

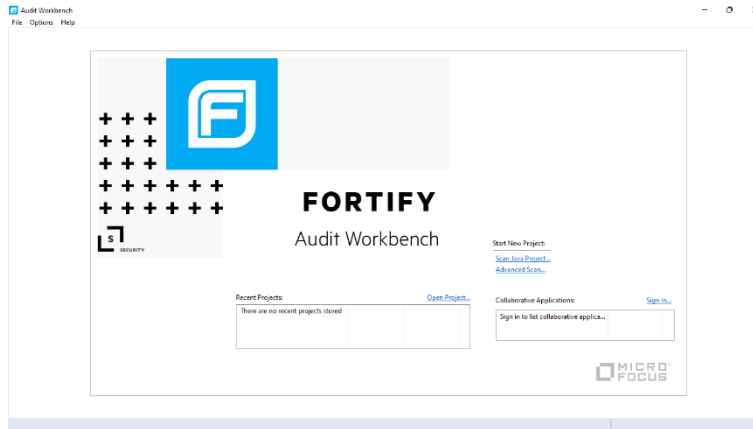
Asignatura	Datos del alumno	Fecha
Diseño y Desarrollo de Programas Informáticos Seguros	Apellidos: Paz López	02/01/2022
	Nombre: Angel Ramón	

ÍNDICE

DESCRIPCIÓN DE LA ACTIVIDAD.....	2
DESARROLLO DE LA ACTIVIDAD	2
1. ASTERISX - BUFFER OVERFLOW (CWE-120, 129, 131, 187).....	2
2. ASTERISX - BUFFER OVERFLOW: FORMAT STRING (CWE-134, 787)	4
3. ASTERISX - INSECURE TRANSPORT: WEAK SSL PROTOCOL (CWE-326)	5
6. ASTERISX - BUFFER OVERFLOW: OFF-BY-ONE (CWE 129, 131, 193, 787, 805)	6
7. ASTERISX - PATH MANIPULATION (CWE-22, 73).....	7
8. ASTERISX - PRIVACY VIOLATION (CWE-359)	8
9. ASTERISX - PASSWORD MANAGEMENT: HARDCODED PASSWORD (CWE-259, 798) 10	
10. ASTERISX - WEAK ENCRYPTION: INSECURE INITIALIZATION VECTOR (CWE-329) .	11
11. ASTERISX - INSECURE RANDOMNESS (CWE-338)	12
12. ASTERISX - NULL DEFERENCE (CWE-476).....	13
13. ASTERISX - USE AFTER FREE (CWE-416).....	14
14. ASTERISX - FORMAT STRING (CWE-364).....	15
15. THE BODGEIT STORE - CROSS-SITE SCRIPTING: PERSISTENT (CWE-79, 80).....	15
16. RACE CONDITION: SINGLETON MEMBER FIELD (CWE-362, 488)	17
17. PUZZLE MALL - PASSWORD MANAGEMENT: HARDCODED PASSWORD	18
CONCLUSIONES.....	19
REFERENCIAS	20

DESCRIPCIÓN DE LA ACTIVIDAD

Se realiza la auditoria del archivo **Asterisx limpio results.fpr**, incluido en la carpeta **LaboratorioISW**, con la herramienta **HP Fortify**, con el objetivo de encontrar vulnerabilidades que se pueden encontrar en el archivo antes mencionado.



Este ejercicio ayudara a conocer una de las herramientas de más alto nivel que asisten la revisión de código y el flujo de trabajo sobre los mismos.

DESARROLLO DE LA ACTIVIDAD

1. ASTERISX - BUFFER OVERFLOW (CWE-120, 129, 131, 187)

Archivo: evsub.c

Línea: 794

Código:

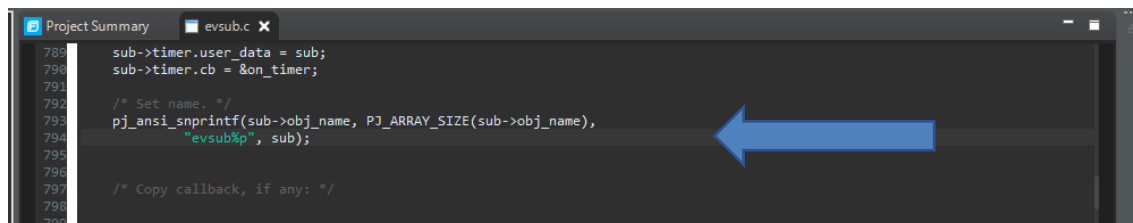
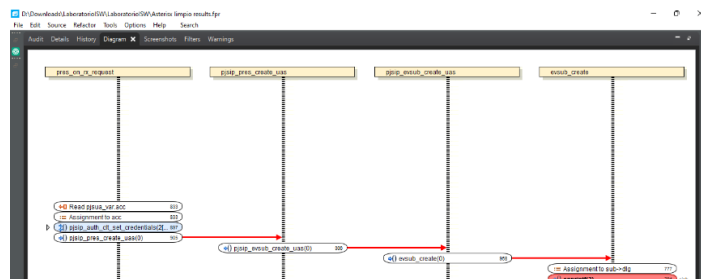


Diagrama:



Explicación:

Es considerada una vulnerabilidad crítica en el programa donde la a función `evsub_create()` en `evsub.c` podría escribir fuera de los límites de la memoria asignada en la línea 794, lo que podría dañar los datos, y hacer que el programa se bloquee o provocar la ejecución de código malicioso. El desbordamiento de búfer es probablemente la forma más conocida de vulnerabilidad de seguridad del software. En un exploit clásico de desbordamiento de búfer, el atacante envía datos a un programa, que almacena en un búfer de pila de tamaño insuficiente. El resultado es que la información de la pila de llamadas se sobrescribe, incluido el puntero de retorno de la función. Los datos establecen el valor del puntero de retorno para que cuando la función regrese, transfiera el control al código malicioso contenido en los datos del atacante.

