PRESENTE,

Acciones a considerar para la correcta liberación de un sitio web.

Por lo general, los sitios web, una vez liberados, quedan disponibles para cualquier usuario en la vasta red internacional de internet, y muchas veces, son puestos a prueba malintencionadamente para ganar acceso a recursos, espacio de disco, secuestro de información o de recursos, la mayoría de estos ataques no buscan dejar evidencia de su intención por lo que los atacantes experimentados pueden hacer uso del sistema vulnerado por mucho tiempo antes de que sea descubierto alguna anomalía.

1. Control de acceso a usuarios.

Passwords.
Las contraseñas de acceso pueden encontrarse en diferentes instancias en el proceso de hospedaje de un sitio web, transversalmente, los diferentes dueños de procesos deben tener en cuenta las siguientes consideraciones, y sea en las contraseñas de acceso a la plataforma, al sistema operativo, el usuario con el que corre el servicio, el usuario para la conexión de la base de datos o de la api, etc., deben de ser contraseñas fuertes, en el entendido de que una contraseña fuerte puede componerse de los siguientes componentes:

- Longitud mínima requerida. - Lo habitual es al menos 8 caracteres para evitar descubrimiento rápido por permutación de caracteres.
- Mayúsculas. – Duplica el set de caracteres a evaluar, si se utiliza en combinación con minúsculas.
- Minúsculas. – Duplica el set de caracteres a evaluar, si se utiliza en combinación con mayúsculas.
- Caracteres Especiales, incluyendo espacios en blanco. – La intención es incrementar la aleatoriedad de la posición de un elemento adicional y un juego de caracteres ampliado.
- Números. - Incrementa el set de caracteres a evaluar, si se utiliza en combinación con mayúsculas o minúsculas.
- Complejidad. – Probablemente la más difícil de considerar, ya que por costumbre utilizamos técnicas para recordar las contraseñas, la aleatoriedad puede incrementar el tiempo de convergencia de algoritmos de fuerza bruta, sin embargo, si es suficientemente corta no tiene mucha ventaja contra algoritmos secuenciales de fuerza bruta.

Frase desafío. (Passphrase).
La frase desafío se utiliza comúnmente como una contraseña fuerte, consiste en aprovechar la habilidad nemónica del ser humano para poder referenciar una frase semi encriptada, como ejemplo podemos decir que una contraseña fuerte de acuerdo a los componentes descritos puede ser:

Contraseña	Complejidad	Romper Encriptación

“Gobierno0”	Buena	42 minutos
“Gobierno.0”	Fuerte	10 años
“G0b13rn0.0”	Muy fuerte	10 años
“Gobierno cero.0”	Muy fuerte	34 billones de años
“G0b13rn0 Un0. C3ro”	Muy fuerte	64 billones de años

En el ejemplo, el tiempo que llevaría romper la encriptación hipotéticamente sería utilizando un algoritmo mono proceso serial, las técnicas actuales pueden hacer que equipos poderosos en recursos disminuyan este tiempo estimado. No obstante, queda claro en la comparación, el beneficio de la complejidad de una frase desafío contra una contraseña tradicional, aunado a la facilidad de recordar la frase.

Contraseñas de diccionario.

Existen aplicaciones que se dedican a probar sistemas de autenticación enviando por script las instrucciones de un intento de acceso, y apoyados en bases de datos de contraseñas comunes, conocidas y combinaciones de las mismas. Programas malintencionados (malware) contribuyen a la alimentación y enriquecimiento de estas bases de datos tomando datos del usuario como el nombre o fechas que existan en documentos y agregándolas al diccionario, algunas muestras de diccionario son accesibles desde internet, una buena práctica consiste en no utilizar contraseñas que incluyan palabras que existan en alguna muestra de diccionario, o que asemejen un patrón establecido por estas palabras diccionario. Algunos de los ejemplos más comunes de contraseñas diccionario (idioma inglés):

123456	111111	princess	football	Charlie
password	1234567	admin	123123	aa123456
123456789	sunshine	welcome	monkey	Donald
12345678	Qwerty	666666	654321	password1
12345	Iloveyou	abc123	!@#$%^&*	qwerty123

Compartición de contraseñas.

