Universidad Cenfotec

Maestría en Ciberseguridad

Documento Final de Proyecto de Investigación Aplicada 2

Proyecto: Creación de un prototipo y procedimiento de buenas prácticas para verificación de seguridad en aplicaciones de software

Castillo Herrera, Jonathan

Agosto 2019

Declaratoria de derechos de autor

Dedicatoria

Dedicado a familia y amigos.

Tabla de contenido

Índice de tablas y figuras

Capítulo 1: Introducción

Antecedentes del problema

La seguridad forma parte de todo lo que hacemos en Internet, desde leer un correo hasta comprar un artículo; y se ha adaptado con el avance de las tecnologías, buscando estar adelante de las amenazas informáticas como *hackers* o script *kiddies*. Los bancos de Costa Rica en pleno 2018-2019 emiten tarjetas con chip que contienen la banda magnética, debido a problemas de actualización con datafonos obsoletos; eso implica un riesgo a la seguridad, pues la información de la banda no es encriptada y podría ser clonada por algún tercero, casos de cajeros automáticos han sido reportados debido a que han sido manipulados con este objetivo.

El activo más valioso de una empresa es la información, por esto, una brecha en la información podría ocasionar daños incalculables, que van desde pérdida de reputación hasta demandas legales o cierre del negocio. Dependiendo del tipo de información y objetivos del negocio, la información resguardada puede generar mayor o menor interés por parte de los atacantes, que inclusive, pueden ser miembros de la organización; por ello, todas las aplicaciones deben ser desarrolladas de tal forma que la seguridad sea una característica del producto y no un adicional (Forbes 2018). Se espera que en el presente año 2019 los ataques informáticos lleguen a causar pérdidas de más de 2 billones de dólares.

Métodos de autenticación o cifrado son obligatorios si se quiere contar con los tres pilares de seguridad que corresponden a confidencialidad, integridad y disponibilidad. La confidencialidad es la propiedad que impide la divulgación de información a entes no autorizados. La integridad indica que la información no haya sido modificada sin autorización. La disponibilidad es la característica que asegura la disponibilidad de la información para quienes la requieren.

Justificación

Un procedimiento se define como “seguir una serie de pasos para desarrollar una labor de manera eficaz”, en este caso, el objetivo es el aseguramiento de aplicaciones. El aseguramiento se alcanza mediante una evaluación, por parte de una empresa o persona relacionada con el campo de ciberseguridad, o por el cumplimiento de alguna ley o estándar de la industria.

La pregunta sería: ¿Qué define que sea segura? La seguridad no es un valor cuantificable, por lo tanto, lo que define su seguridad es si se puede garantizar que se utilice para el fin creado, de la forma establecida por las personas indicadas. Para validar este punto, es necesario realizar en el sistema una revisión contra todo tipo de ataques informáticos, revisar los resultados y ejecutar los cambios necesarios con el fin de mitigar el riesgo encontrado.

No es posible asumir la existencia de la seguridad cuando únicamente en ciertos aspectos se cumple con lo necesario, debido a que los atacantes no

se enfocan en las medidas difíciles, sino en buscar puntos débiles los cuales pueden explotar. Por ello, la seguridad debe ser en capas; cada capa agrega controles con el fin de dificultar o disuadir al atacante a no cumplir su objetivo. En el ámbito del software cada cambio realizado puede implicar un riesgo de encontrar alguna vulnerabilidad, por lo que debe ser evaluado constantemente.

Este documento pretende servir como guía para evaluar la seguridad de las aplicaciones, tanto lo automatizable como lo no automatizable. Esto, para que haya seguridad, al realizar cambios, de que la aplicación mantiene los estándares de seguridad aceptados por la industria.

Definición del Problema

La empresa X es una de las más grandes emisoras de tarjetas del mundo, en la empresa ABC se le provee soporte a aplicaciones de software, tanto desarrolladas de cero como heredadas. Por la sensibilidad del tipo de negocio se realizan auditorías periódicas, por lo que se requiere validar el tratamiento de los datos y su ciclo en las aplicaciones.

Se propone definir un procedimiento de buenas prácticas que valide la seguridad de las aplicaciones desarrolladas para la empresa X. Las aplicaciones contemplan varias fases del ciclo de las tarjetas de crédito y débito, que abarcan desde la compra de materia prima, hasta la distribución y

creación de tarjetas. Para demostrar la prueba de concepto, se establecerá un prototipo de una herramienta que realice la validación de manera automatizada, según lo definido en el método.

Viabilidad

Punto de vista operativo

Se cuenta con el visto bueno de la empresa ABC y empresa X para obtener acceso al código fuente de las aplicaciones; también con un ambiente de pruebas donde pueden ejecutarse las aplicaciones que se requieren evaluar.

Punto de vista técnico

El investigador posee estudios en ciberseguridad y actualmente está adquiriendo el conocimiento para aplicar por la certificación *pentester* de COMPTIA.

Punto de vista económico

No se requiere recurso económico, pues el investigador cubrirá el tiempo para adquirir los conocimientos necesarios y desarrollar la investigación.

Objetivo General

- Crear un prototipo y procedimiento de buenas prácticas para verificación de seguridad en aplicaciones de software.

Objetivos Específicos

- Clasificar los controles de verificación de seguridad de la normativa OWASP aplicaciones escritorio y web en automatizables y no automatizables.
- Especificar una guía de verificación de los controles clasificados como no automatizables.
- Especificar una guía de implementación de los controles clasificados como automatizables.
- Establecer un prototipo de una herramienta que verifique los controles automatizables.
- Desarrollar una metodología de presentación de los resultados obtenidos con la aplicación de los controles no automatizables y la autoevaluación de los controles automatizables.

Alcance

La tesis tiene como alcance definir un procedimiento de buenas prácticas para verificación de seguridad de aplicaciones relacionadas con la emisión de tarjetas de crédito y débito, para crear un prototipo de herramienta automatizada. El prototipo no contará con documentación técnica ni de usuario.

El prototipo contendrá únicamente los controles clasificados como automatizables debido a que son los programáticamente viables. No contendrá todos los controles automatizables por limitaciones de tiempo. Los controles automatizables que serán implementados por el prototipo por medio de un análisis estático son: **2.4, 3.3, 5.10, 9.4, 10.11, 11.1.** Los apartados del estándar ASVS de OWASP Móvil 17 y de Servicios Web 18 no serán tomados en cuenta en esta tesis, debido a que ninguna de las aplicaciones del cliente hace uso de estas tecnologías.

La versión a la fecha de la investigación del estándar ASVS de OWASP es la 3.0.1. Según el sitio web la versión 4 se encuentra en desarrollo, pero no se tomará en cuenta pues aún no se encuentra disponible para el momento de la investigación.

Limitaciones

No hay acceso a los servidores donde estas aplicaciones están instaladas para el cliente (ambiente de producción), solo se cuenta con acceso al ambiente de desarrollo, que es una réplica; por ello, no se tomará en cuenta esta variante para el método creado.

Estado de la Cuestión

Para esta tesis se realizó un estado de la cuestión utilizando una revisión sistemática de Bolchini con ciertos parámetros de búsqueda, los cuales son:

Formalización de la pregunta: ¿Cuáles son las mejores prácticas de verificación de seguridad en aplicaciones?
Enfoque de la pregunta: Identificar cuáles son las mejores prácticas de verificación de seguridad en aplicaciones relacionadas con la emisión de tarjetas de crédito y débito.
Amplitud y Calidad de la Pregunta: Este documento se limita a indicar las mejores prácticas para verificación de seguridad en aplicaciones relacionadas con la emisión de tarjetas de crédito y débito.

Problema: Identificar las mejores prácticas para verificación de seguridad en aplicaciones relacionadas con la emisión de tarjetas de crédito y débito.

Pregunta: ¿Cuáles son las mejores prácticas de verificación de seguridad en aplicaciones relacionadas con la emisión de tarjetas de crédito y débito?

Palabras clave: Application, security, best practices.

Control: Ninguno.

Efecto: Identificación de las mejores prácticas para la verificación de seguridad en aplicaciones relacionadas con la emisión de tarjetas de crédito y débito en la actualidad.

Aplicación: aplicaciones web y de escritorio.

Selección de Fuentes

Criterio de selección de Fuentes: Estándar de Seguridad de Aplicaciones de OWASP.

Lenguaje de los estudios: inglés y español.

Identificación de Fuentes

Métodos de Búsqueda de Fuentes: Utilizando motores de búsqueda.

Texto de Búsqueda: “Security applications standards”.

Lista de recursos: Motor de búsqueda duck duck go.

Selección de Estudios

Definición de criterios de inclusión y exclusión de estudios: Los estudios deben indicar cuáles son las mejores prácticas para verificar la seguridad en aplicaciones.

Definición de los tipos de estudios: Estudios relacionados con seguridad de la información.

Procedimiento para la selección de estudios: Se realizarán las búsquedas en el motor de búsqueda web. Se analizaron los documentos para verificar si contienen lo necesario con los criterios de inclusión; en caso de cumplirlos serán seleccionados y analizados.

Ejecución de la selección

Selección inicial de estudios:

Application Security Verification Standard (ASVS)

Evaluación de la calidad de los estudios: El estudio cumple con las condiciones necesarias, pues detalla los requerimientos para desarrollar aplicaciones seguras. Fue desarrollado por OWASP (Open Web Application Security Project) una organización fundada en el año 2001, que busca el desarrollo, adquisición, operación de aplicaciones confiables; el estándar es de uso gratuito.

Capítulo 2: Marco Conceptual

Es necesario obtener el conocimiento del ambiente sobre el cual estas aplicaciones fueron desarrolladas, pues estos son los pasos que realizan los atacantes para dominar sus objetivos. Por lo anterior, se mencionarán los conocimientos necesarios para comprender el ambiente de desarrollo Microsoft.

Framework .NET

Microsoft NET framework es un framework desarrollado por Microsoft para sistemas operativos Windows. Con las últimas actualizaciones ha evolucionado a .NET Core, el cual es compatible con otros sistemas como Mac OS y Linux. El CLR (Common Language Runtime) es el ambiente de software en donde los programas, con uso del framework, son ejecutados.

