Concordia University College of Alberta

Master of Information Systems Security Management (MISSM) Program

7128 Ada Boulevard, Edmonton, AB

Canada T5B 4E4

A Novel Strategy for Internetwork Segmentation and Zoning

by

LINDSKOG, Dale

A research paper submitted in partial fulfillment of the requirements for the degree of

Master of Information Systems Security Management

Research advisors:

Pavol Zavarsky, Director of Research and Associate Professor, MISSM

Ron Ruhl, Director and Associate Professor, MISSM

A Novel Strategy for Internetwork Segmentation and Zoning

by

LINDSKOG, Dale

Research advisors:

Pavol Zavarsky, Director of Research and Associate Professor, MISSM

Ron Ruhl, Director and Associate Professor, MISSM

Reviews Committee:

Ron Ruhl, Assistant Professor, MISSM
Pavol Zavarsky, Associate Professor, MISSM

Concordia University College of Alberta

Master of information Systems Security Management (MISSM) Program

7128 Ada Boulevard, Edmonton, AB

Canada T5B 4E4

A novel strategy for internetwork segmentation and zoning

By

Dale Lindskog

A research paper submitted in partial fulfillment of the requirements for the degree of

Master of Information Systems Security Management

27 March 2007

Research Advisors:

Pavol Zavarsky
Associate Professor and Director of Research, Information Systems Security Management
Concordia University College of Alberta

Ron Ruhl
Assistant Professor and Director, Information Systems Security Management
Concordia University College of Alberta

ABSTRACT

In this research paper, a new systematic approach to segmenting nodes in an internetwork is proposed. The research paper shows advantages of screening traffic between clients and the servers with which they communicate. The proposed strategy is intended to supplement, but not displace, other approaches to network security zoning. Unlike many other firewall architecture strategies, the new strategy is not intended as a network solution to deficiencies in host security. Rather, this strategy is intended primarily to (1) mitigate the consequences of host compromise, and (2) to facilitate network security monitoring.

1. Bellovin's Dictum

Steven Bellovin is well-known for having remarked that 'Firewalls are a network response to a software engineering problem' [1]. In this essay I will, firstly, consider the extent to which this claim is true. Secondly, and, I think, much more importantly, I will draw consequences for firewall architecture strategy from these considerations.

Let me begin by saying that this remark of Bellovin's needs clarification. For, on one conception of a 'software engineering problem', Bellovin's remark is not plausible. His point cannot merely be that packet filtering[1] is a network response to the problem of exploitable software running on a host. For the sake of argument, imagine that there are no software design or implementation vulnerabilities. If we imagine this, it cannot be denied that firewalls may still play an important role in protecting systems against attacks directed at *configuration* vulnerabilities.[2] An obvious example is failure to disable a dangerous network service.

Bellovin's dictum can be defended, however, by refining our conception of a software engineering problem. Let us define 'software engineering' quite broadly, e.g., as it is defined by IEEE: 'the application of a systematic, disciplined, quantifiable approach to the development, *operation*, and maintenance of software...' [2]. On this conception, then, we might state Bellovin's principle more precisely by saying that 'firewalls are a network response to a *host security* problem'. In fact, this is how Bellovin himself originally characterized it [3], and I shall take it to be his intended meaning.

On this conception, Bellovin's dictum is indeed more plausible. For, there is no doubt that, from a practical point of view, host security problems are a major reason for employing packet filters: packet filtering restricts access to network resources. As a case in point, imagine a particular server that is intended only to service SMTP clients, from the Internet, on its TCP service port 25. In this case, we certainly could use a packet filter to prohibit those clients' access to all *other* service ports on this SMTP serving host.

This example illustrates one kind of way in which a packet filter might be a network response to a host security problem. In this example, the host security problem at issue is the potential failure to disable unnecessary services. This use of a packet filter *ought* to be unnecessary, since if this particular server really is intended only for this particular purpose, then there ought to be no other listening (and thus potentially exploitable) services on that host. That is to say, it is an application of Bellovin's dictum because, in the absence of this particular sort of host security problem, this use of a packet filter is both superfluous and useless. It is superfluous because the services the packet filter is configured to protect are not even listening. It is useless because it does not protect the service that *is* listening.

These considerations show at least the limited applicability of Bellovin's dictum. Here at least, in so far as packet filters are useful, they are responses to problems in host security. Clearly, then, Bellovin's dictum captures some of the benefits of packet filters. It does not, however, capture all benefits. One such benefit immediately comes to mind:

Sometimes we use packet filters, not to (completely) restrict access to unnecessary *services*, but rather,

1 Given that this remark was written in the mid-1990s, I take it for granted that, by a 'firewall', Bellovin had foremost in his mind a packet filter. (But I do not prejudge its more general applicability to firewalls.)

