9.2 Single Access Point

If you need to provide external access to a system, but want to protect it from misuse or damage, define a single access point that grants or denies entry to the system after checking the client requiring access. The single access point is easy to apply, defines a clear entry point to the system, and can be assessed when implementing the desired security policy.

Also Known As

One Way In. Concrete implementations are called Login Window, Guard Door, or Validation Screen.

Example

Consider a small medieval village. It consists of a group of houses in close proximity. While there is little economic prosperity, there is little value in protection from burglars and little interest to robbery. Security for people means that each man protects his own belongings. Each man spends time building weapons and training in their use so that he can defend his family.

Somehow economy prospers (we do not speculate why and how), more and more people move to the village, and more and more value is accumulated within the village. However, since the people need more time for their prospering businesses, they have less time to spend practicing defence. Their level of protection becomes lower, while the threat from burglars grows. In addition, visitors must convince every individual shopkeeper that they are valuable customers instead of thieves before they can actually conduct business with them. The village dwellers wonder how they can simplify protection, so that everybody no longer needs to deal with it many times a day.

Context

You need to provide access to a system for external clients. You need to ensure the system is not misused or damaged by such clients.

Problem

Whenever a system is used by an external client such as a user, the system's integrity is in danger. Often such systems require some security property, like protection from misuse or damage. One means is to check every interaction with an external client to determine whether it is authorized. When the system has a non-trivial inner structure and consists of multiple parts or subsystems, an external interaction of the system can result in many different interactions of the client with the individual parts of the system. Checking each of these sub-interactions is required to protect all the parts, and thus the whole system. First, implementing all these checks can be a burden: second, if the same information has to be presented over and over, these checks can hinder performance and annoy a user: third, assessing the correct implementation of the overall security policy is hard, because of its complexity.

In addition it is a good practise to have a clearly-defined entry point to a system, as you are used to with the main entrance to a building. Such a prominent and well-known entry point makes using the system easier, because we do not need to spend time searching for the entrance.

The solution to this problem must resolve the following forces:

- You need to provide access to a system to make it usable.
- In a complex interconnected world, no system is an island.

- Most systems exhibit a non-trivial structure and are constructed from sub-systems that also need protection.
- Many entry points to a system reduce security, because the additional complexity makes it easier to bypass controls.
- Multiple entry points can have duplicate code for the same kind of checking.
- Repeated checks annoy clients or slow down the system.
- Uniform access to a system can lower its usability if different situations really require different means of access—for example, entering the Windows log-in password on a tablet PC is annoying if no actual keyboard is present on which to type it.
- Uniform access to a system is easier to control.

Solution

Define a single access point for clients using the system. At this access point you can check the legitimacy of the client according to your defined policy. Once clients passed the access point, they are free to use the system from that point on.

Protect the rest of the system's boundary, so that no circumvention of the single access point is possible. The inhabitants of our medieval village build a wall to serve as such a passive boundary protection: a computer operating system denies all activity from anonymous users not logged in.

Make the single access point prominent, so that it is easy to find and absolutely obvious where to enter the system. Nothing is more annoying than circling the city wall looking for a gate. A computer operating system usually shows a log-in window or prompt when no user is currently logged in.

If auditing is required, the single access point can record which clients entered the system and when. It might also record the termination of an client's use of the system.

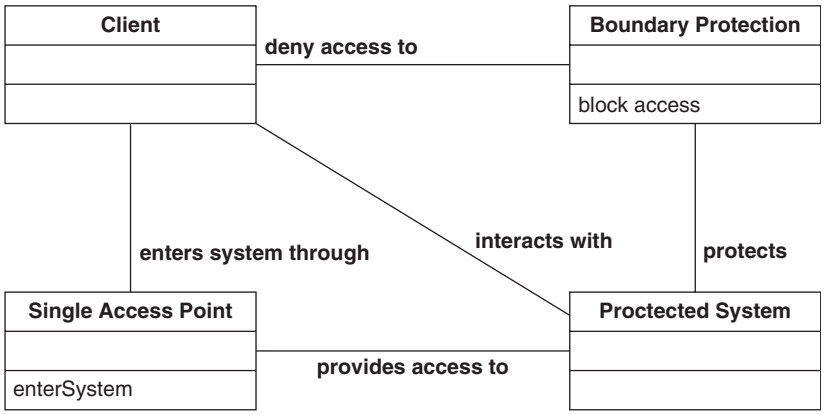
This pattern applies to many levels of abstraction and technology. It further might apply within a more complex system to the system itself as well as its subsystems, which in turn can have additional single access points.

