Cómo disminuir las fallas en la construcción de *software*

Foto: Natalia Fernanda Madrid Vidales

En el curso de Programación Segura del 2016 participaron 36 estudiantes de maestría en Ingeniería de Sistemas. Durante 45 horas y en grupos de tres, experimentaron el desarrollo de *software* seguro mediante la construcción de prototipos de una aplicación bancaria y de una red social. En el taller les iban introduciendo vulnerabilidades y sometiéndolas a prueba para intentar corregirlas.

En el curso Programación Segura, el profesor Martín Ochoa recalca que aunque es casi imposible construir *software* 100 % seguro, se puede reducir sustancialmente la posibilidad de que lo vulneren. Para ello, es fundamental que los ingenieros tengan conciencia de que los atacantes son impredecibles y pueden interactuar con el sistema en formas inesperadas para lograr sus objetivos. En esta entrevista habla de principios que ayudan a evitar los errores.

¿Cuáles son las causas de los errores en programación?

Los errores se cometen en el diseño, en la implementación o en el mantenimiento del *software*. Son costosos de reparar y pueden acarrear graves consecuencias. En el curso Programación Segura, en la Escuela Internacional de Verano 2016, enfatizamos en la falta de conciencia, pues a menudo los desarrolladores están presionados para enfocarse en la funcionalidad del sistema, en que pueda usarse, y no se les ocurren esos casos límite de los que el atacante sí estará pendiente. También hay factores económicos, pues desarrollar *software* más seguro implica mayores costos porque demanda más tiempo, personal más calificado, más pruebas. Discutimos hasta qué punto ese costo se justifica porque un ataque es muy grave, pero la probabilidad de que ocurra es difícil de cuantificar dado que quien lo perpetra es inteligente, puede tener muchas motivaciones y actuar de forma impredecible. En seguridad informática las respuestas están en investigación, aún no se sabe lo suficiente. Nuestra hipótesis es que al aumentar la conciencia, al educar a los profesionales en los tipos de errores más frecuentes y en la mentalidad que deberían tener al desarrollar *software* seguro, se elevará la calidad y se prevendrán ciertos ataques.

En la presentación del curso de Programación Segura, ustedes hablan de ataques "espectaculares". ¿Podría darnos algún ejemplo?

Son innumerables. En el 2014 se publicó un fallo de seguridad llamado *heartbleed*,

que se aprovecha de una vulnerabilidad del *software* gratuito Open SSL, usado por millones de servidores de internet. Esta permite robar datos sensibles de los servidores y se desconoce su impacto real porque cuando la dieron a conocer ya la habían arreglado en la mayoría de servidores. Otro caso es el robo de la base de datos de los más de 70 millones de usuarios de Play Station, de Sony, en el 2011. No solo sustrajeron los números de las tarjetas de crédito, sino información personal de la gente y se cree que ocurrió por un fallo en el diseño del *software* y en el mantenimiento de los sistemas. Son especulaciones porque los detalles de muchos de estos incidentes se mantienen en secreto.

Foto: Natalia Fernanda Madrid Vidales

Martín Ochoa es profesor asistente de la Singapore University of Technology and Design (SUTD), donde investiga en temas de seguridad de software aplicados a infraestructuras críticas y a internet de las cosas. Es ingeniero de sistemas de la Universidad Latina de Costa Rica, con pregrado y maestría en Matemáticas de las Universidades LMU Múnich (Alemania) y La Sapienza (Italia) y Ph.D en Ciencias de la Computación de la Universidad Tecnológica (TU) de Dortmund (Alemania). Antes de vincularse a SUTD, fue investigador posdoctoral en la Technische Universität München e investigador y consultor en seguridad para Siemens en Múnich.

¿Cuáles son los sectores más afectados por fallas de seguridad?

Las infraestructuras críticas como la banca, los sistemas de distribución y tratamiento de agua y de electricidad, las refinerías de petróleo, las centrales nucleares y las telefónicas. Son servicios cada vez más computarizados, que corren algún tipo de *software*, potencialmente vulnerables. Aparte de que con frecuencia la prioridad es la funcionalidad, muchos se desarrollaron con un modelo cerrado porque no estaban expuestos a internet como sí ocurre hoy cuando cada vez más tienden a interconectarse. Así, surgen nuevos vectores de ataque, incluso remotos. A veces es un problema de diseño; otras se relaciona con la complejidad del *software* actual (un sistema se basa en otro y este a su vez en un tercero); como son muchas composiciones heterogéneas es muy difícil garantizar que al juntarse funcionen correctamente.

Tras escuchar a los expertos, uno queda con la sensación de que es inevitable que lo roben. ¿Puede hablarse de unos estándares mínimos de seguridad?

Hay mucha paranoia, aunque es un campo tan nuevo que todavía las nociones de riesgo, de probabilidad de un ataque y de su impacto son muy instintivas. La mayoría de las veces los sistemas se comportan bien. Mientras exista el atacante, debemos estar alerta para mejorar las funciones de seguridad, de modo que los episodios sean raros. Es un trabajo en progreso, no va a ser 100 % seguro, pero se puede aumentar la confianza en los sistemas mediante pruebas y verificaciones. ■

Principios del desarrollo de *software* seguro

Durante el curso de la Escuela Internacional de Verano 2016 los estudiantes analizan cerca de 10 principios abstractos y técnicos. Algunos de ellos son:

- **Sea desconfiado**. Es un error confiar en que los *inputs* del usuario serán los que espera el desarrollador, pues en realidad tiene libertad de hacer lo que quiera con el sistema.
- **Defensa en profundidad.** Es equivocado pensar que el atacante nunca podrá acceder a ciertas máquinas. Implica tener una última línea de defensa para prevenir un desastre, pues asume que los controles fallaron y alguien logró entrar al sistema. Por ejemplo, sabiendo que los usuarios suelen emplear la misma contraseña para varios servicios, que estas a menudo son débiles y que se pueden robar la base de datos, una buena práctica es grabarla como un *hash* o código asociado y no en texto plano. Así es más difícil de recuperar.
- **No inventar su propia seguridad por obscuridad** (consiste en hacer las cosas más complicadas para prevenir ataques)**.** El error está en que a menudo el desarrollador no tiene capacidad de idear algo realmente bueno, razón por la cual es mejor que use lo que ya existe y funciona. Para ello debe estar en contacto con la comunidad y mantenerse actualizado.