly vs they are doing something wrong>
- Steps:
    - <replace with – actual steps on how to recreate this issue. Keep in mind developer will not understand "run this attack tool". So as much as possible try to keep your reproduction steps as generic as possible and without dependencies>