Network Address Blacklist

(a.k.a. Client Filtering, IP Lockout)

Abstract

A network address blacklist is used to keep track of network addresses (IP addresses) that are the sources of hacking attempts and other mischief. Any requests originating from an address on the blacklist are simply ignored. Ideally, breaking attempts should be investigated and prosecuted, but there are simply too many such events to address them all. The *Network Address Blacklist* pattern represents a pragmatic alternative.

Problem

In a Web-based environment, where anonymous access is possible, there can be little to prevent an attacker from brazenly attacking a system. Even if their actions are detected, many attackers will proceed with impunity, knowing that their victims have little or no recourse. Unless the site is a commercial bank or a government agency, the FBI is too busy to investigate hacking attempts. When deterrence fails, there must be some other means of defending a site against anonymous attackers. It is not sufficient to simply rely on the strength of static defenses. Attackers that probe a site and find no resistance are likely to become more brazen.

Locking out individual accounts that appear to be under attack is an important defensive measure, but not sufficient to defend the site. Many hackers focus on network attacks and canned scripts that know nothing about individual accounts. Furthermore, a remote attacker could exploit the lockout mechanism, deliberately causing a large number of accounts to be locked out. Alternately, an attacker could choose a single weak password and then conduct an "account guessing attack" until an account using that password is discovered. The account lockout mechanism would never detect such an attack.

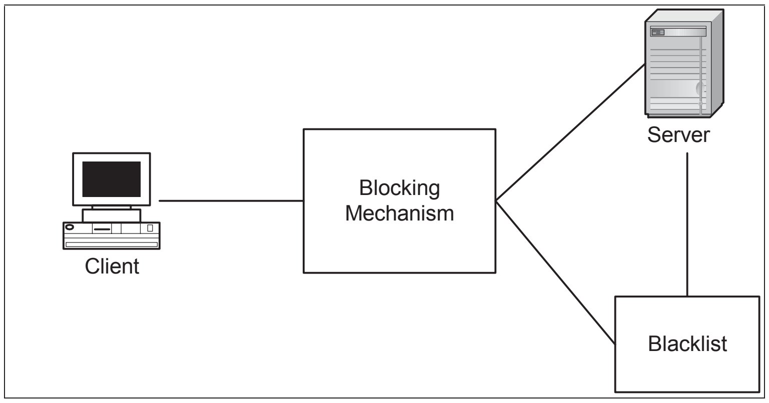
Solution

A network address blacklist mechanism maintains a list of network addresses that have exhibited inappropriate behavior. Once a given network address has exhibited enough such behavior to have obviously malicious intent, the network address will be temporarily added to a blacklist. When a request is received from a blacklisted address, it will simply be dropped on the floor.

All Web accesses are based upon TCP, which uses a handshake protocol to ensure that incoming IP addresses are not spoofed. While simple network-level denial-of-service attacks can be spoofed, more sophisticated HTTP-based malfeasance requires that the attacker be able to receive responses at the originating address. As a result, most attackers will have only have a handful of different IP addresses at their disposal. Blocking requests from a questionable IP address can effectively throttle many automated attacks.

The *Network Address Blacklist* pattern contains the folowing major elements:

- A client (with a network address) that is up to no good.
- The server, which is responsible for detecting and reporting misuse of the system.
- A blocking mechanism that is capable of intercepting client requests before they reach the server and dropping requests from blacklisted addresses.
- A blacklist that controls the blocking mechanism and responds to server requests to blacklist specific addresses.
- (Optionally) a human administrator that can manually remove addresses from the blacklist.



It is important to recognize that in many systems, the application developer will not have access to a fully automated blacklisting mechanism. In those instances, it is perfectly acceptable to allow a human administrator to act as the blacklist. The server will alert the administrator to possible misuse, and the administrator can choose to take manual measures, such as inserting a firewall or Web server access control rule that blocks that address. This is actually the most common form of network address blacklist – many application servers share either a single host, or are collocated behind a single firewall. In these instances, it would be inappropriate to allow a single application to block an address from accessing all the collocated applications.

In normal circumstances, a client that is not on the blacklist will make a benign request of the server. The blocking mechanism will let the request pass, and the server will respond with normal functionality.

When a non-blacklisted client makes a suspicious request, the server will keep a record of the network address and the nature of the request. If the request is sufficiently egregious (or the last in a sequence of requests that exceeds some predefined threshold of tolerance) the server will request that the address be blacklisted. The blacklist will configure the blocking mechanism to deny further requests from that address.

When a blacklisted client makes a request of any sort, the blocking mechanism will simply drop the request on the floor. Optionally, it may log the request for administrator audit.