Aunque este tipo de desbordamiento de búfer de pila todavía es común en algunas plataformas y en algunas comunidades de desarrollo, hay una variedad de otros tipos de desbordamiento de búfer, incluidos los desbordamientos del búfer de pila y los errores de uno por uno, entre otros. A nivel de código, las vulnerabilidades de desbordamiento de búfer generalmente implican la violación de las suposiciones de un programador. La combinación de manipulación de la memoria y suposiciones erróneas sobre el tamaño o la composición de un dato es la causa principal de la mayoría de los desbordamientos de búfer. Las vulnerabilidades de desbordamiento de búfer generalmente ocurren en código que:

- Se basa en datos externos para controlar su comportamiento.
- Depende de las propiedades de los datos que se aplican fuera del alcance inmediato del código.
- Es tan complejo que un programador no puede predecir con precisión su comportamiento.

Recomendaciones:

En este caso, nos preocupa principalmente el segundo caso, porque no podemos verificar la seguridad de la operación realizada por `snprintf()` en `evsub.c` en la línea 794. No utilizar funciones intrínsecamente inseguras como `gets()`, y si es posible evitar el uso de funciones que sean difíciles utilizar de manera segura como `strcpy()`, el uso cuidadoso de funciones limitadas reducirá en gran medida el riesgo de desbordamiento de buffer, siempre que manipulemos la memoria, especialmente las cadenas es importante recordar las vulnerabilidades de desbordamiento de buffer.

2. ASTERISX - BUFFER OVERFLOW: FORMAT STRING (CWE-134, 787)

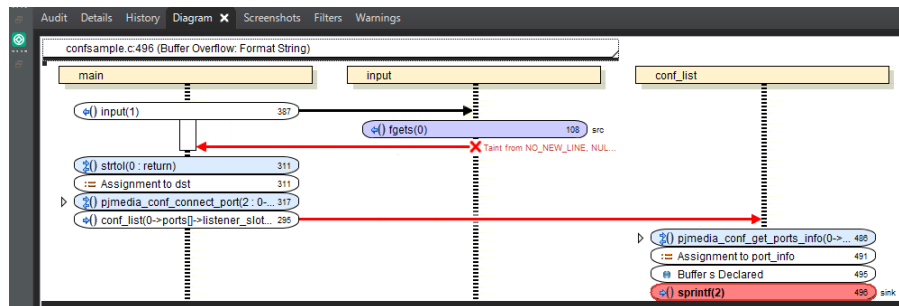
Archivo: confsample.c

Línea: 496

Código:

```
481  
482  
483     printf("Conference ports:\n");  
484  
485     count = PJ_ARRAY_SIZE(info);  
486     pjmedia_conf_get_ports_info(conf, &count, info);  
487  
488     for (i=0; i<count; ++i) {  
489         char txlist[4*MAX_PORTS];  
490         unsigned j;  
491         pjmedia_conf_port_info *port_info = &info[i];  
492  
493         txlist[0] = '\0';  
494         for (j=0; j<port_info->listener_cnt; ++j) {  
495             char s[10];  
496             pj_ansi_sprintf(s, "%d ", port_info->listener_slots[j]);  
497             pj_ansi_strcat(txlist, s);  
498         }  
499  
500
```

Diagrama:



Explicación:

Considerada una vulnerabilidad crítica donde el argumento de cadena de formato para `sprintf()` en `confsample.c` línea 496 no limita adecuadamente la cantidad de datos que la función puede escribir, lo que permite que el programa escriba fuera de los límites de la memoria asignada. Este comportamiento podría dañar los datos, bloquear el programa o provocar la ejecución de código malicioso.

Recomendaciones:

- Evitar que todas las funciones en sus parámetros admitan el operador `%n` en cadenas de formato.
- Asegurarse de que todas las funciones de cadena de formato pasen una cadena estática con el objetivo de no ser controlada por un usuario y de que se envíe el número adecuado de argumentos.
- Usar el método `snprintf()`. Este método realiza la misma función, con la diferencia que permite controlar el número de caracteres.

3. ASTERISX - INSECURE TRANSPORT: WEAK SSL PROTOCOL (CWE-326)

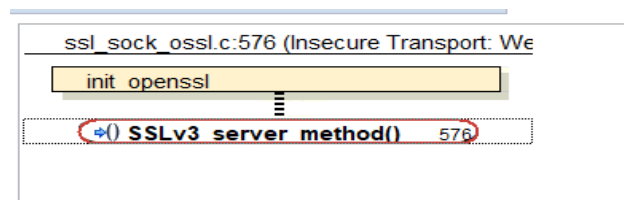
Archivo: evsub.c

Línea: 794

Código:

```
Project Summary  confsample.c  jbsim.c  ssl_sock_ossl.c x
569  || OPENSSL_VERSION_NUMBER < 0x10100000L
570
571  meth = (SSL_METHOD*)SSLv23_server_method();
572  if (!meth)
573  meth = (SSL_METHOD*)TLSv1_server_method();
574  #ifndef OPENSSL_NO_SSL3_METHOD
575  if (!meth)
576  meth = (SSL_METHOD*)SSLv3_server_method();
577  #endif
578  #ifndef OPENSSL_NO_SSL2
579  if (!meth)
580
581
582
583
584
585
586
587  pj_assert(meth);
588
589  ctx=SSL_CTX_new(meth);
590  SSL_CTX_set_cipher_list(ctx, "ALL:COMPLEMENTOFALL");
591
```

Diagrama:



Explicación:

Los siguientes protocolos contienen diversas fallas que los convierten inseguros: SSLv2, SSLv23 y SSLv3 por lo que se debe evitar el uso de estos para transmitir datos confidenciales. Los protocolos TLS (Transport Layer Security) y SSL (Secure Sockets Layer) brindan un instrumento de protección que permite garantizar la autenticidad, confidencialidad e integridad de los datos que son transmitidos entre el cliente y el servidor. Si se tiene una versión insegura de TLS/SSL se tendrá una debilidad en la protección de los datos y podría permitir que un atacante comprometa, robe o modifique la información confidencial. Las versiones que son consideradas débiles de TLS/SSL llegan a presentar las siguientes características:

- No hay protección contra ataques man-in-the-middle.
- No hay protección contra el cierre de la conexión TCP.
- Se cuenta con la misma clave para la autenticación y el cifrado.
- Autenticación de mensajes débil.

