BELARC

Trusting Anonymous Computers

by
Andrew Ridgers
18th June 2005

Introduction

One of the clear desires of the Jericho world is that two computers should be able to decide if they would like to respond to each other. In current environments, these decisions tend to be based on practical elements. Do they share the same network, or do they agree about some password or encryption scheme? Typically, considerable preparation is required if a computer from outside your network is to be allowed trusted access to resources behind your firewall. This paper discusses mechanisms that are needed for trust to be established between two computers with the minimum amount of preparation beforehand.

Scope

In any connection between two computers, the true identity of each party needs to be established. Other concerns include the physical security of the connection, and the environment in which the computers are being used. These areas are not addressed in this paper. This paper focuses on the task of assessing the internal state and configuration of a requesting system. Essentially, we are attempting to decide whether this requesting computer has been compromised in some way, or whether its security state would constitute an unacceptable risk if were to allow a connection with it.

Trust Inside an Organisation

With computers that are under our control, we minimize risk by establishing security policies. We force users to change passwords, we insist that security hot fixes are applied, we only allow access from behind a firewall, and, and, the list goes on. As networks

become more extensive, and security threats more sophisticated, the list of precautions grows. Inside our own network we have difficulties ensuring that all policies are adhered to, so can we possibly expect to extend similar considerations to computers that are outside of our direct control?

External Requests

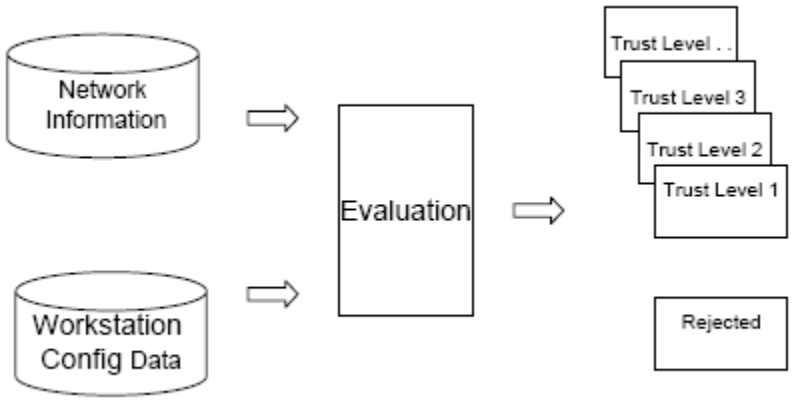
Requiring each computer that offers a service to be able to evaluate each and every request it receives will be prohibitive in terms of cost and maintenance for most organizations. A simpler model is one where all requests are routed to an evaluation point, which can apply rules to establish a trust rating for any inbound request. This model also fits in with preferred practice to reject and isolate attacks at some peripheral point, rather than allowing an attacker to penetrate into the system before rejection. The service providers will then simply extend their services to systems that achieve some trust rating. Notice that, once such a model is in place, there is no reason that it should not be extended to workstations inside the organization as well. We do not need to consider how sensitive the requested services are, since this is typically a business decision, not a technical one. It is enough for us to recognize that allowing any other computer to have access to our services is an event that carries risk. We want to be able to act in a way that minimizes our risk, and also, of course, provide access to our services as widely as possible.

Further, there is no need to assume that we are limited to a single evaluation point. Depending on the nature of the services on offer, and the variety of possible users, it is reasonable to propose different access points for different sources of requests. For instance, we might have one access point for external access between an Internet facing VPN device and the internal network, and a second for the internal network to the finance department network.

What can we discover?

There are two main sources of information that are available. We have information based on previous activity and practice. One of the concerns with this kind of information is that its current value is highly questionable, since we have to regard any and all computers as under constant attack. A machine that passed our most stringent tests one month ago may easily have been compromised since. This kind of information can be stored at the evaluation point, and does not have to be discovered or delivered in order to be used in an evaluation of trust. Typically, this kind of information can be used to decide when to deny a request, but is not very dependable as a source for approving a request. A simple example of this is in the construction of blacklists.

The second kind of information is based on the current state of the requesting system, and can only be obtained by some kind of auditing software. This information needs to be presented to us in order for us to evaluate it. As with the above, having current, or up to date information is essential. If we are to be thorough in our evaluation, then the information about the current state must be as complete as possible.



What information do we need to make an informed decision?

We need to establish that the system

- has blocked any known security risks
- is running an appropriate version of any required software
- is not running prohibited software

Unfortunately, each organization will have its own definition of the first element. Anti-virus software, operating system hotfixes, hotfixes for key applications, password policies, no other open network connections are just examples of elements that might be specified. Provision of all this information to the evaluation process is also complicated, since it needs to be provided in a form that allows evaluation to take place. We must also allow for the fact that, depending on the operating system that the requesting machine is running, the actual data that we need may vary.