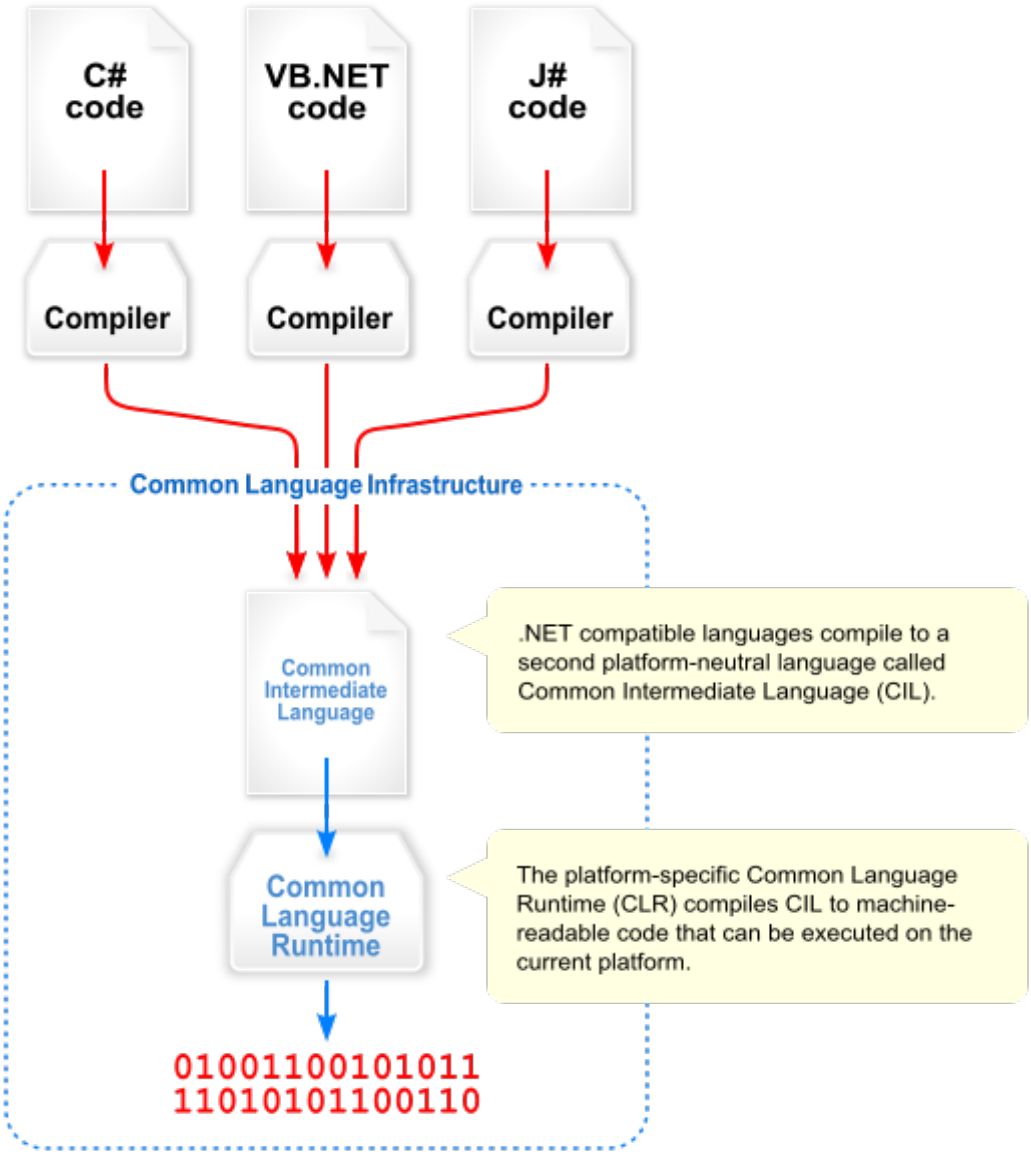

Figura #1: Proceso de traducción de .NET

La arquitectura (figura #1) está formada por el CLI (Common Language Infrastructure) el cual permite utilizar múltiples lenguajes de alto nivel en diferentes plataformas. Transfiere el código compilado a CIL (Common Intermediate Language) en lugar de código máquina y se traduce a CLR (Common Language Runtime) conforme se ejecuta, lo cual lo convierte en código máquina (bits).

Lenguaje C# y VBasic

C# y VBasic son lenguajes de programación orientados a objetos desarrollados por Microsoft dentro del framework de .NET. Sus archivos contienen la extensión .cs y .vb respectivamente.

Windows Presentation Foundation (WPF)

Windows Presentation Foundation es una tecnología de Microsoft que permite el desarrollo de aplicaciones con características de aplicaciones Windows y Web. La interfaz gráfica usa el lenguaje XAML (Extensible Application Markup Language), mientras que C# en la capa lógica.

Internet Information Services (IIS)

Internet Information Services es el servidor web creado por Microsoft para alojar aplicaciones de Web de .NET.

Seguridad en .NET

El framework de .NET cuenta con varias librerías incluidas, que proveen desde criptografía hasta acceso por roles y configuración de archivos.

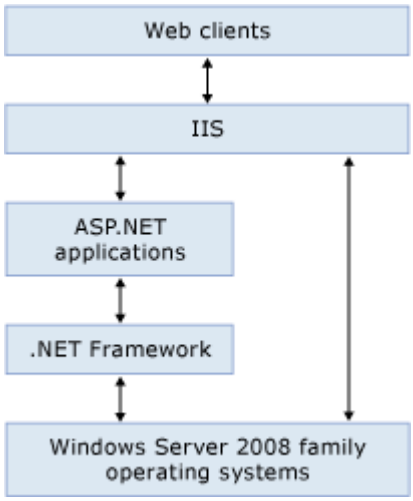

Figura #2: Diagrama de Interacción de IIS

Como se indica en la figura #2 el IIS (Internet Information Services) es el medio por el cual los clientes web se comunican con la aplicación de .NET. El IIS verifica la autorización del cliente para acceder a la aplicación, según su solicitud.

Nuget

Nuget es el encargado de manejar las versiones de paquetes (librerías) instaladas en los proyectos de .NET.

HTML

Hypertext markup Language es el estándar para la creación de páginas y aplicaciones web.

SQL

Es un lenguaje utilizado para el manejo de base de datos relacionales.

Estructura de Proyecto

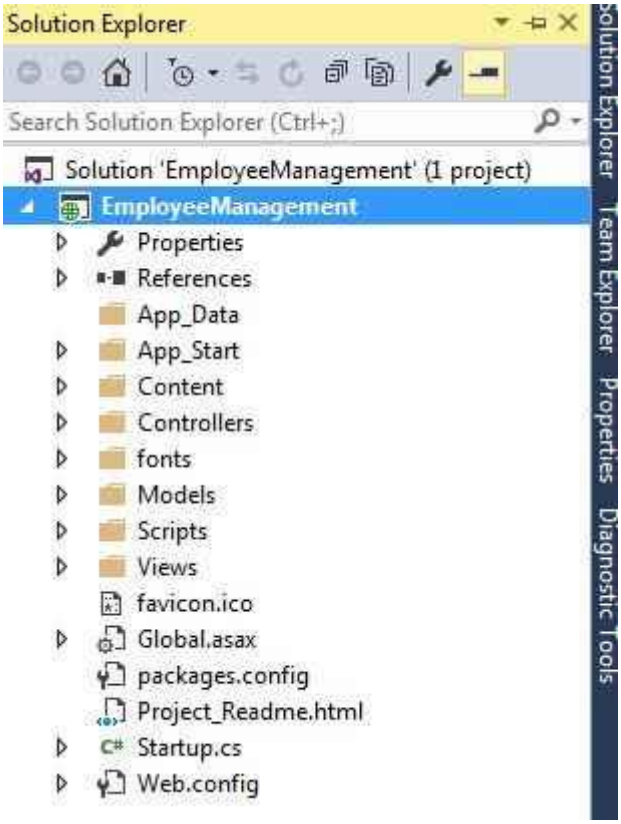

Figura #3: Estructura de Proyecto Web

En la estructura de proyecto web contiene varias carpetas, esto debido a que se basa en el patrón de diseño Model View Controller:

App_Data: Almacena datos como archivos de base de datos y archivos XML que usa la aplicación.

Content: Almacena recursos externos como imágenes.

Controllers: Es el intermediario entre la vista y el modelo, además de ser el punto de contacto con la base de datos, utiliza archivos con extensión .cs.

Models: Contienen los archivos que definen los datos usados en la aplicación; utiliza archivos con extensión .cs.

Views: Al indicar su nombre son los archivos que contienen la parte gráfica de la aplicación; utiliza archivos con extensión. cshtml.

Web.config: Es el archivo XML que define la autenticación y la autorización en la aplicación.

Global.asax: El archivo Global.asax contiene código para responder a eventos del nivel de la aplicación y de la sesión, provocados por ASP.NET o por módulos HTTP.

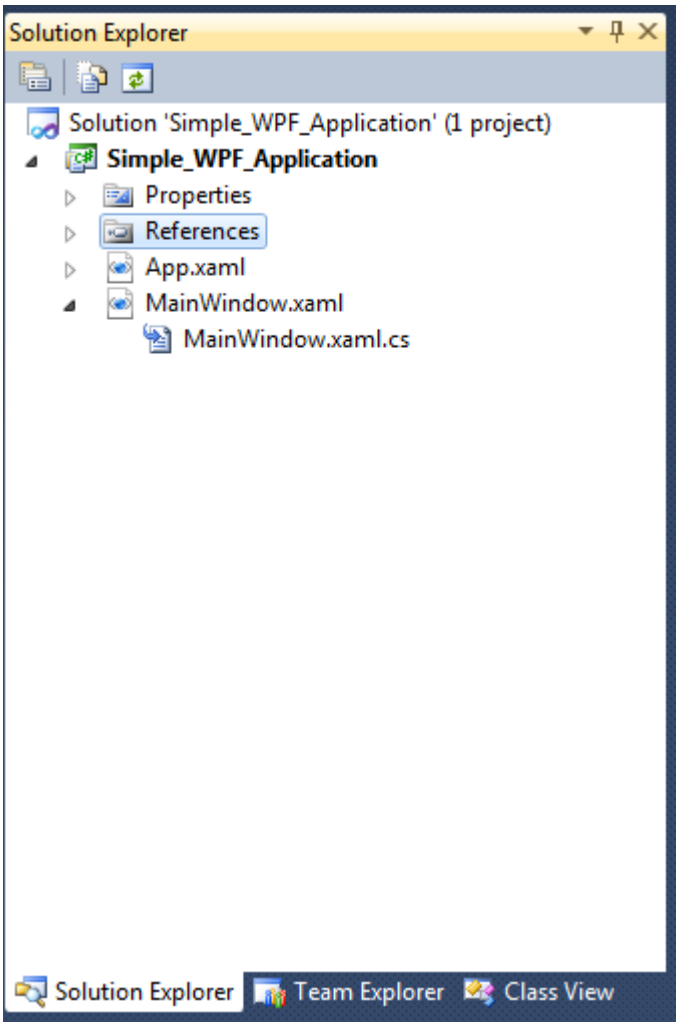

Figura #4: Estructura de Proyecto Escritorio (WPF)

Su estructura inicial está formada por un archivo de .xaml para la interfaz gráfica, sin embargo, esto no quiere decir que no puede ser segmentada con el fin de facilitar claridad. Es importante notar que dentro de los .xaml están los archivos.xaml.cs y esta es la que maneja la lógica de cada ventana.

Análisis Estático y Dinámico

El análisis estático es el proceso donde se evalúa el código del software sin tener que ejecutarlo. Por el contrario, el análisis dinámico evalúa el software mientras se ejecuta. El análisis estático puede realizarse con el código fuente o ingeniería reversa del archivo compilado (.dll en el caso de los archivos producidos por .NET) a un archivo legible; este proceso puede ser realizado por una herramienta como ILSpy.

Estándar de verificación de seguridad de aplicaciones de OWASP

Debido a que las aplicaciones están relacionadas con una empresa emisora de tarjetas de crédito y débito, pero no únicamente para la emisión de tarjetas y lo que se necesita es asegurar los productos desarrollados por la empresa ABC, se decide utilizar como base para la creación del método aplicado, el estándar de verificación de seguridad de aplicaciones de OWASP versión 3.0.1. Este proyecto provee una base de los controles de seguridad para probar aplicaciones y una lista de requerimientos para desarrollo seguro.