Salvo sea un requisito de diseño, se desaconseja que más de un usuario utilice las mismas credenciales para cualquier etapa del flujo de la construcción y funcionamiento de una plataforma web. Las contraseñas de sistema o globales deben estar documentadas y debidamente resguardadas pues comprometen la continuidad del servicio de la plataforma web.

2. Software actualizado.

Las plataformas de sistema operativo, servidor web, interprete cgi, apis de desarrollo, paquetes de compilación, librerías de uso público, frameworks de diseño y de scripting, etc, son elementos que comúnmente se encuentran en la composición de una plataforma web.
Las actualizaciones de cada uno de éstos componentes, no solamente incluyen mejoras o adiciones a las características funcionales, sino que también consideran las correcciones de algunos bugs de seguridad, los cuales casi siempre se encuentran documentados. Un atacante puede utilizar la información disponible para una vulnerabilidad específica existente en una versión desactualizada para ganar acceso y privilegios a través de la falla. Contar con un esquema de actualización de paquetes y parches es necesario para conservar una pequeña ventaja en la posibilidad de tener vulnerabilidades bien conocidas y bien documentadas.

3. Registros de actividades y su resguardo.

Los archivos de registro de actividades pueden ser de lo más variado, desde el sistema operativo, intérprete, e incluso en el desarrollo del mismo código de la plataforma pueden existir elementos que constantemente registran en archivos de texto las actividades de cada componente. Éstos elementos de registro son conocidos comúnmente con el nombre de ***Logs***. Se debe contar con un procedimiento que garantice la integridad de estos registros, incluso el respaldo remoto de los mismos. Esto permite trazabilidad y evidencia en caso de algún incidente.

4. Validación básica de paso de parámetros y formularios.

Paso de parámetros
Si es requerido el paso de parámetros, se prefiere el paso de parámetros sensibles mediante el método POST, respetando los requerimientos y mejores prácticas del área de desarrollo involucrado, pues presenta ligeramente mejores beneficios en el tema de seguridad ya que uno pasa los parámetros y sus valores en la url (GET) y el otro lo hace dentro del cuerpo del código (POST), como referencia, presentamos a continuación algunas diferencias clave entre los métodos GET y POST.

Acción	GET	POST
Botón Retroceder / Recargar	Sin Efecto	Los datos serán reenviados (el navegador deberá advertir al usuario de esta situación)
Añadir a Favoritos	Se permite, la url y parámetros quedarán como un acceso directo.	No se pueden agregar los parámetros a una url de favoritos.
Cache	Los datos se almacenan en cache.	No se almacena en cache.
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded o multipart/form-data.

Acción	GET	POST
		(Utiliza multipart encoding para datos binarios)
Historial	Los parámetros quedan almacenados en el historial del navegador.	Los parámetros no son almacenados en el historial del navegador.
Restricciones en longitud de datos.	Si, cuando se envían datos, el método GET agrega los datos a la URL; y la longitud de la URL es limitada (2048 caracteres, y muchos menos caracteres bajo esquemas de protección de equipos firewall).	No hay restricciones.
Restricciones en tipos de datos	Solamente caracteres ASCII son admitidos	No hay restricciones, se admiten datos binarios.
Seguridad	El método GET es menos seguro, comparado con el método POST porque los datos enviados son parte de la URL.	El método POST es un poco más seguro que el método GET, debido a que los parámetros no son almacenados en el navegador, historial, ni en los logs del servidor.
Visibilidad.	Los datos de los parámetros son visibles para todos, por estar en la URL.	Los datos no son visibles en la URL

Validación de controles input.

Adicionalmente, los parámetros pueden estar accionados por el código de la plataforma, como procedimiento interno, un aspecto importante es que los valores de algunos parámetros requieren la entrada del usuario, ésta entrada de datos se realiza por controles input de formularios web.

Los controles input de formularios web, son susceptibles a ser descuidados en las etapas finales de validación al liberar un sitio, un input sin la validación de las cadenas que se almacenan en el valor, puede pasar parámetros directamente al siguiente motor de evaluación, dentro de los cuales, los más comunes son los siguientes:

- SQL Injection.
- XSS attacks.

