Principals
Table of Contents - 4Q 2016
Updated - 3Q 2020
Developer: BJ Bellamy

Internal vrs External

Internal **is** external. Its a Zen thing.

A rule of thumb is that 80% or the risk to a network comes from inside the network. It is more our nature to think that because there are thousands of attackers *out on the Internet* and only a few people, all of whom we trust, inside our network the majority of the risk would be external.

However, you have to consider the malicious insider, disgruntled employees, friends and relatives that employees bring into the network, vendors who have an internal presence, and of course authorized insiders who become compromised through spear-phishing and similar attacks. In this last case, the actual attacker is on the outside, but they are operating from the inside.

Critical Systems

Customers classify their systems and resources based on what they consider valuable. This is not always compatible with security issues. A security tech needs to review the network and its components to determine what the actual likely risks are.

"Our XYZ site gets the most hits.", "That app does not have any PII." . We hear all sorts of reasons why some resource is or is not at risk. But there is often a difference between what a client believes is valuable, and what an attacker thinks is valuable (which is based on their intent).

There are several misconceptions about what attackers are actually after that began in the media and has been incorporated into our Enterprise mentality.

SSNs, passwords, emails... These are the types of digital assists that most people assume is an attacker's target. And yes that can be the case. But there are several other digital assents that are more often the goal of intrusion.

If we are going to protect the client's digital assists, we need to know which are higher value targets

from the attacker's perspective.

As a rule of thumb, all systems must be considered critical. A given host may not house what most would consider important material, but:

1. every host is a stepping stone to all other hosts
2. the network itself is a stepping stone to other networks which might be the actual target
3. every host offers processing horse power
4. every host offers disk space
5. every host can be used to provide anonymous or covert communications
6. every host can be used as a distraction from the real attack

Another way to state this principal is that "you often do not know what an attacker wants from you."

Organizations are quick to protect what is important to them, and that makes sense. But they fail to then protect resources that are perceived less valuable on the assumption they are less value to attackers also.

For example, we provide less protection for a print server than a database server. However, an attacker may be after processing horsepower rather than our data. In that case, the print server is a better target since its CPU sits idle most of the time, and high CPU utilization is less likely to be noticed on a print server than a database server. And of course, compromising a printer gives an attacker a inside platform from which to attack further.

Read the "Attacker's Motivations (Doc)" paper, listed on the TOC page, for a more in-depth discussion on this topic.

Work Smart, Not Hard

Attackers, along with most people, would rather expend less energy and time on a problem if possible. There are usually many ways to solve a given problem, and there is an art to quickly finding the solution that takes the least amount of time and energy. There is trade-off in determining the "easiest" solution, so balancing the time and effort it takes to determine the fastest and easiest solution helps determine the optimal solution. In other words, you do not want to spend an hour determining which 1 minute solution is best.

You want to approximate your ROI for a given problem. And in the approximating you emphasize the ease of the solution over duration, number of steps, resources, and even the time required. At some point, the cost of protecting a component become cost prohibitive and the risk needs to be transferred to insurance or some other party.

But be clear that there is no laziness involved. This is about efficiency and productivity.

Follow the path of least resistance. Consider these two scenarios. Given that you want to attack a organization's database in some way.

You can choose to attack the host directly. Attempt to identify the software make and model, and then craft a buffer overflow for your attack payload which will open a backdoor through which you can run SQL queries on the actual data.

You can also choose to attack the host indirectly. First compromise the CISO's assistant through a weak password compromise. Then using their credentials to query the data as a trusted insider.

Both approaches could work. But while the first involves several steps with varying degrees of difficulty, the second is less complicated and faster.

You Do Not Know How What You Show Can Be Used Against You

You are a developer, and you have a name. (search discussion forums, product forums, google for the name and agency/KY/Frankfort/...)

An error message can disclose the name and version numbers of a product. Google for the default credentials and for known vulnerabilities and you will be surprised how much damage can be done based on what appears as benign information such as an error message.

The name of an actual executable might be disclosed. (google for the developer's manual and find out what parameters you can pass to the app)

Knowing A Technique/Tool Exists...

The best way to become familiar with a computer program, in this case nikto, is to first read the man page to get a feel for what the program does and what functions it provides. Then find examples and reference and inspect the parameters used by referring to the man page to identify the details of each parameter and the alternatives. This will give you a clear idea of what is possible. You do not need to memorize all the parameters and possibilities, you simply need to know that a thing can be done. Knowing something can be done, you would review the syntax page and man page for the details. If you know a thing can be done, at least you will have an idea of which programs could be helpful leaving simply to refer to the documentation for the details.

Attackers Are Patient

While attackers typically look for the easiest way to accomplish an attack, along with simply poking around and exploring in hopes of coming across something they can take advantage of, they are also very patient if they are motivated. Finding a small vulnerability will drive an attacker to spend much more time and effort examining a target knowing that if one vulnerability can be found, it is very likely that many more vulnerabilities probably exist and can be leveraged.

What You Think Is Impossible Is Done Daily

There are many things that we know cannot be done. There are plenty of processes and possibilities that we can image, but that technology does not allow for.