2 For an elaboration on the distinctions between design, implementation, and configuration vulnerabilities, see [4].

to restrict unnecessary *access* to (necessary) services. For example, imagine that we want to allow access to an SSH server port on the aforementioned SMTP serving host. Suppose further that, although we do indeed want access to this port, we want it only if the connection is initiated from a client on an administrative network. This is something that is easily accomplished with a packet filter. But is this an example of a network response to a host security problem? The addition of the SSH server in this scenario is not the addition of an unnecessary service. It is, we suppose, necessary *per se*. What is unnecessary (we are assuming) is allowing *unrestricted* access to that service port.

This example, I think, is sufficient to show that Bellovin's dictum is not universally applicable. For this is certainly a well-established use of packet filtering, and is clearly not a mere network response to a host security problem. There is, however, yet another counter-example to Bellovin's dictum, and indeed it is a less obvious – and less well-established – use for packet filters: sometimes we use packet filters, not as a response to host security problems, and not as a method of restricting unnecessary access. Rather, sometimes we use packet filtering as a tactic to *mitigate the consequences of host compromise*. This counter-example plays an important role in my argument, and so I will consider it in some detail in the next section.

2. From a tactic to a principle

Sometimes we use packet filters to mitigate the consequences of (potential) host compromise. This counter-example to Bellovin's dictum becomes manifest when we consider motivations behind the classic screened subnet architecture:

The screened subnet architecture is often introduced as an improvement over the screened *host* architecture [5]. Recall that the screened host architecture describes the division of an internetwork into two 'zones' (usually, into a trusted zone (TZ) and an untrusted zone (UZ)). This division is effected by the introduction of packet filtering to control traffic passing between these two zones. Recall further, that the screened *subnet* architecture expands on this, by introducing a perimeter, or 'de-militarized' zone (DMZ) between these trusted and untrusted zones. Historically, it is described as placing a perimeter subnet between an untrusted internetwork, such as the Internet, and the internetwork that you are trying to protect, such as an organization's internal internetwork. Protection is afforded by filtering traffic both (1) between the untrusted zone and the screened subnet, and (2) between the screened subnet and the organization's internetwork.

How is the screened subnet firewall architecture an improvement over the screened host architecture? The improvement turns on the placement of (what we call) the 'bastion host(s)'. A bastion host is so-called because, relative to other hosts in an internetwork, its role requires a higher level of security. Often, a bastion host is 'your public presence on the Internet ... [and] is [therefore] exposed to potentially hostile elements' [6]. Within the confines of the screened *host* architecture, your *only* options, for the placement of bastion hosts, are (a) and (b) below:

(a) You might choose to place bastion hosts on the Internet (or another similarly untrusted internetwork), i.e., outside your organization's defences, or,

(b) You might choose to place your bastion hosts inside the internetwork you are trying to protect.

Either approach has disadvantages, and thus the screened host architecture has *inherent* disadvantages in this context, e.g., in the event that you need to host an Internet server. Let us consider the disadvantage of each approach in turn.

With (a), your bastion hosts must rely purely on host security. While with (b), you place your internal network at greater risk, simply because you are placing a high risk host, most importantly a host accessible from the Internet, behind your packet filter. The screened subnet architecture is the natural solution to this dilemma: your bastion host is still protected by filtering, yet your internal internetwork is no longer exposed in the aforementioned way.

But note: the precise benefits of introducing a screened subnet depends on whether you compare it with (a), or with (b) above. In comparison with the screened host architecture described in (a) above, the introduction of a screened subnet is consistent with Bellovin's dictum: it is indeed a network response to a host security problem. For, in the case of (a), the advantage conferred by the introduction of the screened subnet is that the bastion host no longer relies purely on host security.

But the case of (b) is quite different. The improvement over (b), introduced by the screened subnet, is that the screened subnet *mitigates the consequences of host compromise*. In particular, it mitigates the consequences of a compromise of the bastion host. That is to say, a compromise of this bastion host in (b) gives the attacker a launching pad with unrestricted access to the trusted zone. This is not so if the screened host architecture in (b) is replaced with a screened subnet architecture, where that bastion host is placed in the perimeter.

Using the screened subnet architecture to mitigate the consequences of host compromise is not, therefore, a network response to a host security problem. (Or, at least, it is not provided we refrain including the *mere possibility* of host compromise under the rubric of 'host security problems'. It seems to me unreasonable to do so.) Moreover, I will argue presently that the improvement derives not merely from the principle of zone separation *per se*. That is to say, it is not *merely* an application of the principle that we ought to separate portions of an internetwork into 'zones', based on their security requirements, and then screen traffic moving between those zones. Rather, I will show that the improvement derives from another principle altogether.