After a period of inactivity from a blacklisted address, the blacklist mechanism will remove the address from the blacklist and configure the blocking mechanisms to allow further requests from that address. Alternately, this can occur because of manual administrator intervention, possibly at a user's request.

Issues

Where to Block

If at all possible, the blocking mechanism should be implemented at the network level, either on a separate firewall or using host-based network filtering. This ensures that automated attack tools will be completely blunted. It eliminates the ability of the attacker to consume resources on the Web server. And keeping this component separate makes it easier to validate its proper operation.

There are circumstances where it may be either desirable or necessary to implement the blocking mechanism within the application itself. In such a case, the application would be structured roughly similar to the figure above. When a request is received, it is first checked against the blacklist before being dispatched to the appropriate page. This approach has the advantage of being entirely self-contained, not requiring integration with other components. It has several disadvantages: it adds complexity to the application, it allows blacklisted addresses to continue to consume server resources, and it cannot prevent attacks that exploit vulnerabilities to bypass the application.

When implementing a filtering mechanism it is generally a good idea to continue to log requests by blacklisted addresses. These logs may be necessary if an attacker must be prosecuted, or succeeds in compromising security. However, it is important that the logging mechanism not place a significant burden on the system. If logging events slows the system, stops the system when the logs fill up, or presents a significant auditing burden, an attacker can continue to cause mischief even after being blacklisted.

Administration of the Mechanism

As noted above, many implementations of this pattern provide a relatively static blocking mechanism, and rely on human intervention to make changes to the addresses being blocked. This has the advantage of simplicity. More importantly, it maintains a human in the loop – a fully automated system raises the danger of the mechanism failing, or being attacked, in a manner that denies service to large numbers of legitimate users. If the system will be manned around the clock, an approach that keeps the operator in the loop is preferable. However, if it will need to be unattended for a significant amount of time, a fully automated response should be considered.

When you block a network address, you run the risk of blocking access to an entire organization behind a firewall providing network address translation. This should not dissuade you from using this approach. If legitimate users within the organization complain about being unable to reach the site, they may be in a position to track down the misbehaving party, and even punish that individual.

If the blacklist mechanism is designed to eventually time out and remove addresses from the blacklist, this timeout period should be measured in days or even weeks. However, it will have to be fine-tuned in order to ensure that the list doesn't grow so large that it adversely affects performance. As noted below, the lockout mechanism should be fairly conservative – there should be no way that an honest mistake could result in lockout. This being the case, there is no need from a usability perspective to keep the time out period low.

It may be necessary to provide an interface to manage the blacklist, so that an administrator can remove addresses manually. However, this is not an interface that customer service representatives will need to be able to use. Customers will generally not even know their IP address and customer service representatives will not understand the security implications of de-listing an address that has been detected trying to actively hack the system.

Causes for Blacklisting

Some of the sensors that can be used to determine misuse by a network address are:

- When a single network address generates a number of requests for non-existent pages, they may be performing an automated search of the system.

- If a site does not use cgi-bin scripts, any attempt to request a cgi-bin script will not have come from a legitimate user clicking on a link. Many such requests indicate that an off-the-shelf Web vulnerability scan is being performed.

- If multiple invalid login attempts are received from the same network address, it may be a user who is attempting to guess usernames and passwords. Likewise, if a reserved name such as "guest", "root", or "administrator" is used, the client is likely up to no good. However, it is possible that a legitimate user has simply forgotten their username. In this case, the threshold for invalid attempts should be set quite high.

- If hidden fields or client cookies show clear evidence of tampering, the IP address should be blacklisted. However, this should be tested with a wide variety of Web browsers in order to avoid a large number of mistaken alerts.

- If the server receives requests containing invalid data that should have been filtered by client side validity checks. This is often an indicator that the user is deliberately bypassing those checks in hope that the server will act on the incorrect data. (See the *Client Input Filters* pattern.) It is important to recognize that in some cases—such as when JavaScript is disabled on the client or blocked by the client's firewall—the client side checks may be disabled through no fault of the user. If the site must be accessible to users who will not accept JavaScript, this should not be considered as evidence of attempted misuse of the site.

Possible Attacks

There are a number of possible attacks that could be perpetrated against this pattern:

- Denial-of-Service Attack – an attacker may constantly hit a site after he/she has been blocked, causing that site to eat up resources to verify their address, or the attacker may purposely block addresses so that legitimate clients sharing the same address cannot access the site. This pattern might be ineffective against large-scale, distributed denial-of-service attacks where many individual network addresses are used. Similarly, attackers may make it appear that the attack is coming from critical infrastructure such as a router, database, or firewall, causing you to block these servers, rendering the site in-operable.

