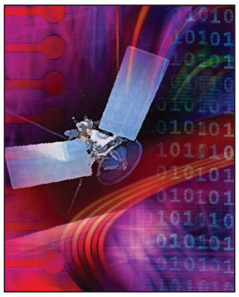
Security Boundaries



Stephen Farrell • *Trinity College Dublin*

Security boundaries are usually defined by a set of systems that are under a single administrative control. These boundaries occur at various levels, and vulnerabilities can become apparent as data "crosses" each boundary. In this inaugural column on practical security, I'll look at a range of boundaries from smaller to larger and describe a couple of vulnerabilities in each case. I'll then discuss a potential path forward (although not a solution) to tackling such issues based on the idea of a system's attack surface. First, however, let's look at some attacks that ignore boundaries entirely.

Side Channels

We currently have a range of resilient and broadly applicable security protocols such as Transport Layer Security (TLS) and IPsec. However, side-channel attacks continue to be found with these protocols, the classic being the "million message" attack,[1] which uses TLS's error-handling behavior to extract a fraction of key information per probe.

At a recent meeting of the IETF's Security Area Advisory Group (SAAG), Russ Housley (currently the IETF chair) reported on a demonstration Yossi Oren and Adi Shamir gave on a side-channel attack that involves power analysis (their demonstration was from the "rump session" of the Crypto 2006 conference; see http://www3.ietf.org/proceedings/06nov/minutes/saag.txt). The basic demonstration showed that even with the power lines from a PC to a USB port physically cut inside the PC, you can still use a USB token to detect power variations during an RSA decryption operation, presumably sufficient to recover the plaintext or private key. This attack nicely demonstrates that side-channel attacks can be very hard to counter, even if we create a new physical security boundary inside a host (by cutting the lines). Of course, virtual boundaries are much more common inside systems today, but they also leave open potential attack vectors.

Host Virtualization

Increasingly, hosts run virtual machines (VMs) for various reasons. Peter Ferrie[2] describes some ways in which an attacker can detect VMs and, in some cases, cause them to crash. These range from timing discrepancies to limitations in the fidelity with which the VM emulates the underlying hardware. Going a step further, Tavis Ormandy[3] demonstrated a range of vulnerabilities in VMs that could let malware subvert the VM (as can, of course, occur with any complex software).

Using a VM to provide better separation between running processes still provides a high level of protection against accidental and many — although not all — intended bad events. However, we must face the fact that virtualization doesn't mean that processes running on each of the guest operating systems are totally separated, no matter how much we'd like that to be the case.

Software Update as a Boundary

The near ubiquity of software updates makes it hard to describe a host's current state at any given moment. Although it seems essential to have automated software updates (to avoid known vulnerabilities), this means that almost all hosts are continually installing new code, that some of this code is privileged, and that those hosts are essentially part of that subset of the Internet running the software in question. Thus, each software update "domain" (whether updates are controlled via an enterprise server or done separately between each end host and the software author) constitutes another kind of security boundary.

That such updates can cause unexpected widespread problems has been demonstrated with the outage one voice-over-IP (VoIP) operator recently suffered (see http://heartbeat.skype.com/2007/08/what_happened_on_august_16.html). From this example, we can easily envisage how an attacker could use a compromised software update mechanism as quite an effective attack vector.

Enterprise Network Boundaries

The demilitarized zone (DMZ) has been a cornerstone of enterprise security for roughly a decade. The basic idea is to isolate the more trusted, inner enterprise network from the untrusted Internet and locate some basic services (DNS, email, and Web) in the DMZ to effectively filter traffic so that systems outside the enterprise network see only services that are authorized to be externally visible. However, the advent of enterprise wireless networks and the fact that many employees access enterprise networks from home (via various VPNs) means that the inside–outside distinction is much fuzzier these days. In some cases, this fuzziness has enabled significant attacks, such as the recent attack on the T.J. Maxx chain in which weak wireless security in one store let attackers gain access to the company network, ultimately resulting in the theft of millions of customers' credit-card details (www.informationweek.com/news/showArticle.jhtml?articleID=199500385).

Partly influenced by such events, the Jericho forum (www.opengroup.org/jericho/) has been espousing *deperimiterization*, essentially arguing that the DMZ concept is no longer sufficient and that modern enterprises must distribute security functions to various other places in the network — for instance, by using security-specific application-layer gateways located in the same rack as an application server so that security enforcement (such as requiring TLS) occurs "closer" to the applications. We could view deperimeterization as essentially replacing the DMZ concept with a more distributed and complex DMZ, so that each end system in the network views the inside–outside distinction differently.