To aid in this demonstration, let us note first that, although the screened subnet architecture is indeed a simple application of this principle of zone separation, it is not the *most* simple. The most simple application of this principle is the screened host architecture previously introduced, for it (the screened *host* architecture) too separates an internetwork into zones. Normally, it separates an internetwork into a trusted and an untrusted zone; the difference between the two architectures is, generally speaking, merely the number of zones.

What, then justifies the *particular* zone separation described by the screened subnet architecture? Of particular interest to myself is the question, What underlies the screened subnet architecture's improvement over (b), i.e., its improvement over placing the bastion host in the TZ of the screened host architecture? To get clear about this, let us ask ourselves the following question regarding the screened subnet architecture. How does network communication with the UZ differ, depending on whether that UZ communicates with the TZ, or with the DMZ?

In a great many cases, the difference between (1) bastion hosts placed in a DMZ, and (2) hosts placed in the TZ, is not simply that bastion hosts communicate with the UZ, while internal hosts do not. Granted, this is occasionally true: internal servers often have no need to access an UZ like the Internet, and, more rarely, workstations are prohibited from doing so. It is quite common, however, for many hosts in a TZ to communicate with hosts in the UZ. An obvious case in point is a web browser running on a workstation, that communicates directly with web servers on the Internet.

Despite this, it is still a good practice to separate bastion hosts into one or more 'demilitarized' zones. *Why is this?* What is it about the nature of a bastion host's network communication that justifies zone separation here?

Consider again an SMTP server accepting mail from the Internet. The ostensible reason for segregating such a bastion SMTP server on to a DMZ, as opposed to hosting that service on an internal, 'trusted' zone, is because this host is highly exposed to the Internet – or some other untrusted zone. As one text puts it, a '*bastion host* is your public presence on the Internet... By design, a bastion host is highly exposed because its existence is known to the Internet' [7].

But what do we mean when we say that 'its existence is known to the Internet'? One possible (but, I think, flawed) explanation is that this service is *advertised*. However, as we all know, it is as easy as a simple port scan to find a listening, accessible service; and this is so regardless of whether that service is explicitly advertised, e.g. via DNS, or by some other means.[3] And nobody is going to propose poking a hole through a packet filter, thereby allowing clients from the Internet to access an unadvertised service port on a host residing in an internal trusted zone. Or, at least, if they do propose such a notion, the justification had better not be that this service is not explicitly advertised to users of the Internet. That would be *security by obscurity* of the worst kind.

Another justification, for segregating a bastion host, is that the 'bastion host is the system that any outsiders – friends or possible foes – must ordinarily connect with to access your systems or services' [8]. And this justification is the correct one, *provided* we interpret 'connect with' as 'connect *to*', i.e., provided we interpret it as a *client-server* connection where the 'outsider' is the client, and the bastion host is the server. Without this clarification, the above justification is insufficient. For it cannot merely be a matter of the fact that hosts on the Internet are *directly* connected with the bastion host. Consider any organization where internal web browsers *directly* access HTTP servers on the Internet (as opposed, say, to *in*directly accessing them via a web proxy). Here too we have a direct connection, yet this fact is rarely used as support for segregating web clients!

No, what we really have in mind (or should have in mind) is that a bastion host has an accessible *service* listening on some TCP or UDP port number. And so, underlying this architectural tactic, i.e., the tactic of segregating the bastion host from the TZ, is the fact that we want to restrict client connections, from that bastion host, to services listening on hosts in the TZ. And what makes this tactic more important than segregating *clients* in the TZ that directly communicate with the UZ? It is, I think, based on a simple principle of network security:

3 By listening *and* accessible, I mean listening and not blocked, say, by a packet filter.

Attack vector principle

- *Other things being equal, server-side exploitation is easier than client-side exploitation.*[4]

One might suggest that this principle is less true today than historically, and I agree; but it is still true. For example, no one is going to suggest that it is less (or even equally) risky to expose a Windows workstation's services to the Internet, than it is to let somebody use the network clients (e.g., a web browser) on that workstation to communicate directly with servers on the Internet. There is a very good, and quite generic reason for this, which is again connected with the client-server model of network communication. The reason is that, in principle at least (but with some exceptions), clients *choose* the servers they initiate connections too, whereas servers do not generally speaking choose the clients that connect to them:

Justification for the Attack vector principle

- *For the most part, but with exceptions (see below),* ***clients choose*** *the servers they communicate with, while* ***servers do not choose*** *the clients they communicate with.*