En el estándar se definen 3 niveles de seguridad: oportunista, estándar y avanzado. Sobre el nivel 1, el de oportunista, se señala: “si se defiende adecuadamente contra vulnerabilidades de seguridad de aplicaciones que son fáciles de descubrir y se incluyen en el top 10 u otras similares”. Por ello, esta propuesta se centra en validar que las aplicaciones cumplan con el nivel 1, la cual comprende 77 controles.

El estándar se divide en 15 dominios para los cuales deben ser verificados los controles, según el nivel:

- V1. Arquitectura, diseño y modelado de amenazas
- V2. Autenticación
- V3. Gestión de sesiones
- V4. Control de acceso
- V5. Manejo de entrada de datos maliciosos
- V7. Criptografía en el almacenamiento
- V8. Gestión y registro de errores
- V9. Protección de datos
- V10. Comunicaciones
- V11. Configuración de seguridad HTTP
- V15. Lógica de negocio
- V16. Archivos y recursos
- V19. Configuración

Clasificación de Controles

Los controles contenidos en el Estándar de Verificación de Seguridad en aplicaciones serán clasificados en dos categorías: automatizable y no automatizable. Se definen como automatizables los controles que son comprobados sin intervención humana, por medio de un algoritmo implementado en software. No automatizables: requiere intervención humana ya sea de forma parcial o total para su verificación.

Controles Automatizables

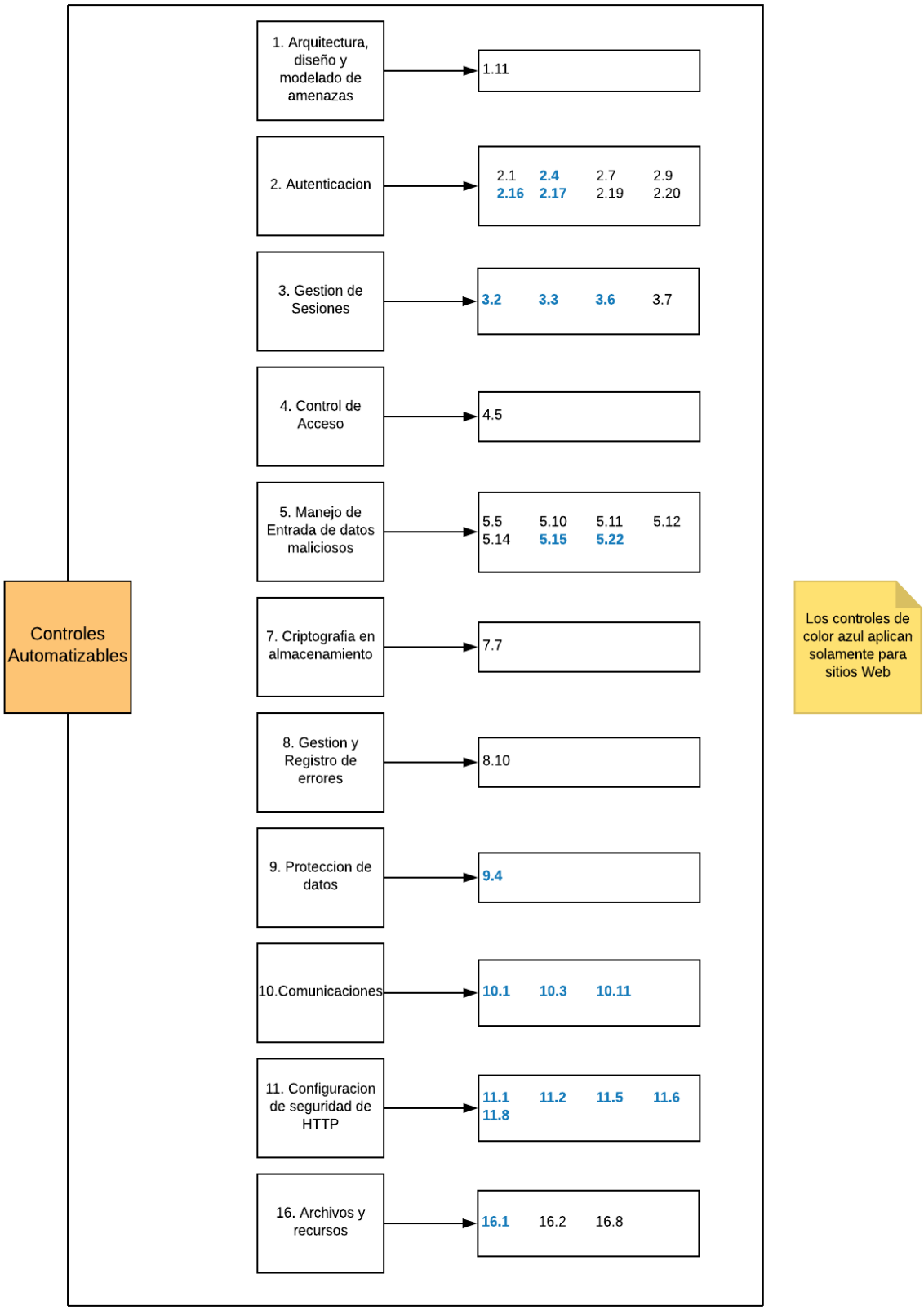

Figura #5: Controles automatizables

Controles No Automatizables

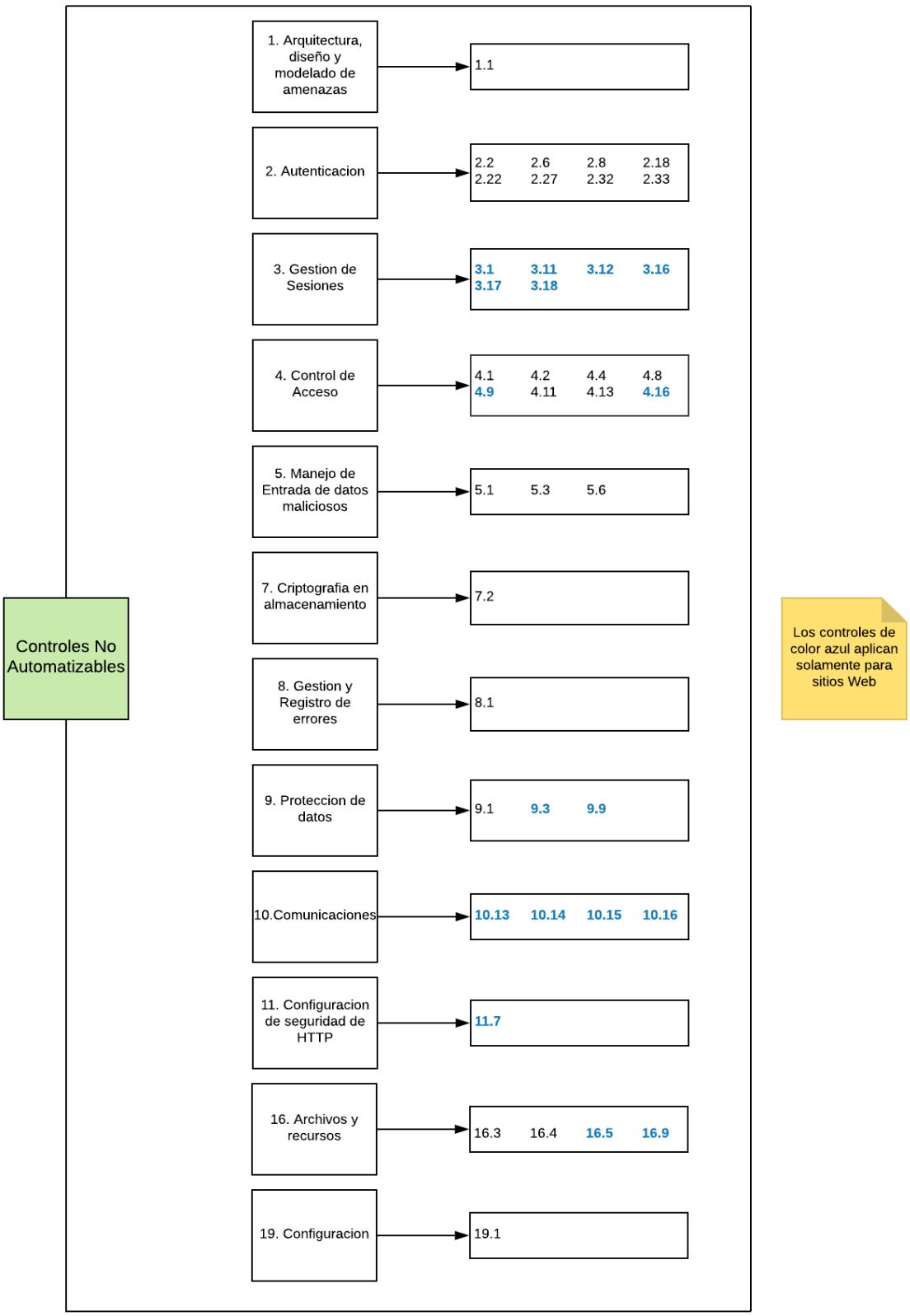

Figura #6: Controles No Automatizables

Guía de implementación de Controles Automatizables

V1. Arquitectura, diseño y modelado de amenazas

1.11 Verificar que todos los componentes de la aplicación, bibliotecas, módulos, frameworks, plataformas y sistemas operativos se encuentran libres de vulnerabilidades conocidas.

Instalando desde Nuget, el paquete de SafeNuget verifica si los paquetes instalados en el proyecto cuentan con alguna vulnerabilidad conocida.

V2. Autenticación

2.1 Verificar que todas las páginas y recursos requieren autenticación, excepto aquellos específicamente destinados a ser públicos (Principio de mediación completa).

Intenta acceder a cualquier dirección del sitio web y verificar si se realiza una validación para saber si el usuario está autenticado. En el caso de una aplicación de escritorio, se requiere validar que se le solicite la identidad al usuario al ejecutarse y según su perfil pueda acceder a diferentes funcionalidades.

2.4 Verificar que todos los controles de autenticación se realicen del lado del servidor.

En las llamadas del cliente al servidor se usan anotaciones que verifican si los usuarios están autorizados para realizar las acciones. Estas autorizaciones son determinadas por roles.

2.7 Verificar que los campos de contraseñas permiten o fomentan el uso de frases como contraseñas (passphrases) y no impiden el uso de gestores de contraseñas, contraseñas largas o altamente complejas.

Mediante el uso de expresiones regulares se puede validar que las contraseñas cumplan con cantidad de caracteres y uso de caracteres especiales.

2.9 Verificar que la funcionalidad de cambio de contraseña solicite la contraseña anterior, la nueva contraseña y una confirmación de la contraseña.

Se debe validar que la funcionalidad de cambio de contraseña compare la contraseña actual antes de realizar el cambio de lado del servidor.

2.16 Verificar que las credenciales son transportadas mediante un enlace cifrado adecuadamente y que todas las páginas/funciones que requieren la introducción de credenciales del usuario, se realicen utilizando enlaces cifrados.

Para comprobar que las credenciales son transportadas mediante un enlace cifrado, se requiere utilizar tecnologías de https por medio de certificados ssl, que puede notarse en la URL.

2.17 Verificar que las funciones de recuperar contraseña y acceso no revelen la contraseña actual y que la nueva contraseña no se envía en texto plano al usuario.

Requiere verificar que los correos donde se envían las contraseñas sean encriptados.

2.19 Verificar que no se utilizan contraseñas por defecto en la aplicación o cualquiera de los componentes utilizados por esta (como "admin/password").

Evaluar el acceso mediante ataque de diccionario para descartar las contraseñas por defecto.