- Resource consumption attack – attackers may grow the blacklist to a large size so that it is inefficient to examine the blacklist, causing a general degradation in performance.

Examples

Manual implementations of this pattern are very common. Best practice in Web security is to implement some sort of intrusion detection, even if it consists only of auditing the usage logs. Administrators that observe repeated misuse originating from certain IP addresses generally contact the ISP in question. If this fails to stop the behaviors in question, the administrator generally has the capability to block further access via a firewall rule.

Commercial intrusion detection systems often incorporate some sort of blocking mechanism. ISS RealSecure, for example, implements “firecell rules,” which allow known attacks to be blocked in real time. This capability can be integrated with certain commercial firewalls, to allow blocking rules to be inserted into the firewall in real time. It should be noted that these are good at detecting network-level attacks, and even attacks on Web server vulnerabilities. But they will not detect application level misuse out-of-the-box. Instead, they offer APIs that allow application developers to report suspicious activities that are specific to the application.

We know of Web applications that track invalid login attempts by IP address. Once a certain number of invalid login attempts have been observed originating from a specific IP addresses, further login attempts are simply ignored. This ensures that the account lockout mechanism cannot be misused to effect denial of service conditions on more than a handful of accounts.

A similar approach has been used to combat unsolicited junk e-mail. The Maps approach maintains a large list of blacklisted IP addresses. As IP addresses are observed as the origination point for SPAM, they are added to the blacklist. The blacklist is made available to servers across the Internet, which discard mail from any blacklisted address. For more details, see http://mail-abuse.org.

We are currently developing an example of a fully automated mechanism that lets Java servlets insert NetFilter (“iptables”) rules on the system hosting the application. The source code will be made available on-line upon completion.

Trade-Offs

Accountability	The network address blacklist is necessary when there is no other recourse, and by definition no accountability. In some sense, denying further access is a form of punishment and could be considered a measure of accountability.

Availability	A network address blacklist can have a negative effect on availability. Specifically, the network address blacklist offers an interface whereby specific network addresses can be blocked from further access: if this interface is misused or implemented incorrectly, it could cause legitimate users to be denied access. Also, all users whose requests originate from behind the same firewall may appear as if coming from the same network address. It is conceivable that one user could cause an entire enterprise to become locked out of the Web site.
Confidentiality	When implemented effectively, a network address blacklist enhances site confidentiality and integrity. This occurs because it will dissuade or prevent attempts to misuse the Web application (since many forms of misuse are intended to compromise those security characteristics). Even if ineffective, it will not adversely affect these properties.
Integrity	See Confidentiality.
Manageability	A network address blacklist can have a positive effect on the administrative burden of the Web site. By eliminating many nuisance attacks, the system administrators can be freed to investigate more significant security concerns. However, if legitimate users are finding themselves locked out on a routine basis, the management burden will be raised, particularly if administrators have to intervene manually to clear an address from the blacklist.
Usability	If the server is tuned too sensitively, it may cause legitimate behaviors to be reported as suspicious. This could have a very negative effect on usability. But if the server is fairly conservative, legitimate users should never see any impact on usability. Those who attempt to misuse the system will see a significant negative effect on their usability of the system.
Performance	A network address blacklist has mixed effects on performance. On the one hand, misbehaving users who might otherwise be consuming considerable resources are prevented from gaining access to the site. On the other hand, the lockout mechanism could itself incur significant performance penalty on all users. If a network address blacklist is implemented in a way that accesses a database for every Web request, the performance will suffer. If the lockout depends on an in-memory blacklist, it is important that the size of the list be bounded and that the list does not itself become a bottleneck.
Cost	Manual administrator lockout adds little additional expense. If automated lockout can be implemented using existing APIs, it can be developed quite cheaply. If novel interfaces are required, it will be

	quite expensive to develop. In either case, quality assurance will be non-trivial.

Related Patterns

- *Account Lockout* – a related pattern that supplements, and should be coordinated with, the network address blacklist.
- *Client Input Filters* – a related pattern that provides a source of information about suspicious activities.
- *Log for Audit* – a related pattern.
- *Minefield* – a related pattern that provides input to the network address blacklist mechanism.

References

[1] Brown University Computing and Information Services. "IP Address Filtering". *Web Authorization/Authentication.* http://www.brown.edu/Facilities/CIS/ATGTest/Infrastructure/Web_Access_Control/Goals Options-ver2.html#anchor136476, October 1997.

[2] Stein, L. and J. Stewart. "The World Wide Web Security FAQ – Version 3.1.2". http://www.w3.org/Security/Faq, February 2002.