As noted above, there are exceptions to my *attack vector principle*. For, in practical terms, clients do not *always* choose the servers they connect with. Consider the case of an attacker that has control of a proxy, e.g., an HTTP proxy used by certain web browsers. In this case, and other things being equal, client-side exploitation specifically of those clients is just as easy as server-side exploitation. The reason is simply that, in this case, the clients effectively do *not* choose the server (qua proxy) that they initiate connections to. This exception to my above theorem will also assume importance below, so I will similarly emphasize it:

Exception to the Attack vector principle

- *When a client is configured to (and habitually does) use a particular server (e.g. a proxy server), then, if this server is compromised (and other things being equal) client-side exploitation (specifically of that client) is as easy as server-side exploitation is generally.*

If, as I stated above, a bastion host is usually your public *presence* on the Internet, then the phrase 'bastion host' is used to refer to networked hosts acting primarily as servers – in the strictest sense of 'server' – for clients located in the untrusted zone. That is to say, they act as servers vis-a-vis the client-server model of network communication so prevalent in TCP/IP. On the other hand, hosts in the trusted zone that communicate with the untrusted zone tend to act as clients vis-a-vis the client-server model of network communication, or at least they do for their communication with hosts outside their zone. I propose that this principle is an important (although perhaps implicit) motivation behind the screened subnet architecture.

4 By 'server-side exploitation', I refer to an exploitation of a server by a client. By 'client-side exploitation', I refer to the inverse. For a useful discussion of the distinction, in the context of network security monitoring, see [9].

3. From a principle to a strategy

I am intrigued by the idea of using this *attack vector principle* to generalize from the aforementioned tactic inherent in the screened subnet architecture. That is to say, I want to propose a generalization of the tactic of separating (into zones) *clients* that communicate with the Internet, from *servers* that communicate with the Internet. I want to suggest that analogous benefits are conferred by *generally* separating – into zones – clients and the servers they communicate with. Below I will describe and defend this general strategy for the placement of clients and servers (in an internetwork). Let me begin by defining a couple of terms that will assume importance in my discussion: the notions of a 'screening zone' and a 'traffic flow':

Screening zones

A screening zone[5] can be viewed as a model-theoretic notion. The domain of interest is the set of all hosts in some internetwork. Within that internetwork, there are one or more screening zones, where a screening zone is (potentially) any subset of hosts in that internetwork. One possible screening zone, therefore, is all hosts in the internetwork. (This would be an internetwork without any screening.) Another possibility is all hosts on some (but not all) networks on that internetwork. (A case in point here would be a screening zone corresponding to a single network within a larger internetwork.) Yet another possibility is a group of hosts that constitute a proper subset of the hosts on a particular network. In short:

- *A screening zone is partly constituted by a grouping of hosts in an internetwork.*

I say 'partly constituted' because there is more to a screening zone than a grouping of hosts. In addition to this, a screening zone is constituted by the fact that, although communication *within* a screening zone is not subject to a network access control rule set (such as a packet filtering rule set), communication *between* screening zones *is* subject to such network access control. In short:

Definition of a screening zone

- *A screening zone is a grouping of hosts in an internetwork, such that traffic to or from a host in that group is subject to network access control* ***if and only if*** *that communication is with a host in a* ***different*** *group.*

With this notion of a screening zone, we have a method of representing firewall architectures in abstract form. In particular, we abstract from any *network* topology, since we are not interested in the number of network segments that constitute a particular topology. (Indeed, there is, on this model, no distinction between a screening router, a screening bridge, or a switch module performing a similar function.) Moreover, we abstract from the *number* of screening devices employed in any particular architecture, for we are only interested in *whether* traffic between hosts is screened, and thus we are not interested in *what* device (or series of devices) do that screening.

5 I use the term 'screening zone', as opposed to 'filtering zone', in order to make it clear that I am not restricting attention to packet filters traditionally understood. For example, the strategy is also applicable to firewalls that do deeper inspection (e.g., so-called 'layer 7 filtering'). Regardless, I concentrate attention on packet filtering traditionally understood, for it is here that the benefits of this strategy are most stark.

It will become clearer further on just what advantage is gained by this abstract model, but it is worth noting here that the decision to collapse multiple screening devices is intended to highlight the fact that I am discussing firewall architecture strategies not concerned with host security, even the security of the screening devices themselves. For example, consider the following two architectures (where 'architecture' is understood in the traditional way):

Recall that UZ, DMZ, and TZ stand for Untrusted Zone, De-Militarized Zone, and Trusted Zone, respectively, and let R*x* stand for a 'firewall' appliance, such as a packet filter:

(1)

```
|--------R1--------R2--------|
   UZ       DMZ       TZ
```

(2)

```
|--------R1--------|
   UZ   |   TZ
        |
        |DMZ
        |
       ---
```

The advantage of (1) over (2) is simply that, with (1), there is no single point of failure for access to the trusted zone, whereas with (2), R1 is just such a single point of failure.