The above description is very generic: here is a list of some concrete examples:

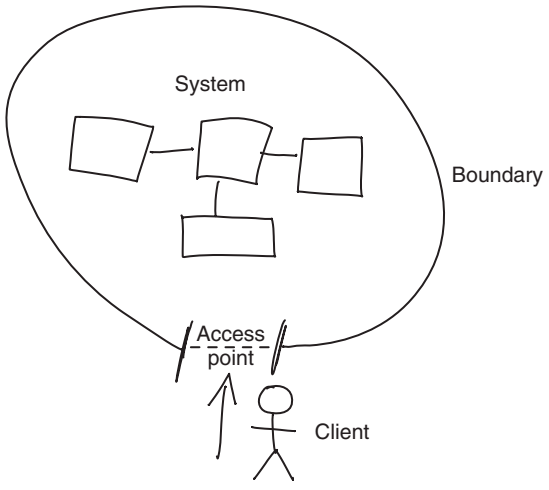
- Function entry point with precondition check (for example in Eiffel)
- Windows operating system log-in screen
- Chapter 12, *Firewall Architectures*
- A security guard in an office building or military base

Structure

The single access point can be represented by the following UML diagram.

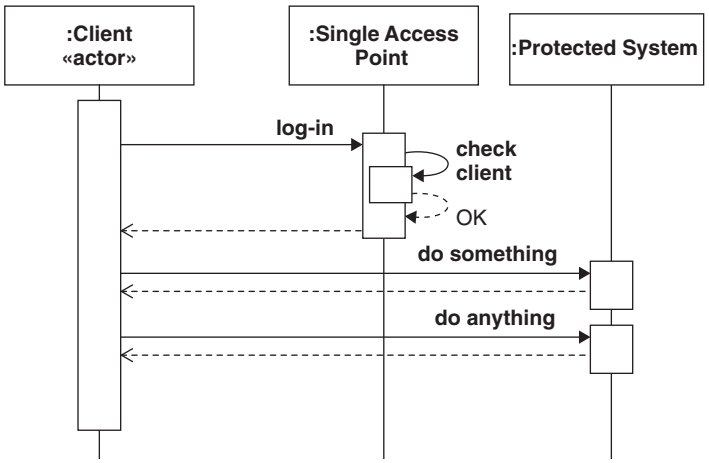


However it is more intuitive to presented it as shown in the accompanying sketch, since it is hard to show the boundary protection of the protected system. Boundary protection is essential to make the single access point efficient in checking clients and hindering intruders to access the system.



Dynamics

The sequence diagram illustrates a regular scenario of an client entering the system. The client logs in at the single access point and then uses the protected system. The passive protection given by the boundary (the city wall) cannot be shown here.



Implementation

To implement the SINGLE ACCESS POINT (279), several tasks are required:

1. *Define your security policy for the system at hand.* Before you start securing your system, you should know what you secure and why. Apply the patterns from this book to obtain the security requirements for the system to be protected. The security policy must contain the trust relationship between the internal subsystems. All of them need to trust the single access point and also each other. Even if such trust can be established, it might be wise to apply the Defence in Depth security principle (see Chapter 15, *Supplementary Concepts*) for extra-sensitive subsystems.
2. *Define a prominent or well-known position for the single access point, or make it transparent for its legitimate users.* Christopher Alexander's MAIN ENTRANCE [AIS+77] gives some guideline about where to place your main entrance, which SINGLE ACCESS POINT (279) definitely is. He writes, 'Therefore: place the main entrance of the building at a point where it can be seen immediately from the main avenues of approach and give it a bold, visible shape which stands out in front of the building.' Microsoft Window's classic

log-in screen with its 'Press ALT-CTRL-DEL to log in' is definitely not following the essence of that rule.

Another option is to make the single access point invisible, but impossible to circumvent. This makes the access transparent for clients, but nevertheless allows the system to be protected. The firewall patterns in this book are examples where the single access point is made transparent for legitimate uses but impassable for intruders.

3. *Optionally implement the entry check at the single access point*. If your system's security policy requires authentication and authorization, the single access point can play a major role in implementing it easily. A typical software system will provide a log-in window for the user to provide an identifier and password. If both match with corresponding stored values, the user is allowed to use the system. See the patterns in Chapter 7, *Identification and Authentication (I&A)*. CHECK POINT (287) shows how to make this checking flexible.

   If you apply CHECK POINT (287), you can also associate a SECURITY SESSION (297), or a so-called 'day pass,' with the client. Every client showing such a day pass is automatically trusted within the system once past the single access point. The single access point will initialize parameters and variables within the client's session to valid values on which the system can rely. In simple cases the 'day pass' can be implicit, by letting the client enter the system and trusting the boundary protection to hinder intruders.