2.20 Verificar que existen mecanismos de antiautomatización que previenen la verificación de credenciales obtenidas de forma masiva, ataques de fuerza bruta y ataques de bloqueos de cuentas.

Se debe contar con medidas de bloqueo de cuenta en caso de múltiples intentos, con el fin de evitar ataques de fuerza bruta. Luego de una cantidad de intentos se debe realizar el bloqueo de la cuenta por un tiempo determinado.

V3. Gestión de Sesiones

3.2 Verificar que las sesiones se invalidan cuando el usuario cierra la sesión.

Se debe tratar de ingresar nuevamente a alguna de las páginas del sitio antes visitadas y esperar que al ingresar solicite nuevamente inicio de sesión debido a que la sesión anterior se encuentra cerrada.

3.3 Verificar que las sesiones se invalidan luego de un período determinado de inactividad.

Es necesario indicar en el archivo de configuración `<sessionState timeout="60" />` el tiempo de expiración para las sesiones.

3.7 Verificar que toda autenticación exitosa y reautenticaciones generen un nuevo identificador de sesión.

Por defecto .NET cambia el identificador de sesión, cada vez que se genera una nueva autenticación. Se debe reutilizar cada uno de los identificadores los cuales deberían ser inválidos.

V4. Control de acceso

4.5 Verificar que la navegación del directorio esté deshabilitada a menos que esto sea deliberadamente deseado. Además, las aplicaciones no deben permitir el descubrimiento o divulgación de metadatos de archivos o directorios, como carpetas que contengan Thumbs.db, DS_Store, o directorios .git o SVN.

Se debe limitar el acceso a directorios que podrían comprometer información, esto se realiza desde el archivo de configuración escondiendo segmentos. Por ejemplo, esta implementación limita el acceso al folder de Uploads, de tal manera que los usuarios no pueden acceder al directorio mediante la URL, ya que uploads contiene archivos proveídos por otros usuarios.

```
<security> <requestFiltering> <hiddenSegments> <add segment="Uploads"/> </hiddenSegments> </requestFiltering> </security>
```

V5. Manejo de entrada de datos maliciosos

5.5 Verificar que se aplican las rutinas de validación de entradas de datos del lado del servidor.

En caso de que la llamada se realice desde alguna herramienta de prueba como postman, la información enviada no sería validada por el UI, por lo que es necesario validar del lado del servidor. .NET provee mecanismos para esta validación como data annotations y fluent validation que realizan la validación por cliente y por servidor.

5.10 Verificar que todas las consultas de SQL, HQL, OSQL, NOSQL, procedimientos almacenados y llamadas de procedimientos almacenados están protegidos por la utilización de declaraciones preparadas o parametrización de consultas, y por lo tanto, no sean susceptibles a la inyección de SQL

Requiere validar que en cada consulta del servidor a la base de datos se utilice los sql parameters de forma que no sean susceptibles a inyección de SQL.

5.11 Verificar que la aplicación no es susceptible a la inyección LDAP, o que los controles de seguridad previenen inyección LDAP.

Requiere realizar una codificación de las llamadas para proteger contra caracteres especiales que pueden realizar una consulta en el directorio de usuarios (Active Directory). La librería AntiXSS (Cross-site scripting) provee métodos para realizar estas acciones.

5.12 Verificar que la aplicación no es susceptible a la inyección de comandos del sistema operativo, o que los controles de seguridad previenen la inyección de comandos del sistema operativo.

Requiere realizar una codificación de los llamados para proteger contra caracteres especiales que pueden realizar una consulta en el sistema operativo. Nunca se deben utilizar como un parámetro del usuario para realizar un comando en el sistema operativo.

5.13 Verificar que la aplicación no es susceptible a la inclusión de archivo remoto (RFI) o inclusión de archivo Local (LFI) cuando el contenido es utilizado como una ruta a un archivo.

Verificar que los scripts externos se encuentren en un content delivery network de forma que no puedan ser manipulados por un tercero con facilidad. Mantener la opción de cross origin resource sharing deshabilitada de no ser necesaria.

5.14 Verificar que la aplicación no es susceptible a ataques comunes de XML, como manipulación de consultas XPath, ataques de entidad externa XML y ataques de inyección XML.

Para prevenir ataques de XPath es necesario validar y compilar la expresión que se vaya a utilizar, para asegurarse de que no ha sido modificada previamente.

5.15 Asegurar que todas las variables string utilizadas dentro de HTML u otro lenguaje web interpretado en cliente se encuentra apropiadamente codificado manualmente o que se utilizan plantillas que automáticamente codifican contextualmente para asegurar que la aplicación no sea susceptible a ataques DOM Cross-Site Scripting (XSS).

Se requiere codificar toda la información enviada hacia el servidor en la URL o en un formulario, de forma que no se ejecute como texto y no como script. También prevenir caracteres innecesarios ayuda a mitigar este tipo de ataque.

5.22 Verificar que HTML no confiables provenientes de editores WYSIWYG o similares sean debidamente sanitizados con un sanitizador de HTML y se manejen apropiadamente según la validación de entrada y codificación.

Existen herramientas para sanitizar el HTML autogenerado por herramientas WYSIWYG. En caso de no hacerlo, se deben usar librerías como AntiXSS que permiten sanitizar el HTML en tiempo de ejecución con la llamada. GetSafeHtmlFragments.

V7. Criptografía en el almacenamiento

7.7 Verificar que los algoritmos criptográficos utilizados por la aplicación hayan sido validados contra FIPS 140-2 o un estándar equivalente.

Desde la versión de 3.0 de .NET framework en adelante los algoritmos de encripción han sido validados contra NIST que maneja un estándar equivalente. Pero en caso de querer usar FIPS 140-2 una de las clases que lo provee es AESCryptoServiceProvider.

V8. Gestión y registro de errores

8.10 Verificar que un registro de auditoría o similar permita la no repudiación de transacciones claves.

Verificar que haya un registro de auditoría donde se guarde información del usuario que realiza las transacciones claves e inclusive la hora. Además, es necesario que no se posea permisos para actualizar estos registros, a menos que sea realizado por un administrador de base de datos.

V9. Protección de datos

9.4 Verificar que la aplicación establece encabezados anticaché adecuados según el riesgo de la aplicación, tales como las siguientes: Expires: Tue, 03 Jul 2001 06:00:00 GMTT Last-Modified: {now} GMT Cache-Control: no-store, no-cache, mustrevalidate, max-age=0 Cache-Control: post-check = 0, pre-check = 0 Pragma: no-cache.

Las operaciones que no deben ser cacheadas deben contener la siguiente instrucción en el servidor:

```
HttpContext.Current.Response.Headers.Set ("Cache-Control", "private, max-age=0");
```

. El max-age define el periodo de retención este caso sería 0.

V10. Comunicaciones

10.1 Verificar que puede construirse la cadena de confianza desde una CA (Autoridad de Certificación) para cada certificado TLS (Transport Layer Security) del servidor, y que cada certificado del servidor sea válido.

Requiere la implementación del objeto X509CertificateValidator que revisa la cadena de confianza del certificado contra las políticas que define cómo se valida el certificado.

10.3 Verificar que se utiliza TLS para todas las conexiones (incluyendo conexiones back-end y externas) autenticadas o que involucran funciones o información sensible, y no recaigan en protocolos inseguros o sin cifrado. Debe asegurarse de que la alternativa más fuerte es el algoritmo preferido.

Requiere que se defina en el archivo global asax el siguiente filtro: `GlobalFilters.Filters.Add(new RequireHttpsAttribute());` de forma que en todo el sitio cada petición requiera https.

10.11 Verificar que los encabezados HTTP Strict Transport Security sean incluidos en todas las peticiones y para todos los subdominios, como `Strict-Transport-Security: max-age = 15724800; includeSubdomains.`

Para realizar esta validación basta con agregarlo en el web.config, de esta forma será agregado en todas las peticiones `system.webServer>`

```
<httpProtocol> <customHeaders> <add name="Strict-Transport-Security"
value="max-age=15724800; includeSubDomains"/> </customHeaders>
</httpProtocol> </system.webServer>
```

V11. Configuración de seguridad HTTP

11.1 Verificar que la aplicación acepte solo un conjunto definido de métodos de solicitud HTTP y que son necesarios, como GET y POST, y métodos no utilizados (por ejemplo: TRACE, PUT y DELETE) se encuentran explícitamente bloqueados.

Para definir los métodos aceptados de solicitud HTTP se requiere la configuración del web.config donde se deben identificar las opciones limitadas

```
<configuration> <system.webServer> <security> <requestFiltering>
<verbs allowunListed="true"> <add verb="DELETE" allowed="false" />
<add verb="PUT" allowed="false" /><add verb="TRACE" allowed="false"
/></verbs> </requestFiltering> </security> </system.webServer>
</configuration>
```

11.2 Verificar que cada respuesta HTTP contenga una cabecera content-type en la que se especifique un conjunto utilizando un conjunto de caracteres seguros (Ejemplo: UTF-8, ISO 8859-1).

Para definir la cabecera se puede realizar en cada página respuesta del servidor utilizando la instrucción Response.ContentType y asignándole el content type acorde con la acción realizada.

11.5 Verificar que los encabezados HTTP o cualquier parte de la respuesta HTTP no expongan información detallada de la versión de los componentes del sistema.

Verificar en el web.config la opción de `<httpRuntime enableVersionHeader="false" />`

11.6 Verificar que todas las respuestas del API contienen opciones X-Content-Type: nosniff y ContentDisposition: attachment; filename="api.json" (u otro nombre de archivo apropiado para el tipo de contenido).

Para agregar el X-Content-Type: nosniff es necesario agregar un custom header en el web.config `<configuration> <system.webServer> <httpProtocol> <customHeaders> <add name="X-Content-Type-Options" value="nosniff" /> </customHeaders> </httpProtocol> </system.webServer> </configuration>`

En el caso de ContentDisposition es necesario definirlo en el `Response.AddHeader()` donde se define el nombre del archivo.

11.8 Verificar que el encabezado "X-XSS-Protection: 1; mode=block" esté presente para habilitar a los navegadores a filtrar XSS reflejados.

Este encabezado puede ser agregado como un custom header desde el archivo de web.config:

```
<httpprotocol>
    <customheaders>
        <remove name="X-Powered-By">
           <add name="X-XSS-Protection" value="1; mode=block"></add>
        </remove>
    </customheaders>
</httpprotocol>
```

V16. Archivos y recursos

16.1 Verificar que las URL de redireccionen y reenvíen solo a destinos clasificados en la lista blanca, o mostrar una advertencia cuando se redirija a contenido potencialmente no confiable.

Para prevenir un ataque por medio de redireccionamiento de URL se necesita hacer una validación donde se reciba una URL como parámetro. Donde se compara con una lista de URLs aceptadas.

16.2 Verificar que archivos no confiables enviados a la aplicación no sean utilizados directamente por comandos de I/O (Entrada/Salida) de archivos, especialmente para proteger contra manipulaciones de rutas, archivo local incluido, manipulación de tipo mime y vulnerabilidades de inyección de comandos de sistema operativo.

Es necesario filtrar el archivo según la necesidad y no dejar que se envíen a la aplicación formatos no permitidos; por medio del Content-Type se identifica el tipo de archivo. En caso de ser necesario, ejecutar alguna instrucción contenida en el archivo. Se deben definir las instrucciones permitidas, de lo contrario, se podría hacer uso de alguna instrucción no autorizada si el archivo es manipulado. Se deben restringir los permisos en la mayor medida posible, y es necesario que todos los archivos sean revisados por un antivirus antes de ser abiertos.