On my abstract model, it would be possible to represent these two architectures identically, with something like:

```
|--------+--------|
   UZ  |   TZ
       |
       |DMZ
       |
      ---
```

where the plus symbol, '+', represents only the fact that traffic is passing between zones (and is therefore screened).

A consequence of this is that, with this form of diagramming, all firewall architectures could be 'collapsed' into a 'hub and spokes' diagram, of varying numbers of 'spokes', depending on the number of screening zones.

Traffic Flows

In addition to 'screening zones', I would also like to define the notion of a 'traffic flow'. Traffic *flows* (as I feel like putting it) from clients to servers. Moreover, as I am using this term, when I talk of a 'traffic flow', I will refer to communication initiated from a client in one screening zone, to a server in another screening zone. That is to say, for my purposes, a traffic flow is essentially traffic that crosses screening zones: a 'traffic flow' is any inter-zone traffic that follows the client-server model of network communication.

Definition of a traffic flow

- *A traffic flow is any client-server communication between screening zones. A traffic flow is directional: the traffic* **flows** *from client to server, from one zone to another zone.*

In the context of the screened subnet architecture we've been discussing, we have two traffic flows: (1) traffic *flows* from the Internet to the perimeter network, and (2) traffic *flows* from the organization's internal internetwork to the Internet. Applying the notion of a screening zone to this screened subnet architecture, we can say that (1) traffic flows from an untrusted screening zone to a demilitarized screening zone, and (2) traffic flows from a trusted screening zone to an untrusted screening zone. (But, e.g., traffic does *not* flow from the demilitarized screening zone to the trusted screening zone.)

What I want to argue here is that it is in the interests of network security that, during firewall design, we pay particular attention to the roles of hosts as clients or servers (or both, as is the case with a proxy or relay: more on this later), when considering strategies for segregating hosts in an internetwork.[6] Resisting the attempt to assert qualifications, that strategy could best be phrased like this:

Client-server segregation strategy

- *In so far as possible, and other things being equal, separate clients from the servers they communicate with, and furthermore, organize hosts into screening zones according to traffic flows.*

I will eventually argue that this strategy is a consequence of my attack vector principle stated above, but let me begin by clarifying the strategy. I mentioned earlier that a specialized instance of this strategy is an implicit motivation behind the screened subnet architecture. It is implicit elsewhere as well: consider the well-established 'split-split' DNS strategy for separating DNS data and functions.[7]

The 'split-split' DNS strategy says that, not only should we separate internal from publicly accessed DNS data (i.e., that we should 'split' our DNS), but furthermore, we should split the *functions* of DNS advertising and DNS resolution (hence the label, 'split-split'). There are all kinds of well documented reasons for hosting DNS advertising and DNS resolution on separate hosts, and certainly it is a best practice to separate, on to different screening zones, an organization's internal DNS advertiser and its public DNS advertiser. However, it is not (as far as I know) a commonly expressed best-practice firewall design to separate, on to different screening zones, DNS advertising and DNS resolution. I argue that such separation – and others like it – is both an application of my strategy, and has distinct benefits for network security.

6 I refer 'hosts' *roles*' as opposed to just 'hosts' because hosts *per se* do not always fall neatly into the categories of 'client' and 'server' (or proxy).
7 For elaboration on the benefits of split-split DNS, see [10].

Consider this proposal for further 'splitting' DNS in light of the distinction between client- and server-side exploitation. Server-side exploitation requires a connection initiated from an attacking client to the server's listening port. Client-side exploitation requires that a client initiates a connection to an attacking server. As I noted above, it is clear that, other things being equal, server-side exploitation is the greater threat, since the server – and especially a public server – has no choice about what client connects to it, whereas client-side exploitation often depends upon the client deciding to initiate a connection to an exploiting server. (Server-side exploitation is akin to hunting, and client-side exploitation is akin to fishing.)

Let us apply this to the distinction between DNS qua advertiser and DNS qua resolver. Suppose that the DNS advertiser services arbitrary DNS clients on the Internet, and that the DNS resolver services requests from internal clients, for the purposes of recursive resolution and caching. Suppose further that the DNS server software is vulnerable to exploitation, and that this exploitation is possible either client- or server-side. (I am setting things up this way to deliberately make 'other things equal'.)

