acmqueue

Lessons from the Letter

Security flaws in a large organization.

Dear Readers,

I recently received a letter in which a company notified me that they had exposed some of my personal information. While it is now quite common for personal data to be stolen, this letter amazed me because of how well it pointed out two major flaws in the systems of the company that lost the data. I am going to insert three illuminating paragraphs here and then discuss what they actually can teach us.

“The self-described hackers wrote software code to randomly generate numbers that mimicked serial numbers of the AT&T SIM card for iPad—called the integrated circuit card identification (ICC-ID)—and repeatedly queried an AT&T web address.”

This paragraph literally stunned me, and then I burst out laughing. Let’s face it, we all know that it’s better to laugh than to cry. Unless these “self-described hackers” were using a botnet to attack the Web page, they were probably coming from one or a small number of IP addresses. Who, in this day and age, does not rate limit requests to their Web sites based on source IP addresses? Well, clearly we know one company that doesn’t. It’s very simple: if you expose an API—and a URL is an API when you’re dealing with the Web—then someone is going to call that API, and that someone can be anywhere in the world.

A large company doing this is basically begging to be abused: it’s not like you’re just leaving your door unlocked, it’s like a bank letting you try 1 million times to guess your PIN at the ATM. Given enough time—and computers have a lot of time on their hands—you’re going to guess correctly eventually. That’s why ATMs DON’T LET YOU GUESS a million PINs! All right, in this case the company was not going to lose money directly, but it certainly lost a good deal of credibility with its customers and, more importantly, possible future customers. Sometimes brand damage can be far worse than direct financial damage.

Now we come to the next paragraph, in which the company admits to not having proper controls over its own systems:

“Within hours, AT&T disabled the mechanism that automatically populated the email address. Now, the authentication page log-in screen requires the user to enter both their email address and their password.”

“Within hours?!” Are you serious? At this point I was laughing so hard it hurt, and my other half was wondering what was wrong with me, since I rarely laugh when reading the mail. The lesson of this paragraph is to always have the ability to kill any service that you run, and to be able to either roll forward or roll back quickly. In fact, this is the argument made by many Web 2.0, and 1.0, and even 0.1 proponents: that, unlike packaged software, which has release cycles measured in weeks and months, the Web allows a company to roll out changes in an instant. In geological time, hours might be an instant, but when someone is abusing your systems, hours are a long time—long enough, it seems, to acquire several hundred thousand e-mail addresses.

Finally, in the next paragraph we find that someone at AT&T actually understands the risk to its customers:

“While the attack was limited to email address and ICC-ID data, we encourage you to be alert to scams that could attempt to use this information to obtain other data or send you unwanted mail. You can learn more about phishing at www.att.com/safety.”

I somehow picture a beleaguered security wonk having to explain, using very small words, to overpaid directors and vice presidents just what risk the company has exposed its users to. Most people now think, “E-mail address, big deal, dime a dozen,” but of course phishing people based on something you know about them, like their new toy’s hardware ID, is one of the most common form of scams.

So, some simple lessons: rate limit your Web APIs, have kill switches in place to prevent abuse, have the ability to roll out changes quickly, and remember to hire honest people who can think like the bad guys, because they are the ones who understand the risks.

One other thing is for sure, this letter’s a keeper.

KV

KODE VICIOUS, known to mere mortals as George V. Neville-Neil, works on networking and operating system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor’s degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. He is an avid bicyclist and traveler who currently lives in New York City.

© 2010 ACM 1542-7730/10/0700 \$10.00