16.8 Verificar que el código de la aplicación no ejecuta datos cargados obtenidos de fuentes no confiables.

Es necesario validar la fuente de los datos antes de ejecutarlos, además de validar los caracteres y codificarlos para procesarlos como texto, en lugar de instrucciones que podrían comprometer la aplicación.

Capítulo 3: Marco Metodológico

Tipo de Investigación

Fidias G Arias (2012), señala: La investigación experimental es un proceso que consiste en someter a un objeto a determinadas condiciones, estímulos o tratamiento, para observar los efectos y reacciones. Este proyecto al someter las aplicaciones ante una revisión, permite sacar conclusiones y posibles perfeccionamientos, con el fin de obtener mejores resultados en cada iteración. Para atacar este problema se utilizará la investigación experimental.

Enfoque

El enfoque es cuantitativo pues según se apliquen los controles, se podrá saber cuánto porcentaje de seguridad tiene la aplicación con respecto al estándar.

Alcance

El alcance en esta investigación corresponde al descriptivo, pues busca detectar las fallas encontradas en las aplicaciones y cómo estas pueden ser remediadas.

Instrumentos

Fidias G Arias (2012) indica que en la observación participante el investigador forma parte de la comunidad o medio donde se realiza el estudio. En esta investigación la observación es participante, todas las buenas prácticas definidas deben ser verificadas, por ejemplo, en una lista de chequeo para las no automatizables. A su vez, el investigador se encarga de la creación el prototipo; tener influencia en el proyecto lo hace un participante.

Técnicas de Análisis

La información obtenida de las fuentes será utilizada para crear tanto la lista automatizada como la no automatizada para cada una de las aplicaciones por evaluar; los resultados serán entregados con el objetivo de concientizar sobre el grado de seguridad que se posee y cuáles son los siguientes pasos por mejorar. Se hará un contraste entre los resultados para verificar si existe una tendencia en el diseño de aplicaciones o si son casos aislados.

Solución Propuesta

La investigación tiene como objetivo la creación de un procedimiento que satisfaga el problema del proyecto, por lo cual se detalla la implementación para validar cada uno de los controles, tanto los automatizables por medio de un prototipo, como los no automatizables a través de una lista de chequeo. Los resultados de la lista de chequeo pueden ser añadidos al prototipo y así contrastar ambas revisiones, para obtener el porcentaje de cumplimiento por cada requisito.

El prototipo realizará un análisis estático de las aplicaciones, pues se cuenta con el código fuente; dicha información será analizada y se mostrarán los resultados según cada uno de los requisitos y controles relacionados. El método de validación es la interpretación abstracta, Patrick Cousot (2000); por medio de semántica, según las reglas definidas para cada control, se indicará si los controles automatizados se encuentran cubiertos.

A continuación, se enlistan ejemplos de archivos que serán revisados por el prototipo, de acuerdo con lo indicado en la guía automatizada, según cada control:

Archivo
web.config
global.asax
archivos html
archivos js
archivos de c#/vb.net
packages.json
archivos .xaml/.cs

Tabla #1: Archivos que serán evaluados

Finalmente, se mostrará una tabla con el porcentaje de cobertura por requisito, según los controles evaluados y se realizará un promedio de los porcentajes para conocer la cobertura del estándar en general:

Capítulo	Cobertura
V1. Arquitectura, diseño y modelado de amenazas	(0%-100%)
V2. Autenticación	(0%-100%)
V3. Gestión de sesiones	(0%-100%)
V4. Control de acceso	(0%-100%)
V5. Manejo de entrada de datos maliciosos	(0%-100%)
V7. Criptografía en el almacenamiento	(0%-100%)
V8. Gestión y registro de errores	(0%-100%)
V9. Protección de datos	(0%-100%)
V10. Comunicaciones	(0%-100%)
V11. Configuración de seguridad HTTP	(0%-100%)
V16. Archivos y recursos	(0%-100%)
V19. Configuración	(0%-100%)
Total:	Promedio de los porcentajes

Tabla #2: Tabla de Resultados

En el prototipo indicado (Figura #5) se puede ver que existen dos requisitos para el evaluador: el archivo csv con los resultados obtenidos de los controles no automatizables en Excel y el directorio, donde se encuentra el proyecto para los controles automatizables.

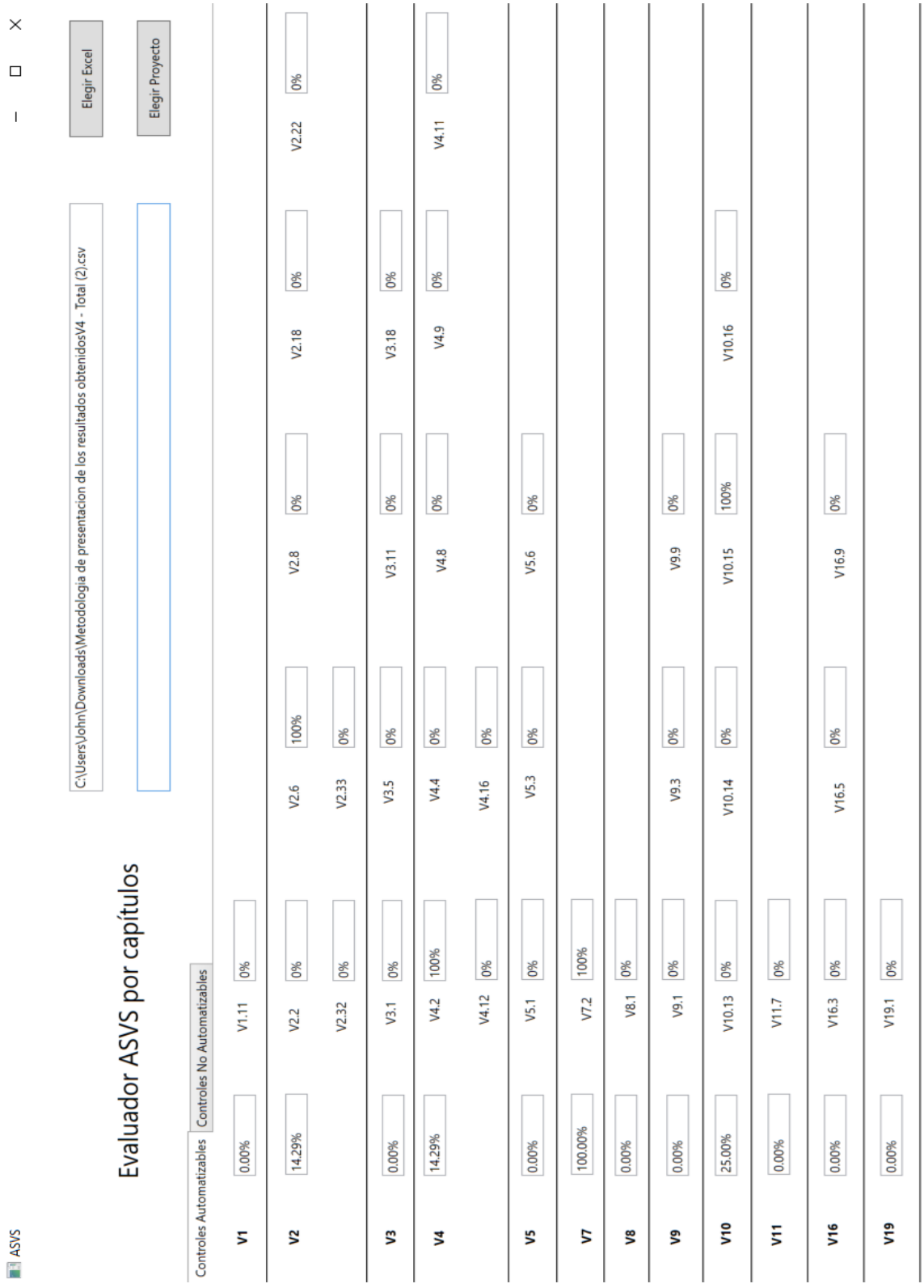

Figura #7: Prototipo de Aplicación Controles Automatizables

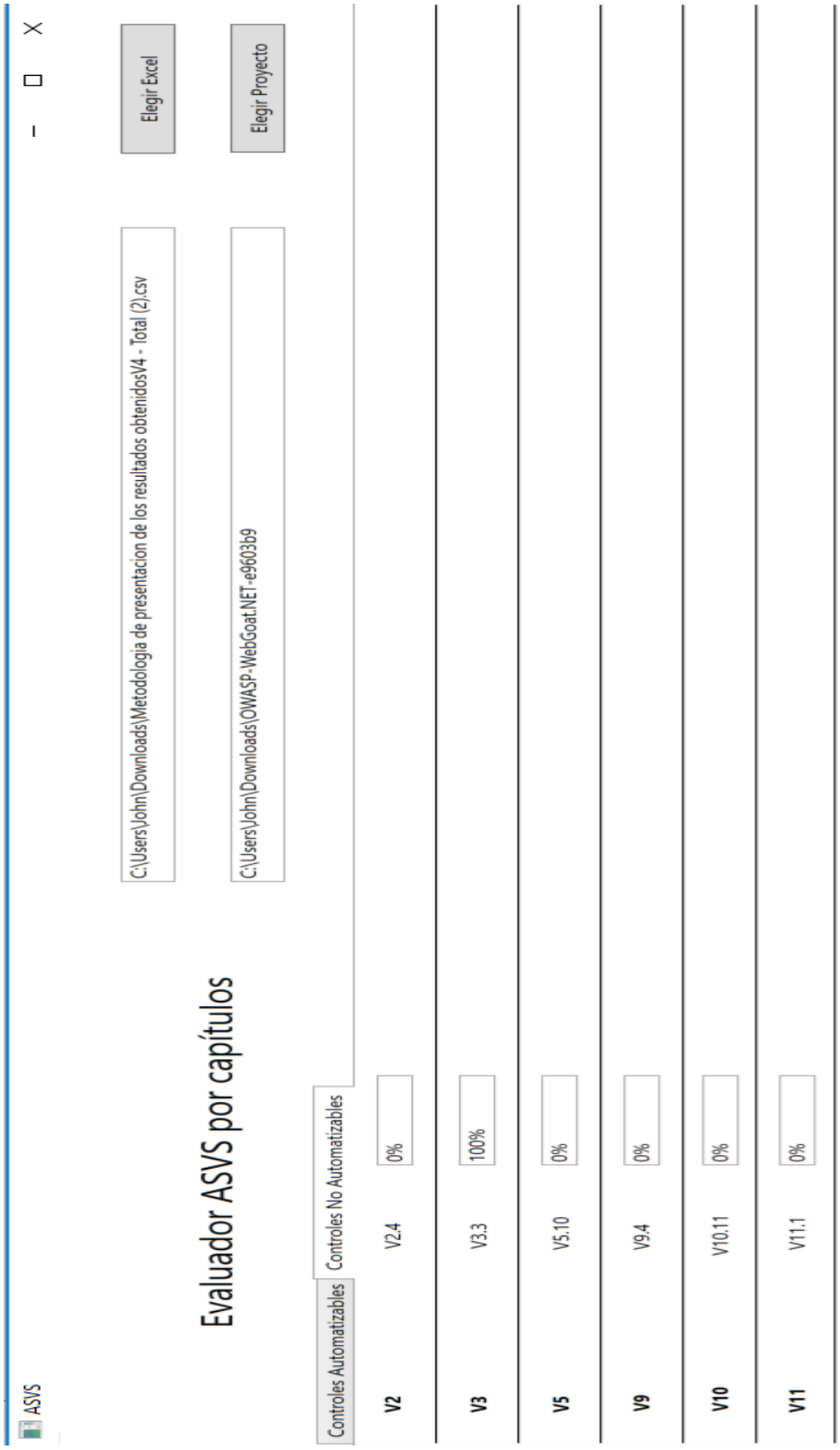

Figura #8: Prototipo de Aplicación Controles No Automatizables

Evaluación de Controles Automatizables

Control 2.4 (Web y Escritorio)

Archivos involucrados	**Descripción**
Controllers cs *archivos cs*	En el caso de una aplicación web debe revisarse cada uno de los controllers, estos deben tener definidos data-annotations e indicar el keyword “Authorize” que señala si quien realiza la acción debe poseer una sesión abierta. También se pueden determinar por roles. Se realiza una revisión por cada encabezado definido en los controles y se obtiene un promedio al calcular la cantidad de llamadas con parámetros/cantidad de llamadas a la base de datos *100. Una aplicación de escritorio puede implementar una autenticación customizada, esto se puede realizar implementando clases de IIPrincipal e IIdentity librerías de .NET. IIdentity para identificar al usuario mientras que IIPrincipal

	que define el contexto de seguridad. Por tanto, en la revisión deben ser tomadas ambas maneras.