Generally speaking, therefore, it is far more likely that this DNS software will be exploited server-side, simply because anybody can take a shot at it (and also, it should be noted, because those who can take a shot at it are 'untrusted'). This by itself is good reason to separate the DNS advertiser's zone from the DNS resolver's zone: *we separate the more highly threatened host from the less threatened host*. Yet careful attention to the client-server model yields other reasons for segregating these two roles. Another reason is to mitigate the effects of host compromise. Part of the purpose of a DMZ is to protect internal hosts in the event that a publicly accessible bastion host is compromised. That is to say, we put such bastion hosts on their own zone in order to protect internal hosts – via packet filtering, for example – from attacks launched from that compromised host. This concern also applies to attacks launched from the compromised DNS advertiser to the DNS resolver. If the advertiser is compromised, and is in the same zone as the resolver, there are at least two attack vectors available to our attacker: client-side, and server-side. (A DNS resolver is both a client and a server.) But if these two hosts are segregated on to different zones, such that client connections from the DNS advertiser to the DNS resolver are forbidden via packet filtering, the only attack vector available is client-side exploitation of the resolver. (Here I am supposing that client connection initiations from the resolver to the advertiser are not prohibited via packet filtering, which is plausible, since the resolver may need to resolve for domains advertised by the advertiser. If this is indeed prohibited then the benefits of separation are even greater.)

One might object here. Does my argument not apply equally to separating servers from one another, or to separating clients from one another, or even to separating proxies from one another? If so, the recommendation is not peculiar to the *distinctions* between clients, servers, and proxies.

This is not a valid objection. Take the example of an SMTP server and an IMAP server, hosted on separate machines. Now suppose the SMTP server is compromised. If these two servers are in the same zone, then the attacker will have one attack vector: server-side; and if these two servers are separated onto different screening zones, then there will be only one, and indeed the very same attack vector: server-side. Why? Because even if client connections, from the compromised server's screening zone, to the other server's screening zone are prohibited, the attacker can still initiate client connections from the untrusted zone. (Remember, we are assuming these servers are accessible from the untrusted zone.) So in the case of separating various servers with identical traffic flows, we gained no advantage. Whereas when we separated proxies from servers, we did gain advantage.

From a real-world point of view, the previous paragraph over-states its case. It is worth emphasizing here that I am purposely simplifying the notion of an 'attack vector'. That is to say, I am restricting attention to two very general categories of attack: client-side and server-side. I acknowledge that this categorization is not all-inclusive. I make no mention of the fact that placing two different servers on different screening zones will help if those servers are not properly locked down. For example, the administrator may have failed to disable unnecessary service ports (and failed to filter them with a host based firewall). What is more, perhaps these hosts are not protected against other types of attacks, such as ARP cache poisoning, or ICMP redirect attacks (supposing they are on the same network segment). And so on. As I say, I acknowledge this, and indeed it would be advantageous to place *every* host in its own screening zone. But this is not feasible.

What I am suggesting here regarding DNS is applicable in general to the distinctions between clients, proxies, and servers. (For a DNS resolver is really a kind of proxy.) For example, it is equally applicable to the distinction between an SMTP client, a pure SMTP server, and an SMTP relay host (which again is both a client and a server). Or again, it is equally applicable to the distinction between an HTTP client, an HTTP server, and a HTTP proxy.

4. Demonstrating the strategy

The extra protection afforded by such network segregation can be demonstrated by imagining the chain of attacks required for various firewall architectures. Let me begin this demonstration by first formally quantifying the attack vector principle, and its exception, formulated in Section 2. above.

To formalize, I will 'weigh' the 'difficulty level' of exploitation in the following manner. *Other things being equal,* I assume that:

- a *server-side exploit* has a difficulty level of 1, and
- a *client-side exploit* normally has a difficulty level of 2, *unless*
- the client is exploited by a server it is configured to use habitually. In this case only, a *client-side* exploitation has a difficulty level of 1.[8]

These numbers are arbitrary. They are intended to signify only that a server-side exploit is easier than a client-side exploit, other things being equal, and with the one exception.

Let us now imagine that an internetwork has one or more clients, one or more servers, and one or more proxies, each associated with one or more application layer protocols. Let us also imagine that, if there are proxies, clients in the trusted zone initiate communication with proxies only; otherwise clients in the trusted zone initiate communication with (pure) servers. Furthermore, proxies initiate communication with servers only, and servers don't *initiate* communication at all (since they are servers). Let us further imagine that the threat goal of a particular attacker is to gain access to a trusted zone, launching that attack, at least initially, from an untrusted zone. (These are important, but not the only possible threat goal and threat origin. Similar demonstrations could be given for threats from and

8 To fix ideas, in my demonstration (below) my example of 'a server that a client is configured to use habitually' will be a proxy server. This is a common case in point, but the reader should bear in mind that this is generalizable to any server that a client is configured to use habitually.

against other zones.) And for the sake of separating issues of firewall architecture from issues of host and network security generally, I will assume that compromising hosts on an 'owned' screening zone is trivial, i.e., that it has a difficulty level of 0.