Las descritas características pueden permitir al atacante modificar, interceptar o manipular datos confidenciales.

Recomendaciones:

Algunas de las recomendaciones que se tienen para versiones débiles de TLS/SSL es desactivar la versión en este caso SSLv3 y reemplazar con TLS 1.2 o superior.

Configurar en su servidor web para no permitir el uso de cifrados débiles.

- Para Apache, se debe ajustar la directiva SSLProtocol que es habilitada por el módulo mod_ssl. Esta directiva se puede establecer a nivel servidor o en la configuración del host.
- Protocolo SSL + TLSv1.2
- Para Microsoft IIS, se deben realizar cambios en el registro del sistema.
 1. Inicio -> Ejecutar -> escribe regedt32 o regedit -> click Aceptar
 2. En el editor del registro, busque la siguiente clave o crear si no existe:
 3. HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Control \ SecurityProviders \ SCHANNEL \ Protocols \ SSL 3.0 \
 4. Buscar el clave server o crear si no existe
 5. Debajo del server clave, buscar el valor DWORD con la etiqueta enabled o crear en caso de no existir y establecer como valor 0.

6. ASTERISX - BUFFER OVERFLOW: OFF-BY-ONE (CWE 129, 131, 193, 787, 805)

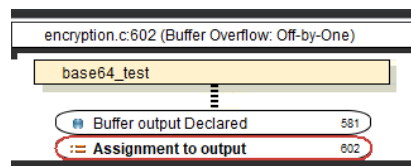
Archivo: encryption.c

Línea: 602

Código:

```
598
599     if (out_len != (int)strlen(base64_test_vec[i].base64))
600         return -91;
601
602     output[out_len] = '\0';
603     if (strcmp(output, base64_test_vec[i].base64) != 0)
604         return -92;
605
606 }
```

Diagrama:



Explicación:

Aunque pareciera que se tratara de un desbordamiento de búfer se advierte que lo asignado en la línea 602, se trata de la terminación de una cadena de caracteres. Es una operación que sirve para convertir el búfer en una cadena de caracteres terminada en null.

Recomendaciones:

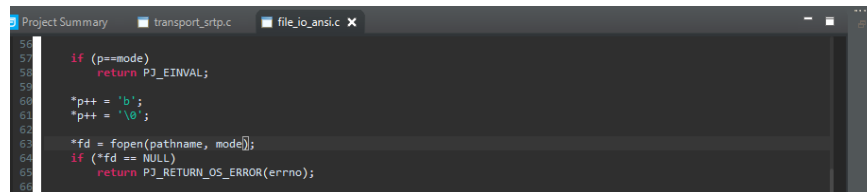
“Not an issue” No es posible que exista un desbordamiento de búfer porque esta operación es perfectamente válida en el lenguaje usado por este proyecto y sirve para convertir el búfer en una cadena de caracteres terminada en null.

7. ASTERISX - PATH MANIPULATION (CWE-22, 73)

Archivo: file_io_ansi.c

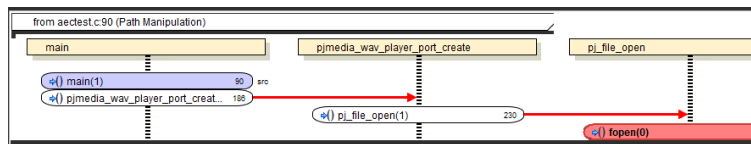
Línea: 63

Código:



```
56
57 if (p==mode)
58     return PJ_EINVAL;
59
60 *p++ = 'b';
61 *p++ = '\\0';
62
63 *fd = fopen(pathname, mode);
64 if (*fd == NULL)
65     return PJ_RETURN_OS_ERROR(errno);
66
```

Diagrama:



Explicación:

Al aceptar que la entrada del usuario tenga control sobre las rutas que son utilizadas en las diferentes operaciones del sistema de archivos se podría dar acceso al atacante o habilitar que modifique los recursos del sistema protegidos. Los atacantes pueden manipular el argumento de la ruta del sistema de archivos para fopen() en file_io_ansi.c línea 63, de manera que pueden acceder o modificar archivos protegidos de otro modo.

Las vulnerabilidades por manipulación de la ruta de archivo surgen cuando los datos controlables de usuario se colocan dentro de un archivo o una URL que es utilizado en el servidor para el acceso a los recursos locales y se pueden dar dos condiciones:

- El atacante puede ingresar una ruta que es usada en una operación en el sistema de archivos.
- Cuando se especifica el recurso y se le concede al atacante una nueva capacidad que de otra forma no se encuentra permitida.

El código puede otorgarle al atacante la capacidad de sobrescribir el archivo o ejecutarlo mediante una configuración controlada por el atacante. Como podemos observar, el atacante

puede especificar el valor que se inserta en el programa main() en aectest.c en la línea 90, y este valor se usa para acceder a un recurso del sistema de archivos en fopen () en file_io_ansi.c en la línea 63.

Recomendaciones:

Se sugiere que el diseño y la funcionalidad de la aplicación se tome en cuenta que los datos controlables por el usuario no se tengan que colocar en rutas de archivo o URL para acceder a los recursos locales en el servidor. Haciendo referencia a los archivos por medio de un número de índice en lugar de utilizar el nombre. Si es inevitable colocar los datos del usuario en rutas de archivo o URL, los datos deben ser validados estrictamente con una lista blanca de valores aceptados. Se tiene que considerar que al acceder a recursos dentro de la raíz web mediante el bloqueo de entrada que contengan secuencias de recorridos de ruta de archivo no siempre es suficiente para evitar la recuperación de la información confidencial.