Still, we have to keep an open mind and assume that what we know to be impossible, may in fact be possible.

For years people believed that a switched network eliminated the possibility of one host sniffing the traffic of another. Sure it was granted that if you could change the firmware on the switch or some other extremely demanding and unlikely processes you could enable a host to sniff another. But this was so speculative that no one pursued it. Instead, some attackers began manipulating the allowed activity of existing protocols in such a way as to route all traffic on a switched network to a specific node, which then re-routed all packets to their intended destinations. This allows one host to sniff all of the traffic on a switched network.

Every day we try to keep up with the new threats, tools, proposed attacks, vulnerabilities, patches and so on. But every one of those days, there are hundreds of thousands of people (many much smarter than we are) working on circumventing or over powering our security measures.

Attackers Have Better Tools And More Knowledge

If Your Computer Contains My Code, It Is No Longer Your Computer

People Want To Help

The reason that social engineering works is not because people are stupid, naive, or foolish. It works because fundamentally people want to be helpful. They want approval, and to feel needed and important. People switch from someone else's artificial procedures of operation over to whatever it takes to meet a need, answer a question, or solve a problem. We are hardwired that way.

While the propensity to want to help is admirable, it can be taken advantage of. The only gating factor is education that takes a hold. Doing an online course is cost effective, but does not translate to effective safe behavior.

While we focus on the actual code, configurations, and other components that are in place, actual attackers include the social engineering

It Does Not Matter If My App Is Hacked

Customers often believe that their simple application does not warrant any security considerations, usually because;

- *It does not use confidential data* -

In reality, confidential data is not the only thing attackers are interested in. Every application, networked or not, represents one or more vectors that offer any number of values to an attacker. In some way, every app represents a stepping stone closer to something of value to an attacker.

- *Only a few people use it* -

The number of people that are intended, expected, or believed to use an application has no baring on the value that compromising that app represents to an attacker.

- *We can recover quickly* -

You may be able to recover quickly, the the damage is still done whether it is recognized or not.

- *No one would be interested in it* -

Again, every app is another potential piece of the puzzle. Fit together enough of the pieces and you have a clear picture of your actual target.

- *There have never been problems before* -

There have never been problems that were recognized or noticed.

This mindset also hides the fact that as an app within an enterprise, its security condition reflects on every other application. If you have one house on fire in a neighborhood, you are at risk of the fire spreading throughout the while community.

Hacker Think

They say that magic is all about misdirection. Well here is a little trick that illustrates the magic in "hacker think".

I do not know how many times I have sat down at someone's desk to do something, looked under

the keyboard and found a post-it note with a password written on it.

So, I take a post-it note, write a complex string on it that looks like it could be a password and put it under my keyboard. I can just image someone finding that and wondering just what they could do with my account, and off they go. All the while I am wondering how much time they would waste with that misdirection.

It is silly and simple, but still an interesting example of thinking in different ways.

There have been occasions where I really did have to write a password down. In those cases I would end the password with 3 spaces and not indicate that on paper. Kind of like salting a password. Again, silly and simple, but an interesting example of "hacker think".

Notice that in both cases you take a look at how people generally behave. Then look for ways to turn that to your advantage (being more secure). And finally, you do not rely on one mechanism. Effective security is comprised of layers of controls that together defend against intrusion to the point you can accept the risk.

Likewise, hacking is often about combining layers of technique to accomplish a task.

Tinker Toys

In tech, and many other areas, everything is made up of "building blocks" - atomic items that cannot be divided down into more simple things (we like to think). And it is the creative, imaginative combining of these items into structures that create systems and solutions.

In the world of tech, it seems that every few months "they" come up with something "new". A revolutionary widget that ingeniously solves problems that were otherwise intractable. The problem is that the majority of the time, that is just marketing rather than invitation.

The building blocks, tinker toys, that are used to build tech solutions, products, concepts are usually new combinations of well established building blocks. Everything relies on the relatively small collection of items that make up our building material.

Of course the building blocks are often improved upon and enhanced, but their fundamental structure remains the same. The little round wheel with holes around it get a couple more holes added or is made thicker. The sticks with the slots at each end come in new colors and lengths. But they are still wheels and sticks, and they still interact the same way.

Think of a "new" tech product or item. Chances are it is simply the latest evolutionary step of a collection of well established components being combined in new ways or enhancing the structure's function.

Actual new technologies are few and far between. For example, quantum computing is a whole new collection of very odd building blocks. They bare no resemblance to existing technologies, operate entirely differently, and solves equations in some cases before they are calculated. This will require an entirely new skill set, paradigms of how/why things are done and why... It will be more momentous then the transition from industrial steam power to an electric powered society.

There Is Nothing New Under The Sun

This is taken from Ecclesiastes. The idea here is that there is seldom a actually new technology is introduced. The majority of new stuff consists of the re-use of established technologies for new and different purposes, or a new combination of established technologies. The good news then is that

you do not need to waste time on all of the new "revolutionary" products when you see them as simply new twists on established technologies.

For those new things are that pertinent to what we do or of interest for some other reason we can review a new thing to identify how it varies from its predecessor components or unique uses.