Its a Trust Thing

Geoff Huston
October 2001

The events of the last few months has led us all to look again at the services and facilities we often take for granted and ask the question as to how resilient such services are in the face of determined attack. The Internet is no exception to such an evaluation, and nor should it be. The Internet is assuming an ever-broader role of underpinning all kinds of commercial, governmental and personal activities, and its now most important that it operates as an essential service.

How resilient is the Internet to attempts to disrupt it? A quick examination of the archives reveals a well packed history of attack and response. My personal experience starts with the Internet Worm of the late 1980's which exploited a bug in a number of commonly deployed utility programs on Unix hosts. Since then I've seen a steady stream of attacks on the DNS name service, attacks using TCP SYN flooding, distributed denial of service attacks, attacks on the routing system, and most recently Code Red and its variants. Why have we been unable to eliminate such attacks on the network and its attached host systems?

One line of reasoning is borrowed from work in formal systems some seventy years ago. At that time the conclusion was reached that any formal system that was sufficiently expressive to be “useful” was sufficiently expressive to contain paradoxes. In other words any sufficiently powerful formal system contains the seeds of its own internal contradictions. Perhaps, when considering the Internet and it constellation of attached host systems, we have developed a system where the components of the system are sufficiently complex that they are vulnerable to attack by virtue of their complexity alone. In the same way that paradoxes are an unavoidable byproduct of any sufficiently powerful formal system, vulnerabilities are an unavoidable byproduct of any sufficiently powerful computing and communications system. While such an observation is tempting, I do not believe that it is either useful or accurate. The vulnerability of the Internet is not in the complexity of its components but in the trust model of the Internet.

So lets look at the trust model of the Internet and see what we might do to improve it.

All networks have an inherent trust model. In some cases the trust model relies on the ability of the network provider to operate the network correctly. In other cases the trust model relies on each user acting appropriately, and the network provider is in no position to enforce such user behaviour. The Internet would fall into this latter category.

At a basic level the Internet is a simple datagram delivery network. The network itself does not define any service other than packet delivery. Surrounding the network is a collection of end-systems. It is these systems that define the service model of the Internet, and define how the network resources are to be shared across competing demands. Some of these systems are operated by the same entity that manages the network, but the overwhelming majority of such systems are operated by other users of the network. In the Internet's trust model every user is trusts, to some extent, the benign intent of every other Internet user.

To illustrate this distributed trust model, lets look at the action of download a web page using a browser.

Once the URL is entered into the browser, the domain name part is passed to the DNS in order to translate the name to an IP address. The DNS is a highly distributed database, where various components of the database are operated by a diverse collection of operators. It would be comforting to believe that the DNS provides accurate answers to queries all of the time. This is not the case. It would possibly be an acceptable compromise to believe that all incorrect answers are the result of temporary faults or inadvertent operator errors. Unfortunately even this is probably not the case. The DNS is a target of various forms of attack, and in some cases such attacks are successful. In such cases the DNS provides incorrect answers, directing the user to a site that may then attempt to compromise any ensuing transaction.

Once the DNS returns an IP address, the next step is to open an HTTP session. The first step is to send a TCP SYN packet to the IP address to start the connection. How does the user know that the packet will be delivered to the correct server? Here the user is trusting the integrity of the Internet's routing system. Again, it is not clear that such trust is well placed all of the time. The routing system is vulnerable to incorrect routing information being injected into the routing system, and as a consequence packets will be misdirected. While the majority of such incidents are attributable to operator error, there remains the potential that the routing system is vulnerable to deliberate attack.

When the TCP session is opened, the user's HTTP session then requests the requested object from the server. Is the object delivered by the server that original object that was placed there by the content originator? Servers are vulnerable to attack, and defacing a web site by substituting altered content in the place of the original is not uncommon. While some forms of content substitution are readily apparent to the user, other forms of substitution may be far more subtle in their intent, attempting to convince the user that the content is genuine and that the web transaction is trustworthy. Other forms of attack to the web transaction are more difficult to trace. If the user's request is passed through a web cache, then an attack on the cache can result in the delivery of substituted content to the user while the content originator's site has maintained its integrity.

So when you enter a URL and get back a page a lot of trust is happening. Probably too much trust.

How can we improve the situation? Many of the basic tools are already available.

Widespread use of a trustable public key infrastructure would assist in allowing users to validate that the content has been generated by the author. In a model of complementary public and private key pairs a web object can be signed by the originator's private key. If the object's signature can be validated against the originator's public key then there is a strong indication that the content is genuine.

Email can also benefit from widespread use of the same public key cryptography, where a message can be signed with the author's private key and the signed message can then be encrypted using the intended recipient's public key. Assuming that the keys have not been compromised, such a message can only be read by the intended recipient, and could only have been sent by the owner of the digital signature.

Imagine for a second if a public key infrastructure was in widespread use and we only accepted email that was digitally signed using keys that we trust. Imagine also that web objects were also signed in a similar fashion. In such a world how would a spammer hide their true identity? If spammers cannot hide behind anonymity, then would spam be as prevalent in such a network? Would various e-commerce applications enjoy greater acceptance if users were confidant that there was both privacy and authenticity associated with such transactions? What would be the point of subverting a web site or its content if the action were immediately visible as an unauthorized alteration? Not only would such a network mitigate some of the problems we see today, but its likely that we would find new services appearing that rely on a useful end-to-end trust model where each end can validate the identity of the other party and also validate the authenticity and privacy of the exchanged data.

We can go beyond the application level in this, an d look at how we can improve upon the trust model in the network itself and its associated service layer. DNSSEC allows a DNS server to ensure that only known remote agents can provide updates to the server, and that the updates are not altered by a third party on the fly. If DNSSEC were an integral part of the operation of the DNS it would be a far more difficult proposition to attempt to insert incorrect data into the DNS. At the network level, providers commonly operate Internet networks with efficiency and unit cost in mind. In-band operation of routing protocols readily fit within such an operational model. The SNMP-based model of network management is also commonly operated as an in-band tool. The result is that the operational management of the network, the monitoring of the network and the payload carried by the network all share the same transport system. There is the risk that a payload packet can masquerade as a routing control packet, or an SNMP write transaction, and if the network does not detect this as an exception, the network can be disrupted, or even conceivably taken over by a third party. The trust model here is obviously not that the network operator trusts users not to generate such attacks. The appropriate model is that the network operator distrusts users, and takes appropriate precautions to ensure that a network element will only accept control and monitoring packets that truly originate from the network operator's internal control points.

It would be useful to have each registry-allocated address block be associated with a public key. This would allow all new routing requests to be signed by the private key associated with the original address allocation, which in turn would assist the ISP in determining the authenticity of the request before entering the address prefix into the Internet's routing system.

Its not that we can eliminate the need for some degree of trust from the Internet, nor that all potential security risks can be comprehensively addressed. But in using a set of basic security tools with our existing network applications we can make some significant improvements upon the current rather open trust model of the Internet. In so doing we can make significant improvements in the level of trust users can reasonably place on the integrity of the Internet as a useful communications medium. And surely that's a good thing.