También tome en cuenta el utilizar listas negras que permiten rechazar o escapar de manera puntual a los personajes potencialmente peligrosos antes de utilizar la entrada, se debe tomar en cuenta que siempre se encuentre incompleta o quede desactualizada por lo que hay que procurar mantenerla actualizada y correcta por si estos llegasen a cambiar.

8. ASTERISX - PRIVACY VIOLATION (CWE-359)

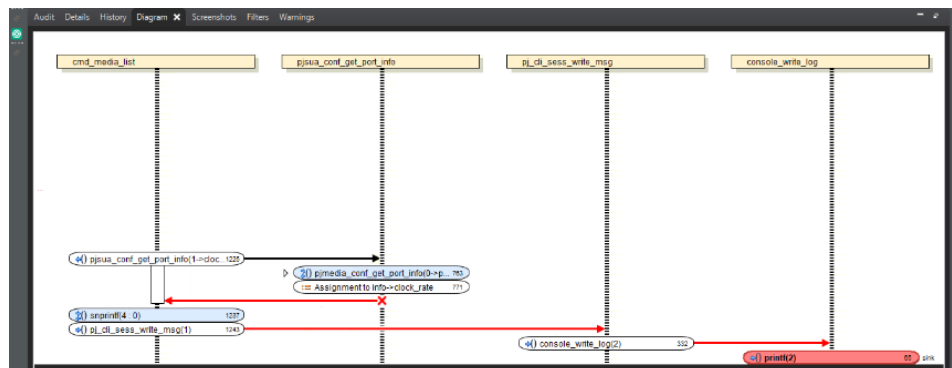
Archivo: cli_console.c

Línea: 65

Código:

```
61  
62 struct cli_console_fe * cfe = (struct cli_console_fe *)fe;  
63  
64 if (cfe->sess->log_level > level)  
65     printf("%s", (int)len, data);  
66 }  
67  
68 static void console_quit(pj_cli_front_end *fe, pj_cli_sess *req)  
69
```

Diagrama:



Explicación:

En muchas de las ocasiones consideramos que se deben registrar todas las operaciones importantes para posteriormente poder analizar e identificar las actividades que resulten anómalas, pero en términos de información confidencial o datos privados el contar con este tipo de práctica puede conllevar a un riesgo. Las violaciones a la privacidad ocurren cuando:

- La información confidencial del usuario es ingresada al programa.
- Los datos que se escriben de manera externa, consola o la red.

Los datos confidenciales pueden ingresar a un programa de distintas formas:

- Del usuario directamente en forma de información personal o su contraseña.
- Mediante el acceso desde una base de datos o un almacén de datos por la aplicación.
- Mediante un tercero o socio de forma indirecta.

Las vulnerabilidades de privacidad de alto perfil, recopilación y gestión de datos confidenciales cada vez se encuentran más reguladas y se toman en cuenta diversos factores como la ubicación, el tipo de operación que se realiza y la naturaleza de los datos por lo que las organizaciones deben cumplir con distintas regulaciones tanto internacionales, federales y estatales. Como podemos observar en el código anterior, se tiene una impresión directa en la función `console_write_log ()` en donde el usuario puede ingresar en la consola información confidencial (contraseña) considerándose un riesgo de privacidad.

Recomendaciones:

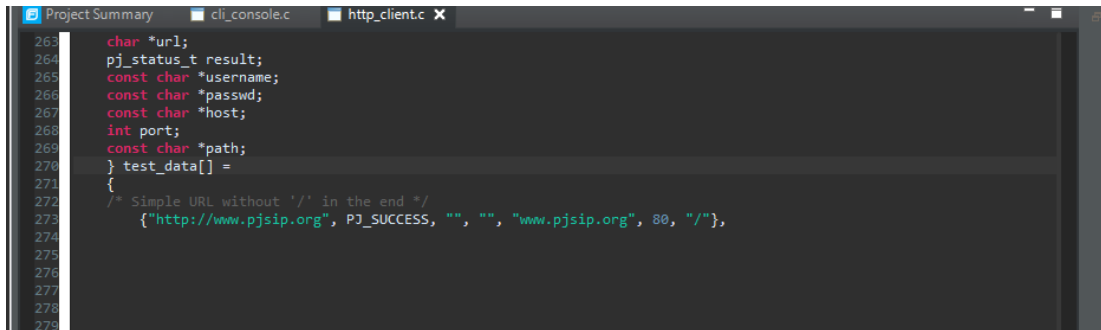
Se sugiere desarrollar políticas de privacidad para las aplicaciones de la organización en donde se indique como el flujo de los datos y como la aplicación hace el uso de estos. De igual forma, como la aplicación debe manejar los datos privados. Debe considerar que las políticas de privacidad internas cumplan con las regulaciones estipuladas de acuerdo con su ubicación y la regulación que se tenga de manera que se cumpla con los requisitos legales. Se considera que la mejor política de privacidad con respecto a los datos confidenciales es aquella que minimiza la exposición de estos. El sistema, procesos y los empleados no deben tener acceso a ningún dato confidencial a menos que sea necesario para cumplir con su función de manera que el acceso a los datos privados se debe restringir al grupo más pequeño posible.

