"Security Through Obscurity" Ain't What They Think It Is

By Jay Beale, Lead Developer, Bastille Linux Project (jay@bastille-linux.org), , Principal Consultant JJB Security Consulting and Training

The most misunderstood statement in the computer security field has got to be "security through obscurity is bad." As security professionals, many of us try to teach a few simple lessons to help system administrators become more security-conscious. Unfortunately, the simplicity of our lessons has backfired. In this article, I'll talk about how obscurity can aid security -- hopefully, I can clear up some of this confusion.

Obscurity: What Do We Really Mean?

First, what does the security professional mean by bad "security through obscurity?" We really mean "security implemented solely through obscurity." This describes the state where your entire method of security resides in hoping that the attacker doesn't know something about the setup of your network, computer or program. One simple case is where you put your company's secrets on an internal webserver, with no password-protection on the pages. Instead of relying on passwords or another acceptable method of access control, you're relying on something different. You're assuming that no one will know about that webserver except for the internal company employees who you've told. This almost seems like a decent assumption, except that anyone with some network discovery tools (like cheops, firewalk, snmpwalk and nmap) can find just about every webserver on your network. See, the problem is that you've used the obscurity of the data's location as your **sole** method of access control. This just doesn't work. While some attackers will never expend the effort to find that webserver, many others can and will. Those will obtain access to information that you wanted to keep them away from.

Let's continue with our example. Suppose you set that same sensitive webserver to run on a non-standard port, such that it runs on port 8000, instead of the usual port 80. You're figuring "Jay's not going to find this webserver -- he'll never think to check port 8000!" Well, I'd accuse you of missing the point! You're still depending on obscurity as the only means of access control. A clueful attacker, or at least one running nessus, will see right through this. In the example of nessus[1], the scanner looks at the first part of the communication it makes with a target and identifies the server running, regardless of port location. So, again, many of the attackers will find this webserver. We haven't yet performed decent access control.

Obscurity just does not work as a method of access control. Someone will find most anything by querying your peer machines, your routers and the target machines themselves. They'll possibly run all manner of active scanner or passive sniffer. They'll even launch "social engineering" attacks, where they trick your users into giving them the information they need, usually by impersonating someone who should normally have access to that information. Since an attacker can often obtain or guess the information that we're obscuring, especially if it is as simple as the listening port of our webserver process, we not implementing decent security. This is what we mean by "security implemented solely through obscurity is bad."

Security Through Obscurity Isn't Bad?

Obscurity isn't always bad. We're usually just talking about how strong the obscurity is and how easily it can be defeated. The original idea "security implemented solely through obscurity is bad" came from criticism of certain cryptosystems. Cryptographers found that some poor cryptosystems had "obscurity of the crypto algorithm" as their only defense -- once an attacking cryptographer could reverse-engineer the algorithm, he could decode all messages made in that algorithm. Let's look at an example.

In the Caesar cipher, every message is encoded simply by changing each letter of the original cleartext into

the third letter after it in the alphabet. So, an A becomes a D, B becomes an E and so on. The word "security" becomes "vhfxulwb." Now, as soon as you know the algorithm, you can reverse it, right? Just replace every letter with the letter of the alphabet that precedes it by 3. This is clearly a really weak system once you know the algorithm. The security of the system was only in the secrecy, or obscurity, of the algorithm.

To take this slightly farther, you can think of the variant of this algorithm where we change that "3" number of a mutually-agreed secret number. Unfortunately, this still doesn't buy you much, because the algorithm is too weak. It doesn't stand up to a "brute force" attack, where we try every key and look for English words. See, that secret can only vary from 1 to 25, assuming a strict alphabetic system, or 1 to 255, assuming a 8-bit byte system. In either case, I can generate all the combinations pretty quickly on a computer, allowing me to quickly find your hidden message.

On the other hand, if I use a strong (not so reversible) algorithm with a nice large keyspace (40 bit encryption has over 1 trillion possibly keys), it's not so easy to eavesdrop on my encrypted communications. Good cryptosystems work well, without their users being in constant fear that everyone will be able to decrypt sensitive material. Why are we talking about crypto? We're trying to show is where this whole idea came from, so you can understand why we're always harping on obscurity.

With this in mind, look at all this brouhaha from a different angle. See, some obscurity is necessary in most commonly accepted authentication systems. When I type my Unix password, I'm counting on a secret element: a key! If you pull all of the obscurity out, everyone knows my password. We can all login. Heck, everyone knows my RSA private key. Everyone has the ability to decrypt my PGP/GPG-encrypted mail and sign e-mail as me. Clearly, we need and have accepted secrecy-based systems. Taking this back to crypto, the main point is this: we obscure the key, but not the algorithm.

OK, so obscurity isn't all bad -- in many situations, it's altogether necessary. To look at where it might be not so necessary, but still beneficial, let's get back to the webserver example.

Is Obscurity Ever Good?

Now, suppose you put a good password, or some method of strong authentication, on your company secrets website. You even make sure to put the site on an SSL server. Heck, perhaps you even go the extra mile and authenticate people with client-side certificates. Let's suppose that you've got fairly reasonable protection for your information. Now we've got decent access control.

At this point, is there any harm to hiding the name of the machine and the port number the server is running on? Really, stop and think about this.