Given these assumptions, I would like to consider a series of simple firewall architectures. The series will begin with a very simple architecture, and they will increase stepwise in complexity. I would like to show how adherence to my firewall architecture strategy confers benefits in terms of preventing the attacker from achieving her goal.

(I) Attack vectors for a two zone architecture (with or without proxies)

Imagine an internetwork with just two screening zones: an untrusted zone (UZ) and a trusted zone (TZ), e.g., the Internet and an organization's internetwork. In this scenario, any server- or client-side exploit of a host on that TZ will achieve the attacker's goal. That is to say, there are two attack vectors, and each consists of a single step:

Either

(a) the attacker server-side exploits a server in the TZ, from a client in the UZ.
Difficulty level = 1

or

(b) the attacker client-side exploits a proxy (if any) or client on the TZ, from a server in the UZ.
Difficulty level = 2

(II) Attack vectors for a three zone architecture (without proxies)

Compare now an architecture with three screening zones: an UZ, a demilitarized zone (DMZ), and a TZ. Without the use of proxies, we have two attack vectors, the first consisting of a single step, and the second being a two step process:

Either

(a) the attacker client-side exploits a client in the TZ, from a server in the UZ.
Difficulty level = 2

or

(b) the attacker server-side exploits a server in the DMZ, and *then* client-side exploits a client in the UZ. *Difficulty level = 1 +2 = 3*

We see here that, with the introduction of the extra zone – the DMZ, designed to separate off bastion hosts – we achieve some improvement over architecture (I).[9] Although there are still two attack vectors, the difficulty levels have gone from 1 & 2 on (I), to 2 & 3 on (II).

If, on the other hand, an organization does make use of proxies, so as to avoid clients in the TZ having to connect directly to servers on the UZ, then, with the same three zone architecture as above, we have two choices for the placement of proxies. (Public *servers*, we'll assume, are always placed in the DMZ, and internal clients, in the TZ.) We could place a proxy in the DMZ with the bastion hosts, or we could place it in the TZ with the clients. Let us take each case in turn.

9 No surprise here. This is really merely a formalization of our previous consideration of the screened subnet architecture in Section 2. above.

(III) Attack vectors for a three zone architecture (with proxies in the TZ)

Either

(a) the attacker client-side exploits a proxy on the TZ. *Difficulty level = 2*

or

(b) the attacker server-side exploits a server on the DMZ, and then client-side exploits a proxy on the TZ. *Difficulty level = 1 + 2 = 3*

We see here that the mere introduction of the proxy does not itself put the attacker at any disadvantage over the previous architecture, at least not from our limited point of view. Rather, it affords advantage only if it is separated from its clients:

(IV) Attack vectors for a three zone architecture (with proxies in the DMZ)

If the proxy is placed on the DMZ, then traffic 'flows' in three directions: from clients in the UZ to servers in the DMZ; from clients in the TZ to proxies in the DMZ; and from proxies in the DMZ to servers, either in the DMZ or in the UZ. In this case, compromise will consist of one of the two following two-step attack vectors. (As previously noted, please bear in mind that, for the sake of my abstraction from host security, I assume that exploitation on an 'owned' screening zone is trivial, i.e. has a difficulty level of zero. Thus I ignore the step of exploiting the proxy on the already owned DMZ network.)

Either

(a) the attacker server-side exploits a server in the DMZ, from a client in the UZ, and then, from a proxy in the DMZ, client-side exploits a client in the TZ. *Difficulty level = 1+2 = 3.*

or

(b) the attacker client-side exploits a proxy on the DMZ, and then client-side exploits a client in in the TZ. *Difficulty level = 2+1 = 3.*

Here we see an improvement over all architectures considered so far and, although the scenarios so far have been successive improvements, none so far have *consistently* applied my security strategy of *separating clients from the servers they communicate with*. (On the other hand, they indeed have successively come closer to implementing my strategy.) For in all the cases so far considered, one of these three types of hosts has shared a screening zone with another type. Let's now move to a consistent application of this strategy:

I suggested earlier that, by applying my strategy to DNS services, we have good reason for splitting DNS traffic between *four* screening zones. This notion is borne out by considering the attack vectors for a four zone scenario. This four zone scenario includes a *proxy* zone (PZ), in addition to the UT, TZ, and DMZ. Proxies, in this case, are hosts that proxy traffic from the TZ to either the UZ or the DMZ (or both). The example we discussed above is the DNS resolver. Another example is an SMTP relay. Yet another is an HTTP proxy. Let us assume, for this scenario, that proxies in the PZ must occasionally connect to servers in the DMZ. This is plausible. For example, a web client on the TZ might occasionally use the web proxy on the PZ to access a web server on the DMZ.