9. ASTERISX - PASSWORD MANAGEMENT: HARDCODED PASSWORD (CWE-259, 798)

Archivo: http_client.c

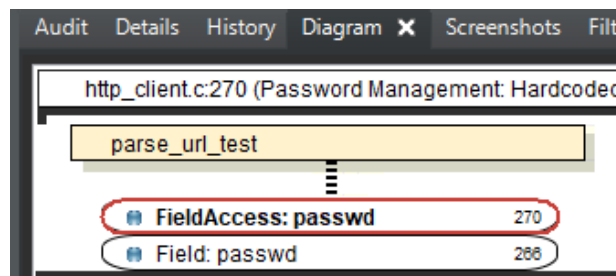
Línea: 270

Código:



```
263 char *url;
264 pj_status_t result;
265 const char *username;
266 const char *passwd;
267 const char *host;
268 int port;
269 const char *path;
270 } test_data[] =
271 {
272     /* Simple URL without '/' in the end */
273     {"http://www.pjsip.org", PJ_SUCCESS, "", "", "www.pjsip.org", 80, "/"},
274
275
276
277
278
279
```

Diagrama:



Explicación:

- El archivo reportado hace parte del conjunto de pruebas automatizadas del proyecto y no es código que se utilice en la ejecución de la aplicación por parte de los usuarios.
- b) La función parse_url_test está destinada para pruebas de desarrollo y contiene datos que no son reales por lo que no existe peligro.

Recomendaciones:

No es necesario el análisis de trazabilidad porque se encontró que la vulnerabilidad es un falso positivo.

10. ASTERISX - WEAK ENCRYPTION: INSECURE INITIALIZATION VECTOR (CWE-329)

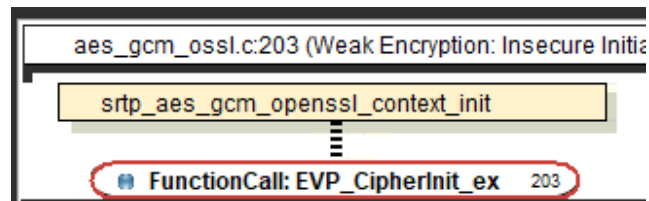
Archivo: aes_gcm_oss.c

Línea: 203

Código:

```
196 break;
197 }
198 }
199 }
200 break;
201 }
202 }
203 if (!EVP_CipherInit_ex(c->ctx, evp, NULL, key, NULL, 0)) {
204     return (srtp_err_status_init_fail);
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
```

Diagrama:



Explicación:

El código utiliza una primitiva criptográfica que usa un vector de inicialización (IV), pero el código no genera IV que sean ampliamente impredecibles o únicos conforme a los requisitos criptográficos esperados para ese primitiva. Por diseño de la aplicación, algunas primitivas criptográficas (como cifrados de bloque) necesitan que los IVs contengan ciertas características para la unicidad y/o imprevisibilidad de un IV. Si estas características no se mantienen, ya sea debido a un error en el programa, la criptografía puede debilitarse o romperse atacando los propios IV.

Si el IV no se inicializa correctamente, los datos cifrados pueden llegar a verse comprometidos y la información puede ser filtrada viéndose comprometida la confidencialidad de los datos. Los vectores de inicialización (IV) deben crearse utilizando un generador de números pseudoaleatorios criptográfico. Al no utilizar un IV aleatorio hace que el texto cifrado sea predecible y susceptible a un ataque de diccionario.

Recomendaciones:

Los diferentes modos de cifrado contienen diversos requisitos para sus IV. Al escoger e implementar un modo, es de suma importancia comprender esos requisitos para tener intacta la seguridad. Usualmente, es más seguro generar un IV aleatorio, ya que será impredecible y tendrá muy pocas posibilidades de no ser único. Los IV no tienen que ser

secretos, por lo que se la generación de IV duplicados se considera una preocupación, se recomienda mantener una lista de IV ya utilizados y verificarla.

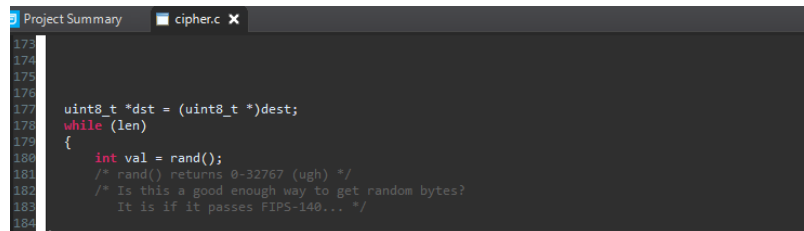
Existen distintas recomendaciones sobre la generación de IV para los modos que se han aprobado.

11. ASTERISX - INSECURE RANDOMNESS (CWE-338)

Archivo: **cipher.c**

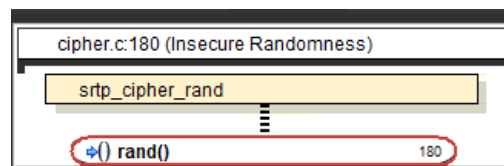
Línea: **180**

Código:



```
173
174
175
176
177 uint8_t *dst = (uint8_t *)dest;
178 while (len)
179 {
180     int val = rand();
181     /* rand() returns 0-32767 (ugh) */
182     /* Is this a good enough way to get random bytes?
183        It is if it passes FIPS-140... */
184
```

Diagrama:



Explicación:

- Se está usando un `rand()` sin la definición de un `RAND_MAX` el valor estará al menos entre 0 y 32767.
- La aplicación de operador `"& 0xff"` enmascara la variable, por lo que deja solo el valor en los últimos 8 bits e ignora el resto de los bits.

Conclusión:

La naturaleza de los números pseudo aleatorias va ligada a la arquitectura del hardware donde se ejecuta dicha función y de la forma como se implementa la función, de ahí que teóricamente existe la posibilidad de predecir el número que se generará, aunque se está enmascarando el valor generado esto realmente no ofrece ningún tipo de cambio sobre el problema.

12. ASTERISX - NULL DEFERENCE (CWE-476)

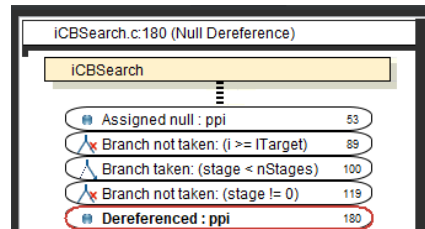
Archivo: iCBSearch.c

Línea: 180

Código:

```
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

Diagrama:



Explicación:

- En la línea 180 se está haciendo uso de los apuntadores ppi, ppe, ppo.
- En la línea 53, se hace la declaración de los apuntadores.