Middleboxes

The term *middlebox*[4] refers to a host inside the network that carries out some nontrivial function (other than routing/forwarding) on behalf of applications running on end systems. Common examples include network address translation (NAT) boxes and firewalls but also content delivery servers. One problem with such middleboxes is that they're often invisible to end systems and so can easily end up omitted from a security analysis. To take one example, the proxies used by the session initialization protocol (SIP) are middleboxes involved in establishing VoIP calls; several such proxies can be involved in a single call, with various ones belonging to enterprises, individuals, or operators depending on the deployment (that is, you won't know in advance which SIP proxies will be involved in call setup). One example of a SIP proxy vulnerability is a buffer overflow reported as CVE-2005-4466 (see http://nvd.nist.gov/nvd.cfm?cvename=CVE-2005-4466), that would have let an attacker potentially execute arbitrary code on a vulnerable SIP proxy. SIP proxies are complex software, so no doubt other vulnerabilities will emerge. We can envisage a similar exploit that would automatically set up a conference call that included the attacker whenever a target makes a call — essentially, a wiretap.

Web 2.0

Although Web 2.0 might be as much a marketing as a technical term, a class of new applications do make extensive use of Asynchronous JavaScript Technology and XML (AJAX), with back-end Web services carrying out their functions. Such applications involve more processes executing on more hosts than traditional two- or three-tier applications, and the interactions between those hosts and processes cross the types of boundaries I discuss here. Essentially, Web 2.0, by distributing computational and storage loads, makes it an order of magnitude harder to describe an overall system and — complexity being security's enemy — much more likely that serious vulnerabilities remain undetected with it.

For example, a back-end Web service might depend on data validation that happened earlier in the overall processing of some request. Initial deployment might ensure that such validation occurs at a relatively trusted (or at least accountable) server, but subsequent deployments could easily add new interfaces in which data validation happens on the client, totally changing the trust model. A good approach, but one that's probably rarely done, would be to conduct data validation at all processing stages.

Web 2.0 services also frequently involve massive amounts of storage — storage that attackers can and have used as a malware distribution vector (see www.techpin.com/malware-spread-through-social-networking-websites/). Basically, with so many end systems accessing so much increasingly complex content (images, video, and so on), it becomes harder and harder to "clean" such data stores.

Lastly, Stefano Di Paola[5] describes how JavaScript applications, as used in Ajax, are vulnerable to previously loaded code overloading system calls (in particular, the `XMLHttpRequest object`). The end result is that mashup-style applications essentially depend on the security of all the Web sites involved in the mashup; if any source sites become compromised, then all mashups using that site might also be compromised.

The Internet Is a Big Place

The next-to-last identifiable boundary I consider here is the current Internet, which is a big place. The impact of scale on our (in)ability to control security is clearly visible at this level — spam's predominance and phishing's continuing success are two good examples of problems we can easily control at a smaller scale, but for which we don't currently have acceptable solutions that can cover the entire Internet. In fact, we could view the lack of Domain Name System security (DNSSEC) deployment as a good indicator that we don't yet know how to provide security for systems on an Internet-sized scale.

One interesting development at this level has been botnets' increasing scale (see www.sans.org/reading_room/whitepapers/malicious/1299.php), which has resulted in attackers increasingly abusing compromised end systems for monetary gain. The fact that botnet zombies appear to exist inside all "secured" boundaries indicates that the security-as-boundaries model might itself have scaling problems.

Above the Application Layer

Finally, in a nice example of social engineering that moves us beyond technical boundaries and into social and economic ones, employees of an unnamed credit union recently found USB tokens containing malware scattered in the public and smoking areas outside their offices (the credit union had paid for a penetration test; see www.darkreading.com/document.asp?doc_id=95556). The employees helpfully picked up the "lost" tokens and brought them inside the corporate network, where the malware was executed once the tokens were inserted into PC USB slots. In this case, the boundary most broken is a corporate policy one, but it's not easy to see how to prevent such an ingress in general, given that a new, interesting peripheral will always exist that's cheap enough to scatter (or distribute at an industry show). User education simply might not suffice here, although it will help if most users don't in fact attach the peripheral to a corporate box.