Does it hurt your site security at all? No, it really doesn't. Your good access control, in the form of strong authentication, is still present. All we've done is made the server slightly harder to find! See, so long as you understand that the server location and port number can't serve as a method of authentication, you haven't harmed your security in the slightest.

Obscurity: Avoid/Block Some Number of Attacks

Well, if you've done no harm, is it worth the minor amount of effort this took? What have we accomplished? Well, for starters, we've blocked some number of attacks. As we hinted at earlier, some number of script kiddies won't find our webserver now. Remember, script kiddies are often very new at this and uninformed[2]. Further, they're often using low-grade simple tools. For example, if your average script kiddie has an exploit against IIS version 4.1, he'll often have a simple exploit-specific scanner which scans large sections of the Internet looking for IIS version 4.1. This simple scanner often has an algorithm like this:

1. Check if port 80 is open on each target machine.
2. Connect to port 80 on the target if appropriate and check the version string.
3. Print out the IP address if this is a vulnerable web server.

If we're running our webserver on a port other than 80, we avoid his scan. Pretty nice, huh? Well, this doesn't work on all attackers. Remember, our attackers aren't all using low-grade tools. He might have gotten lucky and gotten a good scanner that checks multiple ports, or even checks every port. He might also be trying a trivial "machine gun" approach, firing his exploit at every open port without checking for a webserver, vulnerable or no. He might even be a non-script-kiddie hacker, in which case he can write his tools to do this the right way. In any case, what we're trying to get at here is that obscurity can help.

Obscurity can help block some number of attacks, like those from script kiddies with common low-grade tools or attackers who are too lazy to check all the ports on a machine. Note: we're not using obscurity for authentication -- we're just using it to aid an existing good authentication system. So, is this all?

Obscurity: Force An Attacker to Be Less Stealthy

Let's think about what we'd see if we were monitoring the webserver's network connection. In the case where we put our webserver on the common port, port 80, our attacker's information probe "what version of IIS are you running" looks (from our firewall/router logs) like every other web request coming into the machine. Our attacker knows this and is quite happy, because he has little chance of being observed while doing his information probe. Well, what about the case where we put our webserver on port 253?

If we put our webserver on port 253, our attacker generally has got to scan a whole lot of ports on the target webserver to find it. Instead of making a quiet query against port 80 on the target, he'll have to throw at least one packet at many ports on the target to find open ones, then make connections to each of the open ones until he finds the one running the webserver[3]. His attack just got a whole lot "louder." Many portscan detectors, including Bastille Linux developer Mike Rash's upcoming Port Scan Attack Detector, watch for a client to throw packets at our machine on a number of different ports. -- when they see enough different port destinations from a single machine in a set period of time, they alert. See, we've got a much better shot at noticing the scan (or the attack) if the attacker has to go through some pain to find his target.

So, there's another advantage of using obscurity. By obscuring something about our environment, we force an attacker to possibly go through more effort to learn that information before he can execute his attack. This gives us a better chance of observing him! Now, remember, don't get too cocky. He can still find our server. It might not even take tons of extra effort, if he guesses well. We might still not observe him, if we have no procedures or technology set up to do the observing! But we have gained something. But what else did we gain from this little bit of obscurity?

Obscurity: Potentially Slows Down the Attacker

Remember how, in the past example, our attacker had to scan a lot more ports to find out which one is open? He also had to connect to each open port until he found the webserver? Well, if you've ever run a port scanner, you'll notice that significantly slows things down. This is especially advantageous when the attacker is scanning every machine in your domain, as he has to generate a whole lot more queries and wait for a whole lot more responses, just to find his server. To get a ballpark-math feeling for this, think about a domain of 200 machines. Normally, he'd just make about 50-200 queries to find his webserver, with a worst case of 200 queries. Supposing that he's scanning 1000 ports now, looking for open ports on each machine, that's 50,000 - 200,000 queries, with a worst case of 200,000. Actually, it's worse than that! He might be really

thorough and check all 65,535 possible ports, for a worst case of around 13 million queries! Now, that's pretty darned good for slowing him down and making his target scan pretty loud.

Well, here we see how obscurity can really help slow an attacker down. We've seen how additional obscurity can repel many of the inexperienced attackers, especially those with low-grade tools. Further, we've examined how a little obscurity can really force an attacker to be a lot less sneaky. All of this combines for some real advantages. Please, before dismissing a measure simply because it employs or increases obscurity, stop to examine the real effects in common scenarios against different classes of attacker. You'll find there are many situations where a little obscurity can definitely help out.

Jay Beale is the Lead Developer of the Bastille Linux Project, which creates a wildly popular security tightening program. Jay is lucky enough to b have his work on Bastille sponsored by MandrakeSoft. Jay is the author of a number of articles on Unix/Linux security, along with the upcoming book "Securing Linux the Bastille Way," to be published by Addison Wesley. You can learn more about his articles, talks and favorite security links via http://www.bastille-linux.org/jay.

1I'm not trying to pick on nessus. That tool just happens to get the attacks right more often than other scanning tools. Honestly, we could make a simple perl script to do this slightly-more-intelligent scanning for services. We'd just run nmap (as nessus does) then follow up with short connections to each of the open ports found.

2See my past article Why Do I Have to Harden My System, paying special attention to "The Way of the Script Kiddie."

3Yes, he can scan each port and immediately open a connection to the ones that are open. He's still creating a whole lot more variant network traffic than he was before.