```
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

Recomendaciones:

“Exploitable” puesto que, si no se cumple el flujo para que la condición se puede generar un bloqueo por la asignación nula de los apuntadores, dado aunque hay un flujo alterno que puede hacer que los apuntadores lleguen a este punto nulos no se está haciendo ningún tipo de validación previa a su operación.

13. ASTERISX - USE AFTER FREE (CWE-416)

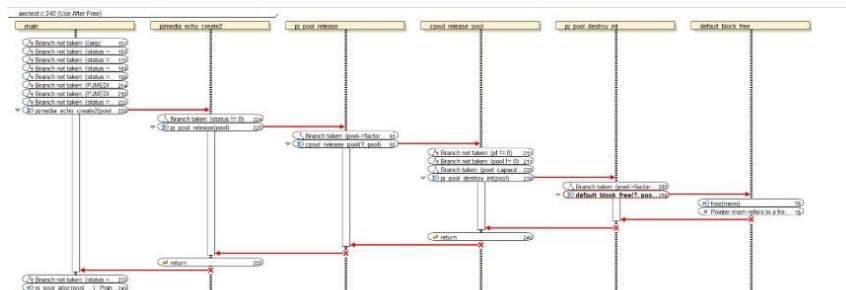
Archivo: aectest.c

Línea: 240

Código:

```
233 if (status != PJ_SUCCESS) {
234     asp_errror(THIS_FILE, "Error creating EC", status);
235     return 1;
236 }
237
238 /* Protecting loop */
239 play_frame.buf = pj_pool_alloc(pool, PJMEDIA_PIA_SPF(&wav_play->info) << 1);
240 rec_frame.buf = pj_pool_alloc(pool, PJMEDIA_PIA_SPF(&wav_rec->info) << 1);
241 pj_get_timestamp(&t0);
242 for (i=0; i < repeat; ++i) {
243
244     if (status != PJ_SUCCESS)
245         break;
246
247     status = pjmedia_echo_playback(ec, (short*)play_frame.buf);
248
249     rec_frame.size = PJMEDIA_PIA_SPF(&wav_rec->info) << 1;
250     status = pjmedia_port_get_frame(wav_rec, &rec_frame);
251     if (status != PJ_SUCCESS)
252         break;
253 }
```

Diagrama:



Explicación:

- Se invoca pjmedia_echo_create2, con pool
- Si la condición se cumple en la línea 225 el echo_common.c, se invoca pj_pool_release
- Como se puede verificar lo que se tiene en este código es un patrón de fábrica, donde dependiendo de la política establecida se vuelve a iniciar en buffer
- Se debe tener en cuenta la configuración para el pi_caching_pool

Recomendaciones:

“Not an issue” Una fábrica no solo proporciona funciones de interfaz genéricas para crear y lanzar grupos, sino que también proporciona una estrategia para administrar la vida útil de estos grupos. Se puede configurar una implementación de muestra, pj_caching_pool, para mantener los grupos liberados por la aplicación para uso futuro siempre que la memoria total esté por debajo del límite

14. ASTERISX - FORMAT STRING (CWE-364)

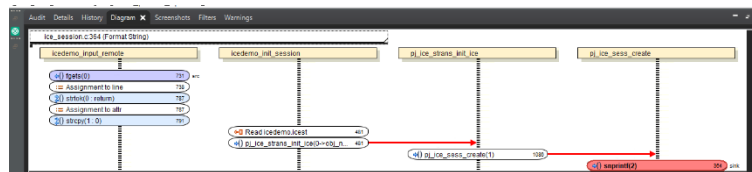
Archivo: ice_session.c

Línea: 364

Código:

```
357 ice->tie_breaker.u32.lo = pj_rand();
358
359
360 pj_timer_entry_init(&ice->timer, TIMER_NONE, (void*)ice, &on_timer);
361
362 pj_ansi_snprintf(ice->obj_name, sizeof(ice->obj_name),
363                 name, ice);
364
365 if (grp_lock) {
366     ice->grp_lock = grp_lock;
367 }
368
```

Diagrama:



Explicación:

Las cadenas de formato deben construirse dinámicamente, definiendo un conjunto de cadenas de formato validas, por lo cual siempre se debe de verificar que el número de directivas de formato en la cadena de formato seleccionada corresponda al número de argumentos a formatear.

- El buffer de salida se asigna con el tamaño del puntero, lo que no permitirá asignarlo con el tamaño requerido.

Recomendaciones:

Un atacante puede controlar el argumento de cadena de formato snprintf(), permitiendo un ataque muy parecido a un desbordamiento de buffer.

15. THE BODGEIT STORE - CROSS-SITE SCRIPTING: PERSISTENT (CWE-79, 80)

The BodgeIt Store			
We bodge it, so you dont have to!			
Home	About Us	Contact Us	Login
Guest user		Your Basket	Search
Our Best Deals!			
Doodahs			
Gizmos			
Thingamajigs			
Thingies			
Whatchamacallits			
Whatsits			
Widoets			
	Product	Type	Price
GZ FZ8	Gizmos	Doodahs	L1.00
Bonzo doo doo dah	Thingamajigs	Thingies	L2.45
ICJ AAA	Thingies	Whatchamacallits	L0.99
GZ KZZ	Thingies	Whatsits	L3.05
Thingie 5	Thingies	Widoets	L3.70
Youknowwhat	Whatchamacallits		L4.32
Youknowwhat	Whatchamacallits		L4.32
Thingie 4	Thingies		L3.50
Doo dah day	Doodahs		L6.50
Thingie 1	Thingies		L3.00