The recent Estonian[6] and Blue Security (see www.theregister.co.uk/2006/05/17/blue_security_folds/) distributed denial-of-service (DDoS) attacks demonstrate how attackers have used botnets for (more or less) orchestrated political purposes and retribution against security providers. In this case, in response to a political controversy, attackers first targeted the Estonian foreign ministry, but then spread the attack to target other government and financial sites. More than 100 separate DDoS attacks were launched against Estonian sites over a two-week period, and in the end, the Estonian authorities had to cut some external links to maintain local access to the relevant sites. In some cases, this prevented international financial transactions, such as cash withdrawals from foreign accounts. These attacks highlight how botnets represent a new boundary as a set of controlled hosts that the botnet operators (and their customers) can quickly leverage for nefarious purposes. Estimates[6] for the costs of these attacks (US$100,000 to $1 million) indicate that botnets' economics might become a significant factor in future Internet security.

Spam's predominance and phishing's success are problems we can easily control at a smaller scale, but we don't currently have acceptable solutions that can cover the entire Internet.

What to Do?

Although this column intends basically to provoke thought about security boundaries and their effectiveness, I should at least offer some hope that we can make progress, even in the face of the problems I've mentioned.

First, you should, of course, follow best practices in secure development, concentrating on reducing risk and not on making any particular part of a system uselessly secure. For brevity's sake, I'll assume, that you already know how to "do the right thing," which includes modeling threats, conducting risk analysis, and deploying standard countermeasures (such as VPNs and audits).

One interesting concept worth highlighting here is that of the *attack surface*,[7] which essentially captures the notion that systems with more complex interfaces are more likely to suffer from unknown vulnerabilities. Perhaps the best example was an attack on an encryption hardware box,[8] in which the existence of both data and key handling interfaces created the possibility of an attacker using the box to decrypt an encrypted key. Even though the vendor had gone to extreme lengths to make the equipment "secure," the fact that there were so many interfaces meant that it couldn't test all call sequences (or perhaps even envision them), creating the vulnerability.

The idea here is to try to capture a system's attack surface in a way that lets you compare it with similar systems to give a relative measure of potential insecurities — if you can say that two different systems (for the same function) have significantly different attack surfaces, then other things being equal (which they're not, of course), you should choose the system with the smaller one.

In some sense, this is really an extension of the least-privilege principle applied to networked systems. The important thing to take away for now is that it doesn't really matter how you initially measure or model the attack surface of a system you're developing or deploying, as long as you act to make that surface smaller or at least only extend it when an extension is justified.

Risk doesn't come in layers, and it isn't corralled into boundaries, and nor should security analysis. We should instead think in terms of leaky boundaries related to our systems, identifying those that are most relevant and attempting to impose appropriate audit and control on information flowing across them. Minimizing the attack surfaces related to all the system, application, and network design work we do seems promising at the moment and warrants continued consideration as the attack surface analysis technique develops in the future.

References

1. D. Bleichenbacher, "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1," LNCS 1462, Springer-Verlag, 1998, pp. 1–12.
2. P. Ferrie, "Attacks on Virtual Machine Emulators," Symantec Advanced Threat Research, Dec. 2006; www.symantec.com/avcenter/reference/Virtual_Machine_Threats.pdf.
3. T. Ormandy, "An Empirical Study into the Security Exposure to Host of Hostile Virtualized Environments," *CanSecWest 2007*; http://taviso.decsystem.org/virtsec.pdf.
4. B. Carpenter and S. Brim, "Middleboxes: Taxonomy and Issues," Internet RFC 3234, Feb. 2002; www.ietf.org/rfc/rfc3234.txt.
5. S. Di Paola and G. Fedon, "Subverting Ajax," *23rd Chaos Communication Congress*, Dec. 2006; http://events.ccc.de/congress/2006/Fahrplan/attachments/1158-Subverting_Ajax.pdf.
6. M. Lesk, "The New Front Line: Estonia under Cyberassault," *IEEE Security & Privacy*, vol. 5, no. 4, 2007, pp. 76–79.
7. P. Manadhata and J. Wing, *An Attack Surface Metric*, CS tech. report, Carnegie Mellon Univ., July 2005; http://reports-archive.adm.cs.cmu.edu/anon/anon/2005/CMU-CS-05-155.pdf.
8. M. Bond and R. Anderson, "API-Level Attacks on Embedded Systems," *Computer*, vol. 34, no. 10, 2001, pp. 67–75.

Stephen Farrell is a research fellow at Trinity College Dublin. His research interests include security and delay/disruption-tolerant networking. Farrell has a Joint Honors B.Sc. in mathematics and computer science from University College Dublin. Contact him at stephen.farrrell@cs.tcd.ie.