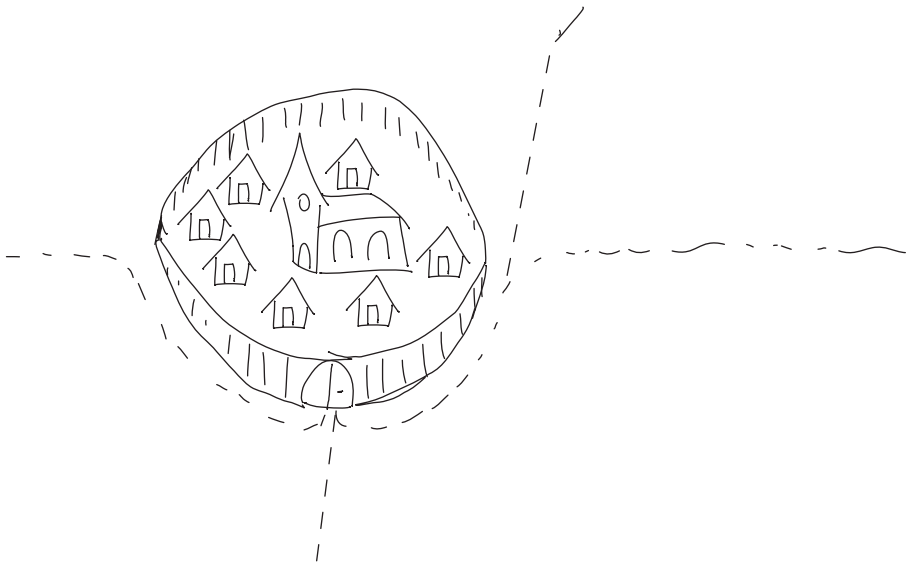
4. *Implement the system initialization at the single access point*. Some protected systems need to be initialized corresponding to their user before they can be used. For example, the Unix log-in program initializes the user's process with their user and group identities, thus enforcing correct authorization later on. It also presets environment variables, as predefined by the system, and executes an initialization script before the user can start working with the system. SECURITY SESSION (297) shows details of how to identify the user throughout their use of the system and keep their related data in a convenient place.

5. *Protect the boundary of your system*. The single access point can only be effective if you provide a closed perimeter to your system. Especially, you need to look for potential 'back doors' that have been left open. It is in the sense of SINGLE ACCESS POINT (279) not to have these. For example, when you set up a firewall (see Chapter 12), you ensure that only those ports that are actually needed are open: all other network connections are disabled.

The boundary protection can be physical, like a city wall, or built into the system itself, such as operating systems not allowing anonymous users to start processes other than via the log-in program.

Example Resolved

The village people build a wall around their dwellings, effectively making it a walled town with a gate. The wall hinders burglar's access to the town, while the gate allows customers and townspeople to enter and leave the town. A single guard at the gate is now able to protect the whole town. The townspeople acquire more time for business by paying the trusted guard.



Despite their successful city wall and city gate protecting their enlarged village, our medieval folk still have some problems with theft from their open houses. They therefore apply the security principle of Defence in Depth, and re-apply SINGLE ACCESS POINT (279) at their individual houses. Each house gets a front door that can be locked, thus protecting its inhabitants, but still allowing them and their visitors in and out. The existing stone walls of their houses already provide good boundary protection, especially because, being medieval, their windows are made from iron bars instead of glass.

Known Uses

Many operating systems, such as Mac OS, Microsoft Windows, and Unix, require a user to log into the system before it can be used. All provide either a dedicated log-in program or a prominent log-in window for the user to provide their identity and password. The boundary protection is built into the operating system by not allowing programs to be run by unauthorized or anonymous users.

Other patterns in this book, such as the firewall patterns in Chapter 12 and PROTECTION REVERSE PROXY (457), provide examples of effective single access points, in which the clients are not always users, but can be network traffic that needs entry to the protected system.

Consequences

The following benefits may be expected from applying this pattern:

- It provides a single place to go for entering the system, a clearly defined entrance for users of the system, and a single place to set up the system or application properly.
- It provides a single place to guard your system: you only need to trust your gate guards at the single access point within your system. However, applying Defence in Depth might be required to improve security further.
- The inner structure of system is simpler, because repeated authorization checks are avoided. The system trusts the single access point.
- No redundant authorization checks are required: once the access point is passed, the system trusts the client.
- It applies to many levels of abstraction.

The following potential liabilities may arise from applying this pattern:

- Having just a single access point may make the system cumbersome to use, or even completely unusable. For a medieval city, if you arrive from the wrong direction, you have to walk right round the city just to reach its gate.
- You need to trust your gatekeeper and your city wall. However, it might be easier to check the single access point instead of multiple ones. Nevertheless, the boundary protection still can be a weak point of your system.
- The single access point might need to check the client on entrance more thoroughly than is required in the concrete situation, thereby annoying the client or slowing down entrance unacceptably.
- In a complex system, several single access points might be required for subsystems.
- The single access point might become a single point of failure. If the single access point breaks down, the system might become unusable, or its security compromised.

See Also

CHECK POINT (287) and SECURITY SESSION (297) both provide details of how to implement access control based upon SINGLE ACCESS POINT (279) in a flexible and effective way.