La mejor manera de mitigar estos riesgos es contar con un procedimiento para validar, escapar y moldear los valores recibidos en un input, antes de pasarlos como parámetros a la siguiente etapa de ejecución. Aplica también a la impresión de valores almacenados en la BD, ya que en caso de almacenarse una instrucción maliciosa en la bd, la impresión del valor, como un código script, por ejemplo, pasaría al motor de ejecución sin ser ésta la intención del funcionamiento de la plataforma desarrollada.

SQL Injection.

Consiste en el paso malintencionado de comandos SQL al motor intérprete encargado de ejecutar dichos comandos sobre la base de datos, de esta manera, se pretende explotar la lógica de

programación creada para la consulta o almacenamiento de un valor, o almacenar instrucciones en la BD, para que sean ejecutadas al imprimir el valor en otra consulta.
Se pretende entonces, prevenir el agregar al dato en el input, un valor que siempre resulte verdadero, permitiendo a la consulta ejecutar la consulta independientemente de que el valor sea válido o existente, una variante consiste en añadir una instrucción SQL al valor, pretendiendo que se ejecute como si fuese parte de la consulta original, a modo de procesamiento por lotes de las instrucciones.

Consulta en código cgi	Valor esperado	Valor malintencionado	Resultado esperado	Resultado malintencionado
SELECT * FROM Users WHERE UserId =	105	105 OR 1=1	Dataset: John Smit.	Dataset: All Table Users
SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass"	John, Doe	" or ""=", "or ""="	Dataset: John Doe	Dataset: All Table Users
SELECT * FROM Users WHERE UserId =	105	105; DROP TABLE Suppliers	Dataset: John Smit.	Dataset: John Smit, Eliminacion de tabla Suppliers

Una buena práctica consiste en la parametrización del valor en la consulta interna codificada, evitando ejecuciones de SQL no intencionales. Otra buena práctica consiste en la limpieza de las cadenas pasadas como parámetros, eliminando caracteres de las cadenas de texto en el valor, que sean interpretados por el lenguaje sql, como operadores. Un ejemplo son los símbolos de operadores (%=;|"><$&+-/.?) AND, OR, LIKE.

Ataques XSS.
Como en el caso anterior, el ataque consiste en explotar la falta de validación de cadenas en los controles input de los formularios, para intentar pasar instrucciones de script activo, sean jquery, javascript, vbscript, e incluso instrucciones cgi (por ejemplo <?php echo ""?>), en la expectativa de que el siguiente motor interprete las instrucciones como parte del código original.
Un ejemplo de esto sería pasar una instrucción meta redirect de un control input, al almacenamiento en la BD de la plataforma, esta instrucción puede ser impresa en un listado, cuando el jquery intente darle formato a la lista, ejecutará también la instrucción impresa. Este comportamiento pudiera ser aprovechado para suplantar la plataforma legitima por una versión falsificada de la misma, intentando obtener valores que un usuario inadvertidamente introduzca, pensando que se encuentra en la plataforma original. Una prueba simple es introducir instrucciones echo o print en los controles de input para observar si poseen esta vulnerabilidad.
Para ambos casos, SQL injection y XSS Attack, siempre es recomendable que las validaciones a las cadenas de los controles input sean realizadas tanto del lado cliente como en el servidor que recibe el parámetro.

5. Content Security Police (CSP).

Las cabeceras de respuesta Content-Security-Policy, se presentan en la repuesta del servidor de la plataforma web, un cliente compatible reforzará la declaración de la política de lista blanca del navegador. Por ejemplo, se puede incluir en la política una ejecución de código JavaScipt mas estricta, tratando de prevenir ataques XSS, en realidad esta implementación conlleva una serie de características que son desactivadas por default:

- Código JavaScript inline.
  - Bloques <script>
  - Manejadores de evento DOM como atributos html (como por ejemplo .onclick)
  - Ligas url javascript: (en la barra url)
- Indicaciones de CSS inline.
  - Bloques <style>
  - Atributos style en elementos html
- Evaluación de código dinámico.
  - eval()
  - argumentos de texto para las funciones setTimeout y setInterval
  - el constructor new Function()
- Instrucciones CSS dinámicas.
  - El método CSSStylesheet.insertRule()

Desarrollar una plataforma nueva mientras se utiliza una CSP es un desafío, pero existen plataformas de desarrollo compatibles con CSP, es un hecho que muchas aplicaciones existentes deben cumplir rediseño o una política CSP no tan severa. La recomendación de buena práctica es utilizar el código css y script mediante las llamadas de carga de archivos externos (<script src>), interpretar vía json en vez de evaluarlos y utilizar EventTarget.addEventListener() para asignar manejadores de eventos.

