# Think Security

Understanding a collection of
Useful Security Practices.

## Useful Security Practices

1. Think in zones.
2. Create chokepoints.
3. Layer security.
4. Work in stillness.
5. Understand relational security.
6. Understand secretless security.
7. Divide responsibility.
8. Fail securely.

## 1. Think In Zones

*Zoning* is the process by which we **define** and **isolate** different subjects and objects based on their unique security.

Although primarily associated with network-specific security, zoning can also be applied to applications, physical areas and (even) employee interactions.

## Defining a Zone

*A logical grouping of resources that have a similar security profile.*

This refers to the *logical grouping* of objects that have similar:

- risks.
- trust levels.
- exposures.
- policies.
- security needs.

## The Three Zones of Trust

- **Trusted zone (internal)** - contains the most valuable and sensitive resources.  This zone is under the control of, and governed by the policies of, the organization.
- **Untrusted zone (external)** - no direct control, no application of policies.
- **Semi-Trusted zone** - still controlled by, and under the policies of, the organization However, the resources here are more directly exposed.  Therefore, they are *more vulnerable*.

## Example Zone Resources

*Trusted Zone:*

- **Network**: internal servers and workstations.
- **Application**: trusted application code and databases.
- **Physical**: server rooms/communications cabinets.

*Untrusted Zone:*

- **Network**: the Internet and dial-up telephone lines.
- **Application**: end-users, external services/devices.
- **Physical**: everything outside the office/campus.

*Semi-Trusted Zone:*

- **Network**: external Web/Mail/DNS/FTP services.
- **Application**: untrusted 3rd party code (applets).
- **Physical**: lobbies, waiting rooms, public access areas.

## 2. Create Chokepoints

The *chokepoint* is a key security tool.

*A tight area wherein all inbound and outbound access is forced to traverse.*

**Examples**: draw-bridge, front-door, firewalls, proxies, IDSs

It is much easier to secure something if there is only one way in and one way out.

## Chokepoint Advantages

- **Security Focus**: focuses attention, enhances security, less taxing on our resources.
- **Ease of Monitoring**: it is much easier to spot an attacker if there is only one place to look!
- **Ease of Control**: easier to implement security mechanisms when everything is localized.
- **Cost Reduction**: centralized security is always cheaper than distributed solutions.
- **Exposure Reduction**: centralized security (if done well) is less prone to error/exposure.

## The Network Chokepoint

Most common type: used all the time.

Typically, all LAN traffic is directed through a single inbound/outbound access point where it is *filtered* and/or *monitored* for adherence to the organizations security policies.

Common traffic through the network chokepoint includes: Internet comms., VPN traffic, vendor/partner/customer WAN comms., wireless traffic.

## The Application Chokepoint

Providing a **single authorization point for access** to application services, which allows for filtering and monitoring.

For example: the idea of ``domains'' within Windows 2000 (XP Server).

Single sign-on/portal applications.

## The Social Chokepoint

Employees, executives, partners and customers can be exposed to attack.

This can lead to a **social engineering attack**.

Rather than introducing draconian measures, social chokepoints can be introduced via appropriate training measures and mechanisms

## Applying Chokepoints

1. Identify all access points to a particular resource or related set of resources.
2. Consolidate all such access points through a single security object.
3. Enforce tight controls, monitoring and redundancy on that security object.
4. Establish a policy for future access points, stating that they must be filtered through an approved chokepoint.
5. Continue to test and scan for new access points that do not filter through a chokepoint.

## Chokepoint Downside

Putting all your eggs into the one basket may not be the best strategy.

The introduction of a single-point-of-failure needs to be considered.

The introduction of a redundant/fail-over chokepoint can help here (sometimes).

## 3. Layer Security

**An undeniable truth**: *every significant application, server, router and firewall on the market harbours some sort of vulnerability*.

Also: it is possible that devices can be misconfigured, unmonitored and improperly maintained.

Nothing can be 100% secure, ever!

## The Layering Solution

Consider at least three layers of security:

- **Internal**: controls applied directly to protected internal objects.
- **Middle**: primary security devices.
- **External**: the front-line of defense against intruders.

| Scenario | External Layer | Middle Layer | Internal Layer |
|---|---|---|---|
| Perimeter network | Screening router | Firewall/IDS | Server-based controls |
| Physical security | External gate | Front door | Internal locked cabinet |

## 4. Work in Stillness

*It is all but impossible to ``hear'' the enemy if there is excessive surrounding noise to confuse us.*

For example: logs, although inherently useful, are next to useless if a cracker's attack is *hidden* within millions of other log entries.