Control 3.3 (Web)

Archivos involucrados	Descripción
web.config	Por defecto, el tiempo de expiración es de 20 minutos; este valor puede ser definido, se debe verificar y que posea un tiempo determinado. Por ejemplo `<sessionState timeout="60" />` define un tiempo de 60 minutos. Se verifica el tiempo de expiración definido en el archivo, en caso de encontrarlo exitosamente, se indica con un 100%, de lo contrario 0%.

Control 5.10 (Web y Escritorio)

Archivos involucrados	Descripción
archivos cs *Controllers cs*	Cada una de las llamadas a ejecución de sql (executesqlcommand) debe ser revisada para verificar que se usen parámetros en lugar de sentencias de sql en texto anidadas, las cuales

	serían susceptibles ataques de inyección de sql. Se realiza una revisión por cada llamada realizada a la base de datos y se obtiene un promedio al calcular cantidad de llamadas con parámetros/cantidad de llamadas a la base de datos *100.

Control 9.4 (Web)

Archivos involucrados	**Descripción**
global.asax	El encabezado anti caché puede ser definido en el global.asax y señalar que la max-age debe ser de 0, lo cual indica que la información no debe ser retenida en caché. Se verifica que el encabezado anti caché se encuentre definido como 0 en el archivo, en caso de encontrarlo exitosamente se indica con un 100%, de lo contrario 0%.

Control 10.11 (Web)

Archivos involucrados	Descripción
global.asax *web.config*	El strict-transport-security include subdomains es necesario para redirigir todo el tráfico por medio de https. Se verifica que strict-transport-security se encuentre definido en el archivo, en caso de encontrarlo exitosamente, se indica con un 100%, de lo contrario 0%.

Control 11.1 (Web)

Archivos involucrados	Descripción
web.config	Se deben definir los métodos aceptados de solicitud HTTP `<add verb="DELETE" allowed="false" />` que no permite el DELETE. `<add verb="PUT" allowed="false" />` que no permite el PUT. `<add verb="TRACE" allowed="false" />` que no permite el TRACE. `<add verb="GET" allowed="true" />` que

	permite el GET. `<add verb="POST" allowed="true" />` que permite el POST. Se verifica que los métodos se encuentren definidos para no permitir llamadas PUT, TRACE, DELETE, pero sí GET, POST; en caso de encontrarlos exitosamente se indica con el porcentaje, utilizando la cantidad de métodos hallados entre el total de métodos * 100.

Capítulo 4: Resultados Obtenidos

Utilizando el procedimiento de buenas prácticas se realizó la revisión en 2 aplicaciones relacionada con emisión de tarjetas de crédito/débito.

Aplicación 1(Escritorio)

Controles no Automatizables

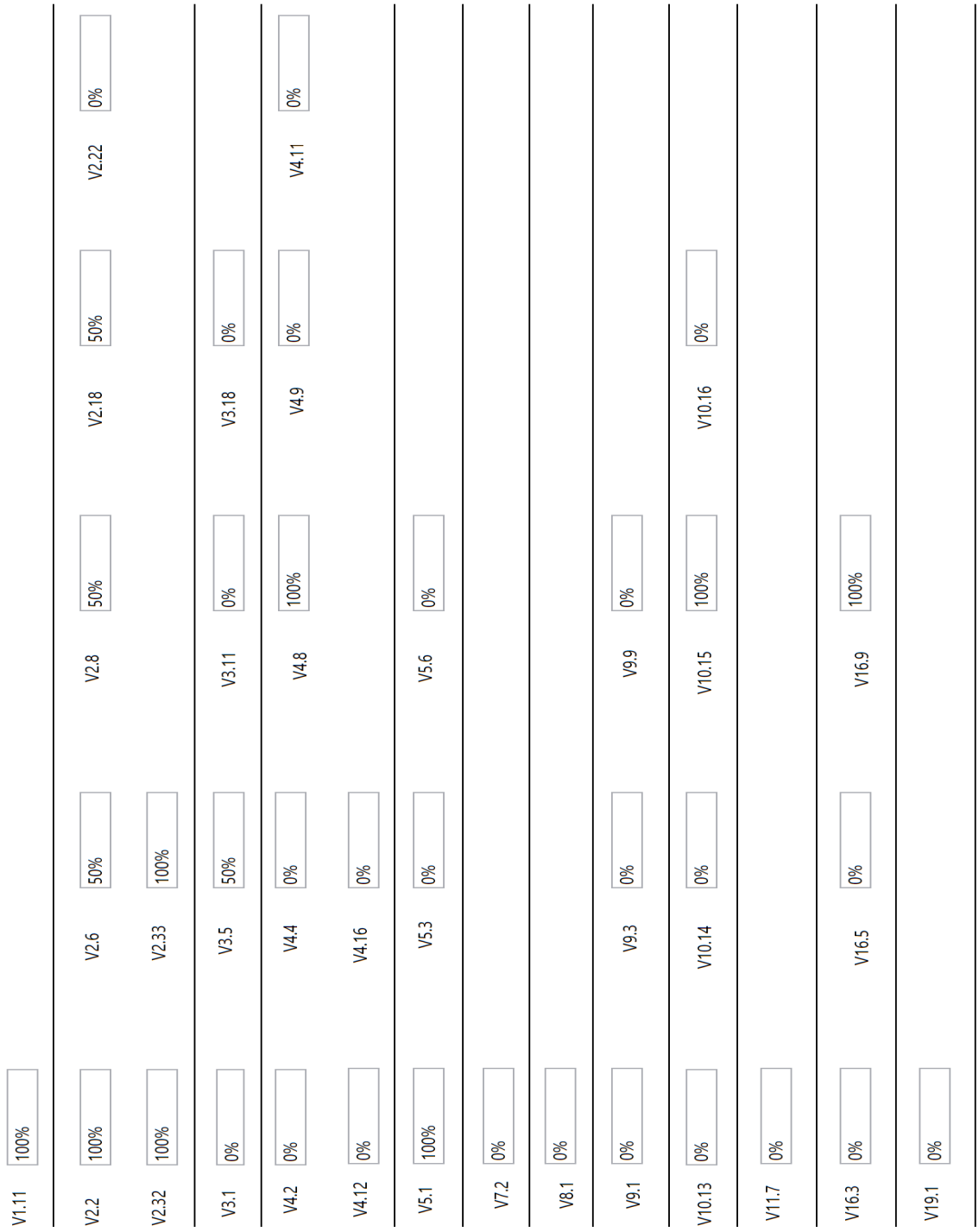

Figura #9: Resultados No Automatizables Aplicación 1

La aplicación se enfoca en hacer los flujos para la creación de las tarjetas. Obtuvo una cobertura superior en capítulos 1, 2, 3, 4, 5, 10 y obtuvo la mayor cobertura en los capítulos 1 y 2 relacionados con arquitectura, diseño y modelo de amenazas, como el manejo de sesiones.

Autenticación. El capítulo 2 cuenta con los controles:

- Bitácora de intentos
- Protección de la contraseña
- Protección ante errores evitando revelar información adicional

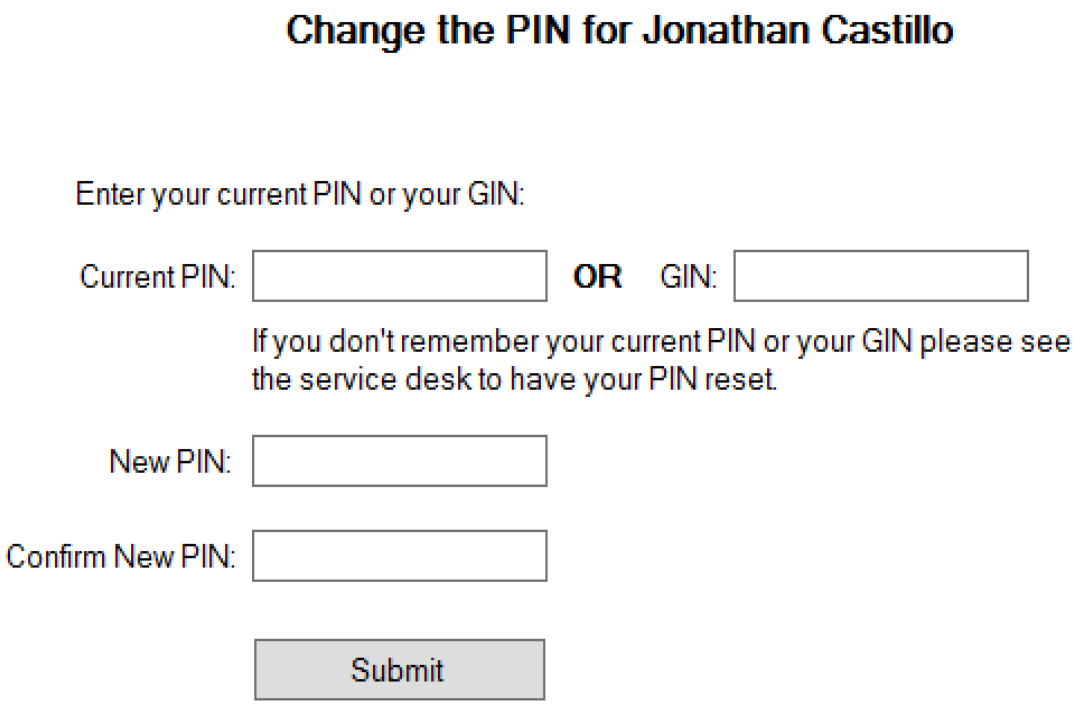

Figura #10: Cambio de PIN Aplicación 1 (2.8)

En el capítulo 3 se cuenta con el cierre de sesión en la aplicación desde cualquier ubicación. Sin embargo, no existe un control de sesiones a las que el usuario pueda tener acceso, por lo que si la aplicación reporta accesos desde diferentes dispositivos no se tomará en cuenta.