6. Ocultamiento de mensajes de error y debug.

El ocultamiento de mensajes de error y de depuración, en una buena práctica por si misma, ya que ayuda a minimizar la información acerca de la plataforma en la que se ejecuta o se encuentra desarrollada la plataforma. De este modo dificulta al operador malintencionado, hacerse de información acerca de la versión, plataforma, web server o cgi sobre los cuales se ejecuta la aplicación.
Debe existir dentro del procedimiento de liberación de una plataforma web, una etapa que sea dedicada a la validación del ocultamiento de los mensajes de error tipo y mensajes de depuración. Si debe existir mensajes de error, deben ser los mensajes desarrollados para este fin específico, con mensajes propios del desarrollo y no del sistema.

7. Subida de archivos.

Se recomienda no utilizar métodos que permitan subir archivos al servidor mediante la plataforma web. Si no existe una alternativa, existen algunas consideraciones que se vuelven obligatorias en su implementación:

- Remoción de privilegios de ejecución del archivo.
- Renombramiento de las extensiones del archivo vía validación.
- Validación de existencia de comandos de ejecución en archivos de texto plano.

En la medida que algunos controles no puedan ser del todo implementado, se requiere considerar las implicaciones que conlleva la escritura de un archivo tipo scritp o shell en las carpetas del servidor, y de cómo esto representa una posibilidad latente de secuestro del servidor o permita otorgar puertas traseras al servidor mediante ejecución de comandos al nivel del sistema operativo.

8. Aislamiento de servidores de producción. DMZ

Es recomendable que la infraestructura de procesamiento se ejecute en un entorno que pueda ser monitoreado, que sus accesos puedan ser regulados y que se puedan determinar y clasificar intentos de conexiones no relacionadas al servicio que proporciona. Este entorno muchas veces se denomina DMZ (*demilitarized zone).* El concepto sugiere que los servidores puedan tener acceso a internet para la publicación de sus servicios, al mismo tiempo que el acceso de los mismos servidores hacia la red interna se encuentra restringido, de modo tal que, en caso de comprometer un servidor alojado en la dmz, no se compromete el acceso hacia los recursos de la red interna.

9. Uso de cifrado ssl.

El sitio debe seguir las recomendaciones para ser hospedado mediante el servicio de https, cada vez más máquinas de búsqueda e indexación favorecen en los resultados a los sitios que si cuentan con https sobre los que no lo hacen. Adicionalmente el servicio se ofrece cifrado de punto a punto entre el navegador visitante y la plataforma, dificultando la captura de datos al escuchar el medio de transporte.

10. Respaldos.

Se debe contar con un esquema que permita la recuperación en caso de incidentes sobre los componentes que soportan la plataforma web, en esencia, se debe contar con un esquema de respaldos que garanticen la integridad del código fuente, parámetros operativos del servidor, BD y webservice. Así como el control de las versiones de los elementos de librería utilizados. Se debe considerar respaldar, para recuperación en caso de falla:

- El código fuente de la plataforma.
- El código de las librerías y plataformas.
- Los instaladores del software de desarrollo.
- Bases de datos.
- Parámetros operativos de:
  - Sistema operativo

- Web Server
- Syslog
- Sistema de administración de BD.

11. Security checkups.

Se debe considerar la tarea de realizar security checkups sobre las plataformas desarrolladas, estas pruebas pueden realizarse en coordinación con el equipo de respuesta a incidentes, los resultados pueden arrojar posibles puntos de mejora, en caso de no ser aceptable el riesgo evaluado tras el uso de estos checkups.
Algunas plataformas que pueden utilizarse para la realización de los security checkups:

- Netsparker
- OpenVAS
- Security Headers – IO
- Xenotix XSS Exploit Framework.

(resultado de https://securityheaders.com/?q=www.yucatan.gob.mx&followRedirects=on)