There must first be **stillness**, if we are to make any sense of the **noise** an attack makes.

## Creating Stillness

The ``trick'' is to tune each device to only report that which is of interest.

- Study the logs and alerting features of each device.
- Initially set the logging activity to be sensitive.
- Adjust settings to filter out ``normal'' activity.
- Anything else that appears can be treated as suspicious/malicious (but *may* not be), and can be blocked/filtered.  Remember to document everything.
- Logging/monitoring is then restarted *in stillness*.

## 5. Understand Relational Security

The security of any object is dependent on the security of its *related objects*, and if we fail to see these relationships, we will be unable to properly address security.

Thus, we have *Relational Security.*

It is important to realize that **chains of relationships** often exist within IT environments.

## Three Important Chains

1. **Vulnerability Inheritance**: if a highly secure system is ``connected'' to a poorly secured system, then they are **both** poorly secured.

2. **Chained Values and Risk**: risks to a single object must take into consideration its relationships to other objects (shared risk).

3. **Chains of Trust**: when trust is extended, exposure must be considered.

## Exploiting Chains

Attackers are continually thinking in terms of chains and are looking for the easiest and quickest method of gaining access to an object.

Common paths of entry (exploitable chains) are:

- Direct or dial-up Internet connections.
- E-mail (networked communications).
- Partner, vendor and consultant network connections.
- Modems, remote access and VPN connections.
- Removable media (all types).
- Employees that work on multiple systems (accounts).
- New computers configured outside your environment.

## Chained Privileges

Consideration should not only be given to the subject to which we are granting access, but also to the other subjects that have access to that subject.  If X grants access to Y, and Y is accessible by Z, then X is *somewhat accessible* by Z as well.

Such relationships *must be considered* when making a security decision.

## Chains: Key Points

Everything is connected to everything else.

Never consider any system safe simply because it does not connect directly to the Internet.

Most organizations have a lack of concern when attaching their own networks to the networks of ``trusted'' partners and vendors.  **Question**: are we willing to trust all of their connections as well?  And if so, by how much?

## 6. Understand Secretless Security

Basic security often relies on some form of **secrecy**.

Unfortunately, secrets are very hard to keep!

Surprisingly, the best security solutions are those that rely *as little as possible* on secrecy for protection.

It should always be assumed that all secrets are going to be discovered, sooner or later.

So, better to rely on *secretless security*.

## Example of Secretless Security

**Open Encryption Algorithms**: the less secret these are, the better they are.  Secret algorithms are not secure, nor secureable.

**Open Security Applications**: ``Many eyes make all bugs shallow'', the more open the source code, the more secure it can be made.

**Secretless Authorization**: avoid the use of secrets for gaining entry.  That is, *no more passwords*. Examples: fingerprints, retinal scans, etc.

## Passwords: A Necessary Evil

Keeping a large number of passwords secret is **extremely difficult** if not **impossible** for large organizations

However, passwords do appear to be the ``cheap'' option, so they are very, very popular.

**A good article**: ``*Psst ... I Know Your Password*'' from: `http://zdnet.com.com/2100-1105-920092.html`

## 7. Divide Responsibility

Don't put all your eggs in the one basket.

Never, ever, ever assign all security responsibilities to one employee, one system or one process.

Everyone should have to request access, be required to authenticate as well as have their actions restricted and logged just like everyone else.

Anyone not restricted is a security threat to the IT environment.

## Practicing Division of Responsibilities

**Maintain redundant staff**: dual-training, backup employees, shadows, buddy-systems, etc.

**Monitor everyone equally**: universally enforce all security measures.  Enforce monitoring for all.

**Enforce security rules on everyone equally**: no one should be allowed to bypass any security measure.

Always follow **layered security practices** (as discussed earlier).

## 8. Fail Securely

*Attackers commonly use exploits that cause services to fail due to unexpected events.*

When services crash, security holes can open-up.

All too often, systems are configured that, in the event of a crash, all security is bypassed: ``Just keep things running *at all costs!!!*''.

Services should be configured to **fail securely**.

## Practicing Failing Securely

Many attacks are designed to disable a firewall or network device, wait for an unsecured failover to kick-in, then take advantage of the failover's weaker security.  This should be guarded against.

Potential failure scenarios need to be considered **before** any new implementation occurs.

Redundant, back-up, failover services need to be secured, too, not just the primary services!

## Summary

1. Think in zones.
2. Create chokepoints.
3. Layer security.
4. Work in stillness.
5. Understand relational security.
6. Understand secretless security.
7. Divide responsibility.
8. Fail securely.