(IV) Attack vectors for a four zone architecture (TZ, UZ, DMZ, PZ)

Either

(a) The attacker client-side exploits a proxy on the PZ, from the UZ, and then client-side exploits a client on the TZ. *Difficulty level = 2+1 = 3*

or

(b) The attacker server-side exploits a server on the DMZ, then client-side exploits a proxy on the PZ, and finally client-side exploits a client on the TZ. *Difficulty level = 1 + 2 + 1 = 4*

Clearly this is the most advantageous scenario so far. Indeed, this evaluation does not do justice to all the advantages of this strategy.

For, so far, I have progressed from architectures which failed to completely separate clients from servers from proxies, to an architecture which did conform to this principle of separation. Yet recall that my strategy was not merely to separate clients from servers from proxies, but to separate them *according to traffic flows*. Although we *did* separate them according to traffic flows in (IV) above, this is perhaps not immediately evident, since there is not, in the above architecture, any effective difference between mere separation, and separation according to traffic flows. Let me clarify by adding another role for proxies to the previous, four zone architecture.

(V) Attack vectors for a five zone architecture (TZ, UZ, DMZ, PZ, RPZ)

To fix ideas, let us restrict attention to HTTP. Imagine that we have a web client in the TZ, a web client in the UZ, a web server in the DMZ, a web proxy in the PZ (used by clients in the TZ), and a *reverse* web proxy in the reverse proxy zone (RPZ). This reverse proxy, let us imagine, is used by clients in the UZ, to indirectly connect to servers in the DMZ.

In this case, we have two proxies, but they are in different zones. The reason they are in different zones is that they have different traffic flows. For the proxy in the PZ, traffic flows from the TZ to the PZ, and also, traffic flows from the PZ to the UZ. Whereas for the proxy in the RPZ, traffic flows from the UZ to the RPZ, and from the RPZ to the DMZ. I won't do the calculation, but you can guess that using five zones here, as opposed to four (with the proxy and the reverse proxy in the same zone), adds difficulty to the attacker's goal.

One last point is worth making in this connection. Recall that we considered only one attack goal: access to the trusted zone. This was merely for the purposes of illustration. Consider, as but one further example, access to the proxy zone. If proxies and servers resided in the same screening zone, then a server-side compromise of a server affords the attacker unscreened access to a proxy. Likewise, if reverse proxies and servers resided in the same screening zone, then a server-side compromise of a reverse proxy affords the attacker unscreened access to the server.

The strategy I am proposing here is, I think, really just a specialized version of the strategy of least access. Consider: untrusted clients do not need client access to proxies used by trusted clients; proxies do not need client access to trusted clients; bastion hosts do not need client access to anything; nothing needs client access to pure clients; and so on. Moreover, consistent application of the strategy ought to result in the fact that intra-zone communication is completely unnecessary, or at least, this is so in an ideal world where hosts divided categorically into clients, servers, and proxies. For example, in architecture (V) above, proxies do not need to contact other proxies; clients do not need to contact other clients; servers do not need to contact other servers.[10] And this fact points to yet another advantage of this strategy: to the extent that this strategy is applied consistently, private VLANs could – to that extent – be employed to forbid intra-zone communication completely.[11]

10 It is not necessarily true that proxies do not need to contact other proxies. But if they *do*, then this strategy implies that they should be in different screening zones.

11 In the event that two hosts need to engage in intra-zone communication, private VLANs are useless for *restricting* that access, since private VLANs either forbid or allow traffic without discrimination: they cannot be used to *screen* that traffic.

References

[1] Bellovin, Steven M., 'Shifting the Odds: writing more secure software' (1996), p. 9. Available at: http://www.cs.columbia.edu/~smb/talks/odds.ps

[2] IEEE Standard 610.12, quoted in Wikipedia. Available at: http://en.wikipedia.org/wiki/Software_engineering

[3] Bellovin, Steven M., 'Firewalls are Necessary' (1994). Available at: www.cs.columbia.edu/~smb/talks/firewalls.ps

[4] Alberts, C., and Dorofee, A., *Managing Information Security Risks: The OCTAVE Approach*, (Addison Wesley Professional, 2002), section 7.1

[5] Zwicky, E. D. et. al., *Building Internet Firewalls*, 2nd Edition (O'Reilly, 2000), section 6.3.

[6] Zwicky, p. 241.

[7] ibid.

[8] ibid.

[9] Bejtlich, R., *Extrusion Detection: Security Monitoring for Internal Intrusions* (Addison Wesley, 2006), Chapter 3.

[10] Stewart, J., *DNS Cache Poisoning – The Next Generation*. Available at: http://www.lurhq.com/dnscache.pdf