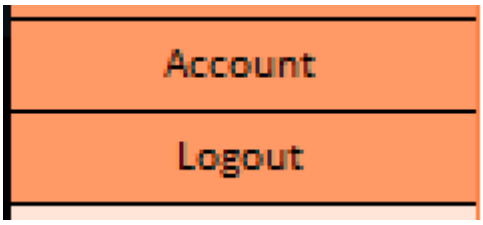

Figura #11: Cierre de sesión Aplicación 1 (3.1)

En el capítulo 4, relacionado con el control de acceso, se garantiza que los controles fallen de forma segura. Las acciones de control de acceso no son registradas. Respecto al manejo de entrada de datos maliciosos, en el capítulo 5 se cuenta con protección contra ataques de buffer.

Criptografía. En el almacenamiento los datos respecto al capítulo 7 no se almacenan encriptados excepto los sensibles; pero también todas las conexiones a bases de datos se encuentran encriptadas, para que, en caso de obtener acceso al archivo de configuración, no necesariamente se tenga acceso a la información.

En el capítulo 10 Comunicaciones, se destaca que los algoritmos utilizados para la encriptación de los datos siguen estándares de la industria en las conexiones a la base de datos.

Controles Automatizables

Controles No Automatizables	Controles Automatizables	Resultados
V2	V2.4	0%
V3	V3.3	0%
V5	V5.10	33%
V9	V9.4	0%
V10	V10.11	0%
V11	V11.1	0%

Figura #12: Resultados Automatizables Aplicación 1

En los controles automatizables no se encontró cobertura en la mayoría, excepto en la protección contra ataques de inyección de SQL. Se destaca que de 12 consultas encontradas 4 fueron utilizando parametrización.

Log

2.4 No se encontraron controles de autenticación del lado del servidor.
3.3 No se encontró definido el tiempo de expiración de la aplicación.
5.10 Se encontraron 4 de 12 consultas con parametrización para no ser susceptibles a la inyección de SQL.
Archivos con parametrización:
Schedule.cs

Archivos sin parametrización:
CheckProductionFiles.cs
Item.cs
PrintHandler.cs

9.4 El encabezado anti caché no se encuentra definido.
10.11 El encabezado HTTP strict-transport-security no se encuentra incluido en las peticiones y para los subdominios.
11.1 No se encontraron los métodos de solicitud HTTP definidos en la aplicación.

La Cobertura de Controles no Automatizables obtuvo **23.35%** mientras que los automatizables **5.5%**.

Aplicación 2 (Web)

Controles no Automatizables

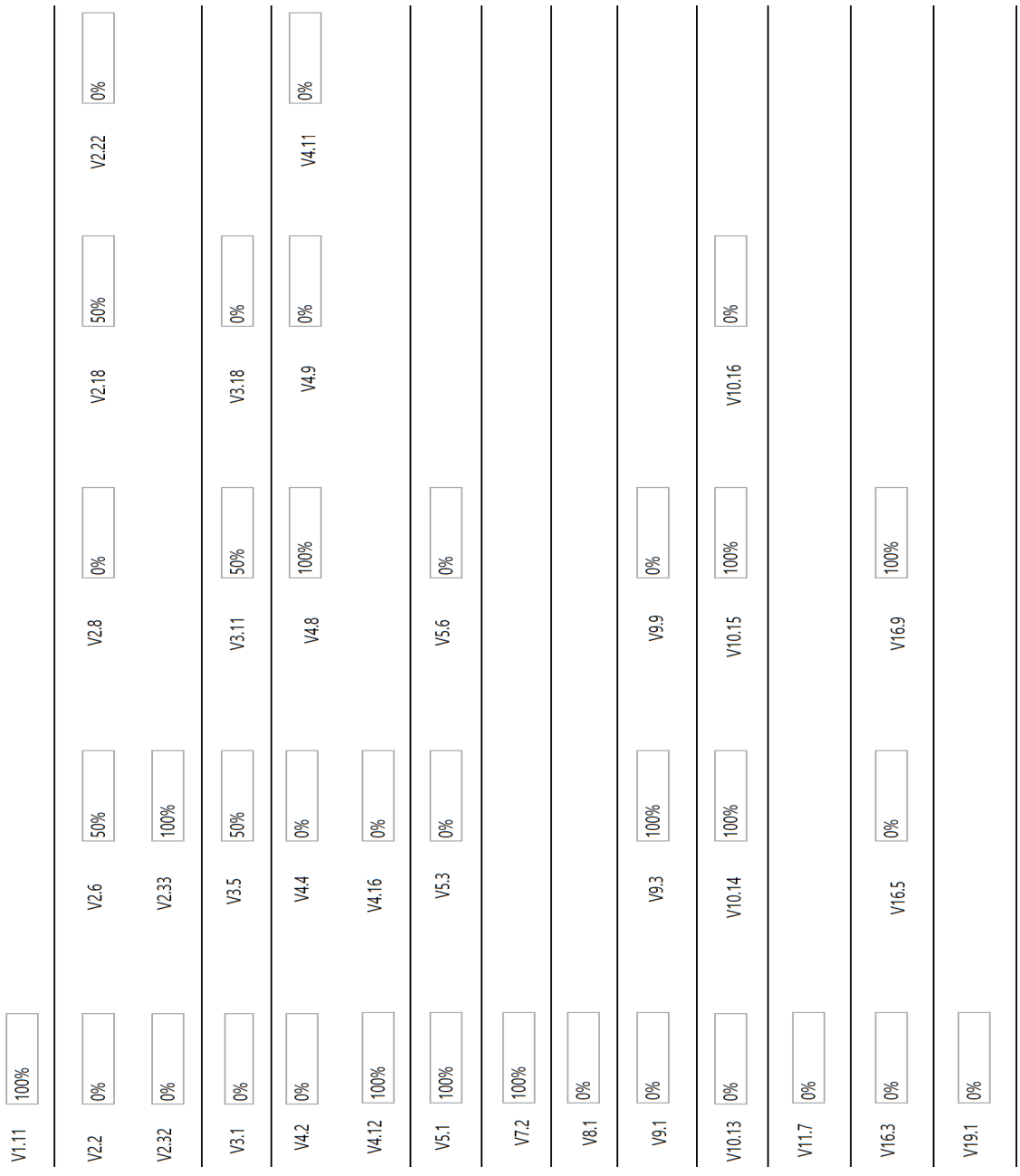

Figura #13: Resultados No Automatizables Aplicación 2

Esta aplicación se encarga del control de errores de otras aplicaciones, así como aplicaciones de políticas, con el objetivo de centralizar las configuraciones. Se obtuvo una cobertura mayor en los capítulos 1,5, 7, 9, 10 y 16.

La aplicación cuenta con protección contra ataques de cross site request forgery para garantizar que la página deba ser visitada antes de realizar una petición por medio de llamadas @Html.AntiForgeryToken() . En el capítulo 5 se puede verificar que en la aplicación se utilizan buenas prácticas ante ataques de buffer. A su vez, muestra errores genéricos cuando la encriptación falla (capítulo 7), contiene errores customizados que previenen información innecesaria para el usuario y cumple el objetivo del capítulo 8.

Además, la aplicación sigue, al igual que la aplicación 1, los estándares, y cumple puntos del capítulo 10. El sitio no requiere Flash, Javascript o Silverlight punto que se pregunta en el capítulo 16.

Controles Automatizables

Controles No Automatizables	Controles Automatizables	Resultados
V2	V2.4	43%
V3	V3.3	0%
V5	V5.10	61%
V9	V9.4	0%
V10	V10.11	0%
V11	V11.1	0%

Figura #14: Resultados Automatizables Aplicación 2

De los 6 controles de seguridad revisados se encontraron cubiertos 2, el relacionado con inyecciones de SQL y controles de autenticación del lado del servidor. Se muestran los siguientes resultados en los logs.

Log

2.4 Se encontraron 23 de 54 que contienen controles de autenticación del lado del servidor.

Archivos con controles de autenticación del lado del servidor:
ApplicationConfigurationsController.cs
ApplicationsController.cs
CIPLogsController.cs
DataController.cs
DocumentsController.cs
GemaltoEmployeesController.cs
GeneralLogsController.cs
GridDataViewsController.cs
HomeController.cs
JAAMSLogsController.cs
ListItemsController.cs
LogChangesController.cs
LogEventsController.cs
NotesController.cs
PolicyController.cs
QueueManagerTagsController.cs
SequenceDetailPluginsController.cs
SequencePluginsController.cs
SequencesController.cs
SystemSettingsController.cs
UserProfileController.cs
VectorLogsController.cs
ZonesController.cs

Archivos sin controles de autenticación del lado del servidor:
ChangeLogController.cs
EmployeesController.cs
EnvironmentParametersController.cs
GemaltoEmployeesController.cs
GridDataViewsController.cs
ListItemsController.cs
PackagesController.cs
PublishController.cs
SystemSettingsController.cs
TW_vwApplicationConfigurationsController.cs
TW_vwApplicationsController.cs
TW_vwCIPLogsController.cs
TW_vwJAAMSLogsController.cs
TW_vwLogChangesController.cs
TW_vwLogEventsController.cs
TW_vwLogsController.cs
TW_vwPoliciesController.cs

TW_vwQueueManagerTagsController.cs
TW_vwSelectedSequenceDetailPluginsController.cs
TW_vwSequenceDetailPluginsController.cs
TW_vwSequencesController.cs
TW_vwVectorLogsController.cs
TW_vwZonesController.cs
vw_DocumentsController.cs
vw_ListItemsController.cs
vw_NotesController.cs
vw_SystemSettingsController.cs
AccountsController.cs
EnvironmentParameterController.cs
MenuController.cs
PackageController.cs

3.3 No se encontró definido el tiempo de expiración de la aplicación.
5.10 Se encontraron 11 de 18 consultas con parametrización para no ser susceptibles a la inyección de SQL.
Archivos con parametrización:
PublishController.cs
PackageController.cs
Application_PackageViewModel.cs

Archivos sin parametrización:
TW_vwVectorLogsController.cs
ActionDataHelper.cs

9.4 El encabezado anti caché no se encuentra definido.
10.11 El encabezado HTTP strict-transport-security no se encuentra incluido en las peticiones y para los subdominios.
11.1 No se encontraron los métodos de solicitud HTTP definidos en la aplicación.

La cobertura de controles no automatizables obtuvo **36.19%** mientras que los automatizables **17.33%**.

Propuesta de Mejora

El prototipo presenta una primera iteración de la herramienta para verificación de seguridad en aplicaciones. Fueron implementados solo 6 de los 35 controles automatizables, por lo que se continuaría con la implementación de los controles restantes.

Este prototipo realiza únicamente análisis estático, agregar opciones de análisis dinámico según se requiera en el control aportaría una capa de revisión adicional. En este documento se encuentra detallada la guía de implementación de los controles restantes, la cual sería el medio de apoyo para la implementación futura.

Conclusiones

La aplicación de una guía de mejores prácticas garantiza un seguimiento homólogo con respecto a la seguridad para las aplicaciones desarrolladas.

Contar con una herramienta que analiza los elementos mínimos de seguridad agrega valor a la empresa X. Basándose en un estándar a su vez permite que pueda adaptarse a otros tipos de proyectos de desarrollo.

