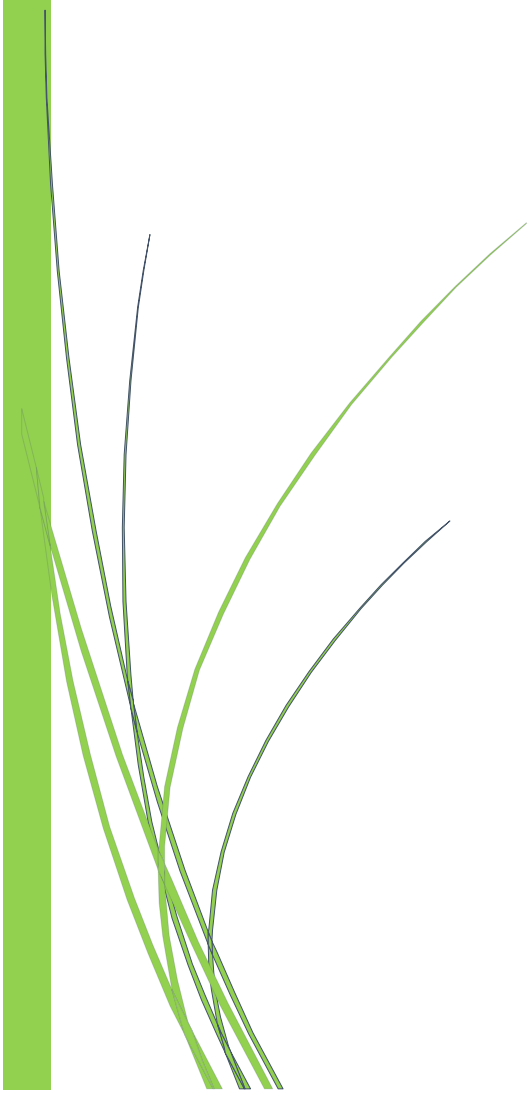
Análisis de Vulnerabilidades

Funciones vulnerables en C.



Rodríguez Gallardo Pedro Alejandro

En el lenguaje de programación C existen varias funciones vulnerables a Buffer Overflow, a continuación veremos varias de estas a la par veremos las funciones mejoradas de las mismas cuya función es mitigar esta vulnerabilidad.

gets()	**fgets()**
gets() no comprueba la longitud del buffer y esto siempre resulta en una vulnerabilidad. char * gets (char * str); • str :Apuntador del bloque de memoria (arreglo de caracteres)	char *fgets(char *str, int n, FILE *stream) • str : Apuntador de un arreglo de caracteres • n : Máximo número de caracteres que se van a copiar en str • *stream : Apuntador de un FILE object que identifica una entrada trasmitida . • stdin puede ser usado como argumento para leer de la estrada estándar.
strcpy()	**strncpy()**
strcpy() No comprueba la longitud del buffer y se podría sobrescribir zonas de memoria contiguas. char *strcpy(char *dest, const char *src) • src: La cadena que se va a copiar • dest: El apuntador del arreglo donde se va a copiar.	strncpy() combate esto pidiendo la longitud del buffer. char *strncpy(char *dest, const char *src, size_t n) • src: La cadena que se va a copiar • dest: El apuntador del arreglo donde se va a copiar. • n: Los primeros n caracteres que va a ser copiados de src a dest.
sprintf()	**snprintf()**
sprintf() No comprueba la longitud de buffer, esto puede producir desbordamiento. int sprintf(char *str, const char *string,...); • *str Buffer donde se va a almacenar la cadena • string Cadena que se va a pasar al buffer.	snprintf() Tiene la doble ventaja de prevenir los desbordamientos de buffer y devolver el tamaño buffer mínimo necesario. int snprintf(char *str, size_t size, const char *format, ...); • *str Es el buffer donde el la salida de printf va a ser colocado. • size Es el numero máximo de caracteres que van a ser escritos en el buffer. • format formatos como "%d", etc.
printf, fprintf, sprintf y snprintf	
Todas las funciones anteriores son factibles a los ataques de formato de cadena, los cuales pueden causar fugas de información, reescribir la memoria.	Para mitigar este ataque siempre se debe codificar el formato de cadena. Nunca se debe dejes venir directamente de la entrada de cualquier usuario.

Referencias:

- https://www.owasp.org/index.php/Reviewing_Code_for_Buffer_Overruns_and_Overflows
- https://dhavalkapil.com/blogs/Buffer-Overflow-Exploit/
- https://www.geeksforgeeks.org