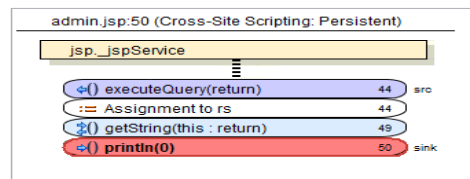
Archivo: admin.jsp

Línea: 50

Código:

```
40 conn.createStatement().execute( UPDATE SCORE SET STATUS = 1 WHERE TASK = HIDDEN_ADMIN );
41 }
42
43 stmt = conn.prepareStatement("SELECT * FROM Users");
44 rs = stmt.executeQuery();
45 out.println("<br/><center><table class='border' width='80%'>");
46 out.println("<tr><th>UserId</th><th>User</th><th>Role</th><th>BasketId</th></tr>");
47 while (rs.next()) {
48     out.println("<tr>");
49     out.println("<td>" + rs.getInt("userid") + "</td><td>" + rs.getString("name") +
50         "</td><td>" + rs.getString("type") + "</td><td>" + rs.getInt("currentbasketid") + "</td>");
51     out.println("</tr>");
52 }
53 out.println("</table></center><br/>");
54
55 stmt = conn.prepareStatement("SELECT * FROM Baskets");
56 rs = stmt.executeQuery();
57 out.println("<br/><center><table class='border' width='80%'>");
58 out.println("<tr><th>BasketId</th><th>UserId</th><th>Date</th></tr>");
59 while (rs.next()) {
60     out.println("<tr>");
61     out.println("<td>" + rs.getInt("basketid") + "</td><td>" + rs.getInt("userid") + "</td><td>" + rs.getString("date") + "</td>");
62     out.println("</tr>");
63 }
64 out.println("</table></center><br/>");
65 }
```

Diagrama:



Explicación:

Las vulnerabilidades de cross-site scripting (XSS) ocurren cuando:

- Los datos ingresan a una aplicación web a través de una fuente que no es de confianza. En el caso de XSS persistente (también conocido como almacenado), la fuente que no es de confianza suele ser una base de datos u otro almacén de datos de back-end, mientras que en el caso de XSS reflejado suele ser una solicitud web. En este caso, los datos ingresan en `executeQuery()` en `admin.jsp` en la línea 44.
- Los datos se incluyen en contenido dinámico que se envía a un usuario web sin validación. En este caso, los datos se envían a `println()` en `admin.jsp` en la línea 50. El contenido malicioso enviado al navegador web a menudo toma la forma de un segmento de JavaScript, pero también puede incluir HTML, Flash o cualquier otro tipo de código que ejecute el navegador. La variedad de ataques basados en XSS es casi ilimitada, pero comúnmente incluyen la transmisión de datos privados como cookies u otra información de sesión al atacante, redirigir a la víctima al contenido web controlado por el atacante o realizar otras operaciones maliciosas en la máquina del usuario bajo el disfraz del sitio vulnerable.

Recomendaciones:

La solución a XSS es garantizar que la validación se produzca en los lugares correctos y se realicen comprobaciones para las propiedades correctas. Debido a que las vulnerabilidades XSS ocurren cuando una aplicación incluye datos maliciosos en su salida, un enfoque lógico es validar los datos inmediatamente antes de que salgan de la aplicación. Sin embargo, debido a que las aplicaciones web a menudo tienen un código complejo e intrincado para

generar contenido dinámico, este método es propenso a errores de omisión (validación faltante). Una forma eficaz de mitigar este riesgo es realizar también la validación de entrada para XSS. Las aplicaciones web deben validar su entrada para evitar otras vulnerabilidades, como la inyección de SQL, por lo que aumentar el mecanismo de validación de entrada existente de una aplicación para incluir comprobaciones de XSS es generalmente relativamente fácil. A pesar de su valor, la validación de entrada para XSS no reemplaza la validación de salida rigurosa. Una aplicación puede aceptar la entrada a través de un almacén de datos compartido u otra fuente confiable, y ese almacén de datos puede aceptar la entrada de una fuente que no realiza la validación de entrada adecuada. Por lo tanto, la aplicación no puede confiar implícitamente en la seguridad de este o cualquier otro dato. Esto significa que la mejor forma de prevenir vulnerabilidades XSS es validar todo lo que ingresa a la aplicación y sale de la aplicación destinada al usuario.

El enfoque más seguro para la validación de XSS es crear una lista permitida de caracteres seguros que pueden aparecer en el contenido HTTP y aceptar entradas compuestas exclusivamente por caracteres en el conjunto aprobado.

16. RACE CONDITION: SINGLETON MEMBER FIELD (CWE-362, 488)

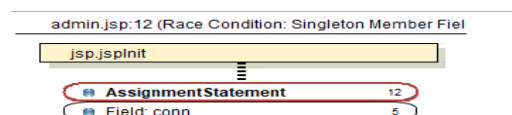
Archivo: admin.jsp

Línea: 12

Código:

```
1  <%@ page import="java.sql.*" %>
2  <%@ page import="java.math.*" %>
3  <%@ page import="java.text.*" %>
4  <%= %!
5      private Connection conn = null;
6
7      public void jspInit() {
8          try {
9              // Get hold of the JDBC driver
10             Class.forName("org.hsqldb.jdbcDriver");
11             // Establish a connection to an in memory db
12             conn = DriverManager.getConnection("jdbc:hsqldb:mem:SQL", "sa", "");
13         } catch (SQLException e) {
14             getServletContext().log("Db error: " + e);
15         } catch (Exception e) {
16             getServletContext().log("System error: " + e);
17         }
18     }
19
20     public void jspDestroy() {
21         try {
22             if (conn != null) {
23                 conn.close();
24             }
25         }
26     }
27 }
```

Diagrama:



Explicación:

Muchos desarrolladores de Servlet no comprenden que un Servlet es un singleton. Solo hay una instancia del Servlet, y esa instancia única se usa y se reutiliza para manejar múltiples

solicitudes que son procesadas simultáneamente por diferentes subprocesos. Un resultado común de este malentendido es que los desarrolladores usan los campos de miembros de Servlet de tal manera que un usuario puede ver inadvertidamente los datos de otro usuario. En otras palabras, almacenar datos de usuario en campos de miembros de Servlet introduce una condición de carrera de acceso a datos.