Bibliografía

- Fidias G Arias (2012) El proyecto de investigación. 6ta edición. Caracas, Venezuela.
- OWASP (2017). Estándar de Verificación de Seguridad en Aplicaciones 3.0.1, España. https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf
- Official Microsoft Developer Network, https://msdn.microsoft.com/en-us/
- Sitio Oficial de OWASP, https://www.owasp.org
- Forbes (2018) Seis ciberataques de los que debes protegerte en 2018, https://www.forbes.com.mx/seis-ciberataques-de-los-que-debes-protegerte-en-2018/
- Patrick Cousot (2000) Paper: Abstract Interpretation Based Formal Methods and Future Challenges. París, Francia.

Anexos

1. Guía de Verificación de Controles No Automatizables

V1. Arquitectura, diseño y modelado de amenazas
1.1 Verificar que todos los componentes de la aplicación se encuentran identificados y asegurar que son necesarios.
1. Verificar por medio del archivo packages.json que los componentes añadidos al proyecto sean los que se requieren

V2. Autenticación
2.2 Verificar que todos los campos de credenciales no reflejen las contraseñas del usuario. Cargar la credencial por parte de la aplicación implica que la misma fue almacenada de forma reversible o en texto plano, lo que se encuentra explícitamente prohibido.
1. Verificar que la contraseña nunca es retornada a la aplicación desde el servidor
2.6 Verificar que los controles de autenticación fallan de forma segura para evitar que los atacantes puedan iniciar sesión.
1. Realizar pruebas de inicio de sesión incorrectas, para confirmar si los errores no indican información adicional.
2. Realizar pruebas de inyección de SQL para validar que la información está parametrizable y no revelará información adicional como la versión de la base de datos

2.8 Verificar que toda función relacionada con la autenticación (como registro, actualización del perfil, olvido de nombre de usuario, recuperación de la contraseña, token perdido / deshabilitado, funciones de help desk o IVR) que pueda ser utilizada de forma indirecta como mecanismo de autenticación, sea al menos tan resistente a ataques como el mecanismo primario.
1. Verificar que no se realiza un inicio de sesión al recuperar la contraseña sin tener que escribirla nuevamente.
2. Verificar que al actualizar el perfil se requiere la contraseña actual.
2.18 Verificar que no es posible enumerar información mediante las funcionalidades de: inicio de sesión, reinicio o recuperación de contraseñas.
1. Realizar pruebas con datos incorrectos para confirmar si los errores no indican información adicional.
2. Realizar pruebas de inyección de SQL para validar si la información está parametrizable y no revelará información adicional.
2.22 Verificar que las funcionalidades de recuperar contraseña y otras formas de recuperar la cuenta utilizan mecanismos de TOTP (Time-Based One-Time Password) u otro tipo de soft token, push a dispositivo móvil u otro tipo de mecanismo de recuperación offline. El uso de un valor aleatorio en un correo electrónico o SMS debe ser la última opción, ya que son conocidas sus debilidades.
1. Verificar que se cuenta con métodos adicionales de recuperación de contraseña, aparte del correo electrónico.
2.32 Verificar que las interfaces administrativas de la aplicación no sean accesibles a intrusos.
1. Verificar que intrusos no puedan acceder al web admin.

2.33 Autocompletar navegadores e integración con gestores de contraseñas debe estar permitido, a no ser que se encuentren prohibidos por políticas de riesgos.
1. Verificar que el autocompletado trabaja en todas las funciones de envío de formularios.

V3. Gestión de Sesiones
3.1 Verificar que no se utiliza un gestor de sesiones personalizado, o que, si el gestor de sesiones es personalizado, este sea resistente contra los ataques más comunes.
1. Verificar que se esté utilizando el Session state de .NET.
3.5 Verificar que todas las páginas que requieren autenticación poseen acceso fácil y visible a la funcionalidad de cierre de sesión.
1. Verificar que la opción de cerrar sesión se encuentra en todas las páginas (master page o menús)
2. Validar que todos los accesos a las páginas cuenten con la etiqueta Authorize.
3.11 Verificar que los identificadores de sesión son suficientemente largos, aleatorios y únicos para las sesiones activas.
1. Verificar que el identificador de sesión que se esté usando sea el SessionID del Session state en .NET
2. Verificar que la sesión sea identificable de forma única (Guid).
3.17 Verificar que una lista de sesiones activas esté disponible en el perfil de cuenta o similar para cada usuario. El usuario debe ser capaz de terminar

cualquier sesión activa.
1. Verificar que existe una funcionalidad para que el usuario pueda administrar sus sesiones.
3.18 Verificar que al usuario se le sugiera la opción de terminar todas las otras sesiones activas después de un proceso de cambio de contraseña exitoso.
1. Iniciar sesión desde otro dispositivo diferente al que se encuentra autenticado para verificar si la opción de cerrar otras sesiones aparece.

V4. Control de acceso
4.2 Verificar que existe el principio de privilegio mínimo - los usuarios solo deben ser capaces de acceder a las funciones, archivos de datos, URL, controladores, servicios y otros recursos, para los cuales poseen una autorización específica. Esto implica protección contra suplantación de identidad y elevación de privilegios.
1. Verificar que la aplicación cuenta con autenticación por roles.
4.4 Verificar que el acceso a registros sensibles esté protegido, de forma tal que solo objetos autorizados o datos sean accesibles por cada usuario (por ejemplo, proteger contra la posible manipulación hecha por usuarios sobre un parámetro para ver o modificar la cuenta de otro usuario).
1. Validar que la información mostrada en el sitio se filtre para el usuario, según su autorización.
2. Validar que las funcionalidades que requieren parámetros en la url no puedan ser manipuladas por el usuario para accesar información que no posee autorización.
4.8 Verificar que los controles de acceso fallen de forma segura.

1. En caso de dar error, verificar que no se muestren errores detallados al usuario.
4.9 Verificar que las mismas reglas de control de acceso implícitas en la capa de presentación son aplicadas en el servidor.
1. Verificar que aunque el usuario no tiene acceso visual a las funcionalidades, tampoco tenga forma de acceder a ellas, por ejemplo, sabiendo la url específica.
4.11 Verificar que todas las acciones de control de acceso pueden ser registradas y que todas las acciones fallidas son registradas.
1. Verificar que se cuenta con registro de las acciones de control de acceso tanto exitosas como no exitosas.
4.12 Verificar que la aplicación o su infraestructura emite tokens anti-CSFR aleatorios y existe otro mecanismo de protección de la transacción.
1. Verificar que los formularios cuentan con @Html.AntiForgeryToken() y en caso de que el token no sea enviado el servidor rechaza la petición.
4.16 Verificar que la aplicación ejecute correctamente la autorización contextual para no permitir la manipulación de parámetros de la URL.
1. Verificar que aunque el usuario manipule las URLs, el servidor valide, según su usuario, si tiene autorización para acceder al contenido solicitado.

V5. Manejo de entrada de datos maliciosos
5.1 Verificar que el entorno de ejecución no es susceptible a desbordamientos de búfer, o que los controles de seguridad previenen desbordamientos de búfer.
1. Verificar que no se use el palabra clave “unsafe”.
5.3 Verificar que las fallas de validación de entradas de datos del lado del servidor sean rechazadas y registradas.
1. Verificar que existe un historial de errores generados en la aplicación.
5.6 Verificar que un único control de validación de entrada es utilizado por la aplicación para cada tipo de datos aceptado.

V7. Criptografía en el almacenamiento
7.2 Verificar que todos los módulos criptográficos fallen de forma segura, y que los errores sean manejados de tal manera que no permitan ataques Oracle padding.
1. Verificar que no se muestren errores detallados sino un error genérico en caso de que falle la encriptación.

V8. Gestión y registro de errores
8.1 Verificar que la aplicación no emita mensajes de error o rastros de pilas que contengan datos sensibles que podrían ayudar a un atacante, incluyendo el identificador de sesión, versiones de software/entorno y datos personales.
1. Verificar si están implementados los errores customizados para evitar

mostrar información sobre el error que guíe al atacante.

V9. Protección de datos
9.1 Verificar que a todos los formularios con información sensible se les haya desactivado el almacenamiento de caché en el cliente, incluyendo funciones de autocompletar.
1. Verificar que los campos de los formularios contengan la etiqueta autocomplete="off" cuando se refieren a información sensible.
9.3 Verificar que toda información sensible es enviada al servidor en el cuerpo o cabeceras del mensaje HTTP (por ejemplo, los parámetros de la URL nunca se deben utilizar para enviar datos sensibles).
1. Verificar que toda llamada que envíe dato sensible use métodos de POST.
9.9 Verificar qué datos almacenados en el cliente (como almacenamiento local de HTML5, almacenamiento de la sesión, IndexedDB, cookies normales o las cookies de Flash) no contengan información sensible o información personal identificable.
1. Verificar, en cada funcionalidad, la información almacenada localmente en el navegador.

V10. Comunicaciones
10.13 Asegurar que forward secrecy se esté utilizando para mitigar que atacantes pasivos puedan grabar el tráfico.
1. Revisar en las conexiones que en el intercambio de llaves se esté utilizando

el algoritmo Diffie Hellman.
10.14 Verificar que una adecuada revocación de certificados, tal como el protocolo de estatus de certificado en línea (OSCP), está habilitado y configurado para determinar el estado de vigencia del certificado.
1. Para realizar la verificación de OCSP en el código se necesita implementar una librería cómo Bounty Castle.
10.15 Verificar que se utilicen únicamente algoritmos, cifradores y protocolos fuertes, a través de toda la cadena de confianza, incluyendo certificados raíz y certificados intermediarios de la autoridad certificadora seleccionada.
1. Verificar que los algoritmos, cifradores y protocolos fuertes sean los recomendados por la industria.
10.16 Verificar que la configuración de TLS esté en línea con las mejores prácticas actuales, particularmente debido a que configuraciones comunes se convierten en inseguras a medida que transcurre el tiempo.

V11. Configuración de seguridad HTTP
11.7 Verificar que la política de seguridad de contenido (CSPv2) está en uso de tal manera que ayude a mitigar vulnerabilidades de inyección comunes de DOM, XSS, JSON y Javascript.
1. Verificar si en los headers está definido CSP (Content-Security-Policy).

V16. Archivos y recursos.
16.3 Verificar que los archivos procedentes de fuentes no confiables sean validados para ser del tipo del cual se espera y sean analizados por escáneres

antivirus para evitar la carga de contenido malicioso conocido.
1. Verificar que exista un antivirus instalado en el servidor que se pueda acceder desde la línea de comandos para escanear los archivos subidos.
16.5 Verificar que datos no confiables no se utilicen en recursos de dominios compartidos (CORS) para proteger contra el contenido remoto arbitrario.
1. Verificar que el contenido recibido de un dominio confiable es sanitizado antes de utilizarse para prevenir la ejecución de código malicioso.
16.9 Verificar que no utiliza Flash, Active-X, Silverlight, NACL, Java u otras tecnologías del lado del cliente, que no sean soportadas de forma nativa a través de los estándares de navegador W3C.
1. Al ingresar al sitio no debería aparecer información solicitando permiso para ejecutar Flash, Silverlight o Java.

V19. Requisitos de Configuración
19.1 Todos los componentes deben estar actualizados a las configuraciones y versiones de seguridad adecuadas. Esto debería incluir la eliminación de configuraciones y carpetas innecesarias como aplicaciones de ejemplo, documentación de plataforma y usuarios preestablecidos o de ejemplo.
1. Verificar que los componentes sean revisados periódicamente para actualizaciones y configuraciones de hardening.