We can simplify this process considerably by the use of recognized benchmarks, such as those published by the Center for Internet Security (http://www.cisecurity.org/), to establish that the system has not been compromised. These benchmarks recognize all of the elements published above as well as many other operating systems settings. Since these benchmarks encompass a variety of different variables in use in different operating systems, they provide a much more efficient way of quantifying the current security stance of a system.

We also need to be able to define any other conditions for success, such as having particular versions of custom software. It would be impractical to suggest that the requesting machine should deliver every possible piece of information when sending its request, so it makes more sense that the evaluation point ask for it. This will involve us in some additional, unavoidable steps. When an initial request is received, it must be possible for the evaluation point to respond with a request for additional information. There must be something on the requesting machine that has both collected the original information and which has the ability to respond to this additional request.

Equally important to being able to specify software that must be present on a system is the ability to specify software that must be absent. Eventually, such exclusions must be able to cover other aspects of the system state, such as, for example, no other active network connections.

How do we discover and receive this information?

Our central requirement is that the requesting system must be capable of providing detailed state information. This information should include any relevant items on the system, including, but not limited to, Service Packs, Critical and Security Hotfixes, Audit and Account Policies, Password Policies, Event Log Policies, Anonymous Account restrictions, Security Options, Available Services and User Rights. Rather than pass all of this detailed information, this should be tested against current benchmark definitions. Since these definitions are not static over time, the source of this information must be able to receive updates. For instance, one of the more common security benchmark definitions is that a system should have applied all Security Hotfixes. This is a moving target, and the source of the state information must have an update mechanism to ensure that it is using an up to date definition of a benchmark requirement.

Communication of this information and configuration of the evaluation point to specify what actions are appropriate are responsibilities for the vendors of network equipment (and they all have initiatives in this area).

How do we handle failure to comply?

With a mechanism established, a request from a system can now be rejected. What is more, the rejection can be for a specific reason or reasons, or simply because there was no audit agent to supply the required security information. Since we accept that our benchmark targets are not constant, we have to recognize that a system may connect successfully yesterday, and then fail today. It is important to handle such failures gracefully, or we will create a poor relationship with those partners or customers that we wish to allow access to our services.

When there is simply no audit agent on the requesting system, our choice is limited. We can simply reject or ignore the request if we see fit. Alternatively, we can route the request to a dialogue which will explain the need to install an auditing agent in order to process the request.

Where the failure is more specific, perhaps just a single missing hotfix, we have other options. If the audit agent is capable enough, we could instruct the agent to install the hotfix in order to make the connecting system conform to our requirements. Again, the alternative is to route the request to a dialogue where the requester is informed of the nature of the failure, and shown how to correct the issue. In fact, since the evaluation point should have very specific information about why the request is being rejected, there is an option to build very complex dialogs to assist users.

There is no single correct action when a requesting system fails to meet some minimum requirement. High security systems may choose to simply not acknowledge any request that fails to meet the required standard. Automatically applying a software patch may be

an attractive solution for some organizations, where there is a clear agreement that requesting computer must conform to some minimum standard. Generally, this choice will be both inappropriate and wrong when, for instance, a partner's computer is attempting to connect. The partner may well have their own standards for applying patches, and doing something in contradiction of their policies will be unwelcome.

Practical Considerations

Systems must be capable of scaling with no added cost. If the trust evaluation system is so primitive that it can only approve or deny a request, then there will be a high support load associated with most of the denied requests. A system which is able to specify the exact nature of the failure directly to the requesting system should remove most of this support load, and will be inherently more scaleable.

Some elements of a security benchmark will vary over time – for instance, the list of security hotfixes for a particular operating system, or the recommended version of an antivirus definition file. Such information must be automatically assimilated by the evaluation point, so that we do not create windows of opportunity while waiting for definitions to be manually updated. It will not be enough for the evaluation point to simply ask if the anti-virus definitions are up date. We must also know whether the anti-virus software is configured appropriately and whether a system scan has been run recently.

Assessing a system as secure, or trusted, is a continuous process. If a transaction requires a long connection, it should be possible for us to specify that the requesting system is frequently challenged. It must be possible to configure the evaluation point to require that tests are reapplied at some frequency, and specify an action to be taken if the state of the requesting system deteriorates in some way.

What are the key elements of a preferred model?

We began by asking how we could reduce the amount of preparation and negotiation that is required before allowing external computers to connect to our systems for services. Our conclusion is that a simple security audit agent on the requesting machine, coupled with support at our evaluation point, can provide us with a comprehensive mechanism to evaluate risk from connecting systems. This security audit agent can even be offered via, say, a website, as a necessary precursor for access to offered services. This would allow an organization to offer controlled access to sensitive services without any negotiation with the prospective user.

This model offers services to systems that meet some minimum trust rating, which can be scored and specified along multiple dimensions. Each requesting system will be assigned a trust rating by an evaluation point, based on a benchmark score supplied by the system when challenged during network access. If the trust rating is high enough, then the requesting system is allowed access to the service.