Recomendaciones:

No use campos de miembros de Servlet para nada más que constantes. (es decir, hacer que todos los campos de miembros sean finales estáticos). Los desarrolladores a menudo se ven tentados a utilizar campos de miembros de Servlet para los datos del usuario cuando necesitan transportar datos de una región de código a otra. Si este es su objetivo, considere declarar una clase separada y usar el Servlet solo para "envolver" esta nueva clase.

17. PUZZLE MALL - PASSWORD MANAGEMENT: HARDCODED PASSWORD



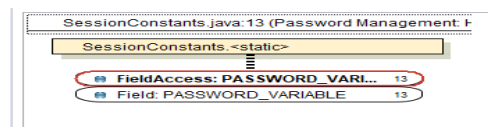
Archivo: **SessionConstants.java**

línea: **13**

Código:

```
9  // SessionConstants.java
10 public final class SessionConstants {
11
12     public static final String USERNAME_VARIABLE = "username";
13     public static final String PASSWORD_VARIABLE = "password";
14     public static final String QUESTION_VARIABLE = "question";
15     public static final String ANSWER_VARIABLE = "answer";
16     public static final String ROLE_VARIABLE = "role";
17     public static final String EMAIL_VARIABLE = "email";
18
19     public static final String PHONE_VARIABLE = "phone";
20     public static final String ADDRESS_VARIABLE = "address";
21     public static final String CREDIT_VARIABLE = "credit";
22
23     public static final String LOGIN_MSG_VARIABLE = "loginmsg";
24     public static final String REGISTRATION_MSG_VARIABLE = "registrationmsg";
25
26     public static final String FLOW_PHASE2_VARIABLE = "phase2";
27     public static final String FLOW_PHASE3_VARIABLE = "phase3";
28
29 } //end of class
```

Diagrama:



Explicación:

Se considera una vulnerabilidad alta donde las contraseñas codificadas pueden comprometer la seguridad del sistema de una manera que no es fácil de solucionar. Nunca

es una buena idea codificar una contraseña. La codificación de una contraseña no solo permite que todos los desarrolladores del proyecto vean la contraseña, sino que también hace que solucionar el problema sea extremadamente difícil. El código que evaluamos se ejecutará correctamente, pero cualquier persona que tenga acceso a él tendrá acceso a la contraseña, un empleado con acceso a esta información puede usarla para ingresar al sistema.

Recomendaciones:

Las contraseñas nunca deben codificarse de forma rígida y, por lo general, deben ofuscarse y administrarse en una fuente externa. El almacenamiento de contraseñas en texto sin formato en cualquier parte del sistema permite que cualquier persona con permisos suficientes lea y pueda hacer un uso indebido de la contraseña. Como mínimo, hash las contraseñas antes de almacenarlas.

CONCLUSIONES

La herramienta HP Fortify nos ayudó a encontrar diferentes vulnerabilidades en diferentes software el cual aprendimos varios errores donde un atacante puede aprovechar esas vulnerabilidades para poder atacar y realizar alguna actividad que ponga en riesgo nuestra información o recursos ,el programador debe conocer bien sobre las funciones que generan estas vulnerabilidades, las soluciones correspondientes, la manera correcta de cifrar la información y no dejar datos en archivos a simple vista como ser contraseñas o accesos al sistema y entre otros problemas. El programa tiene la capacidad de manera automática el procesamiento de las aplicaciones para permitir al programador o usuario concentrarse en los riesgos según su nivel de prioridad ya sea crítico, alto, moderado y como lo dice el (Schmitt, s.f.) Gerente general de HP Security Fortity “la tecnología de análisis de detección de HP Fortify está revolucionando los enfoques tradicionales de la seguridad de las aplicaciones al implementar el aprendizaje automático para priorizar automáticamente los problemas que importan y quitar el ruido, lo que mejora radicalmente los resultados y el esfuerzo necesarios para proteger las aplicaciones sensibles” con ello aprendemos que vulnerabilidades son las más prioritarias y así como auditores tratar dicha vulnerabilidad para evitar riesgos en el futuro.

REFERENCIAS

- Insecure Transportation Security Protocol Supported (SSLv3) | Netsparker. (2021). Retrieved 28 December 2021, from <https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/insecure-transportation-security-protocol-supported-ssl3/>
- Best practices to avoid security vulnerabilities in your iOS App | Humble Bits. (2021). Retrieved 28 December 2021, from <http://blogs.quovantis.com/best-practices-to-avoid-security-vulnerabilities-in-your-ios-app/>
- (2021). Retrieved 28 December 2021, from http://webdiis.unizar.es/~ricardo/files/slides/invitedTalks/slides_BetaBeers-13.pdf
- Buffer overflow: así funciona esta gran fuente de vulnerabilidades. (2021). Retrieved 28 December 2021, from <https://www.redeszone.net/tutoriales/seguridad/fallo-buffer-overflow-desbordamiento- buffer-que-es/>
- Center, S., Definitions, I., & manipulation, F. (2021). File path manipulation. Retrieved 28 December 2021, from https://portswigger.net/kb/issues/00100b00_file-path-manipulation
- Privacy Violation | OWASP Foundation. (2021). Retrieved 28 December 2021, from https://owasp.org/www-community/vulnerabilities/Privacy_Violation
- CWE - CWE-1204: Generation of Weak Initialization Vector (IV) (4.6). (2021). Retrieved 28 December 2021, from <https://cwe.mitre.org/data/definitions/1204.html>
- Schmitt, J. (s.f.). Obtenido de <https://www.itsitio.com/cl/hp-fortify-incorpora-aprendizaje-automatico-2/>