Title: A METHOD FOR PROTECTING LOCAL SERVERS FROM DENIAL-OF-SERVICE ATTACKS

Abstract: The invention relates to a computer comprising a local server and a local client. The local server is arranged to serve the local client. The computer is connectable to a network. The local server comprises a prevention module for preventing DOS attacks. The prevention module is set to forbid network access to the local server from outside the computer.
A Method for Protecting Local Servers from Denial-of-Service Attacks

The invention relates to computers equipped with a local web server, and more particularly to personal computers connectable to the internet.

A local server (for example a web server) provides services to application on the local host that the server resides or to the local area network that the host connects to. Such a server typically relies on standard Internet protocols and security protocols for communication and security. Like other Internet nodes, a local server also faces security threats from the local computer or from the network. Protection of such servers is necessary to enable their services.

One of security threats is denial-of-service attack (DoS). Such attacks prevent legitimate users from accessing network resources, such as services on a user's computer or over the Internet. Large-scale DoS attacks could potentially paralyze the Internet.

DoS attacks execute in many different ways. One example is flooding based attacks, such as SYN flooding and ICMP flooding. Various mechanisms are developed for prevention, detection and response to DoS attacks.

This is explained in particular in
- Zwicky, E.D., Cooper, S. and Chapman D.B., Building Internet Firewalls, O'Reilly, 2000,
- “Denial of Service Attacks,” CERT Coordination Center, http://www.cert.org/tech_tips/denial_of_service.html,
- Houle, K.J. and Weaver, G.M. “Trends in Denial of Service Attack Technology,” CERT Coordination Center, October 2001, and
These mechanisms together with other security means, such as firewall and intrusion detection system, provide managed security services to enterprises or ISP networks, which consists of routers, servers, and hosts. However, these mechanisms may not protect a local server from DoS attacks launched within the local computer that the server runs, because the attacker (malicious software) is inside the firewall. Such attacks are causing more and more problems as increasingly sophisticated techniques to distribute or propagate malicious software are on the rise.

The purpose of DoS attacks is to prevent or impair legitimate use of computer or network resources; for example, preventing users from access a popular Internet server. Common targets of resources are bandwidth, processing power, and storage capabilities. The most common DoS attack type is packet-flooding attack, which involves sending a large number of packets to a destination to cause excessive consumptions of resources.

DoS attacks can happen at the application layer as well as at network protocol layers. The DoS defense mechanism must be built at both application and network protocol layers.

Much research work have been done to defend or to mitigate DoS or DDoS attacks in the last decade as seen in particular in:

- “Denial of Service Attacks,” CERT Coordination Center, http://www.cert.org/tech_tips/denial_of_service.html,
- Houle, K.J. and Weaver, G.M. “Trends in Denial of Service Attack Technology,” CERT Coordination Center, October 2001,

• "Distributed Denial of Service (DDoS) Attacks/tools," http://staff.washington.edu/dittrich/misc/ddos/.

One basic approach is to detect DoS attack and to filter out attack packets. The Internet infrastructure is hierarchical. The attack detection and attack packets filtering can be done at different levels of the network, for example, at local computer, local network, local ISP network, or upstream ISP network. The effectiveness of the attack detection and packet filtering is dependent on the network level or levels that the defensive mechanisms are executed.

Existing research and commercial security products, such as intrusion detection systems (IDS), firewalls and global defense infrastructure, protect enterprise or ISP networks that consist of servers, routers, and host computers. These methods are important and critical to maintain the operations of the Internet and local networks. Most of the techniques protect systems (networks, servers, routers, PCs) from outside attacks. Insider attacks to local servers, such as attacks launched by malicious software inside a PC, are still at large because attack packets may not go to the outer network, i.e. they are inside the firewall.

Lee etc. proposed a port hopping method described in Lee, H.C.J. and Thing, V.L.L., "Port Hopping for Resilient Networks," 2004, http://www.diadem-firewall.org/publications/VTC2004Fall_Port-Hopping.pdf. The UDP/TCP port number used by the server varies as a function of time and a shared secret between the server and the client. The method simplifies the detection and filtering of malicious packets. It works if network clients have agreed with its server on the port hopping mechanism. However, because this method requires both clients and their server implement the port hopping, it does not work for a standard Internet service, such as a web server, which allows access from a standard client; nor for a standard Internet client, such as a web browser, which accesses a standard server.
In order to defend against network-based attacks, many attack
detection or intrusion detections methods are developed. Most existing
detection methods are based on statistics of the packets or signatures (or
patterns) of packets flows:

- Chang, R.K.C., "Defending Against Flooding-Based Distributed
  Denial-of-Service Attacks: A Tutorial," IEEE Communication
  Magazine, October 2002,

- A. Habib, M. Hefeeda, and B. Bhargava, "Detecting Service
  Violations and DoS Attacks," NDSS Conference Proceedings,
  Internet Society, 2003,

- Karig, D. and Lee, R., "Remote Denial of Service Attacks and
  Countermeasures," Princeton University Department of
  Electrical Engineering Technical Report CE-L2001-002, October

http://www.princeton.edu/~rblee/ELE572Papers/karig01DoS.pdf,

- Porras, P. and Neumann, P. "EMERALD: Event Monitoring
  Enabling Responses to Anomalous Live Disturbances," National
  Information Security Conference, 1997,
  http://www.sdl.sri.com/projects/emerald/emerald-niss97.html,

- Siaterlis, C. and Maglaris, B., "Towards Multisensor Data Fusion
  for DoS Detection," The 19th Annual ACM Symposium on
  Applied Computing, March 2004,
  http://www.netmode.ntua.gr/papers/papers/siaterlis_sac04.pdf,
  and

- Valdes, A., Skinner, K., "Adaptive, Model-based Monitoring for
  Cyber Attack Detection,"

These methods require the understanding of normal or abnormal
packets probabilities or signatures and being able to separate normal and
abnormal packet flows. The systems need to be trained and be adaptable to
changes of the statistics or signatures. Most of these methods deal with
known attacks well but have difficulties to unknown attacks.
In Lu, H.K., "A Method of Detecting Network-based Attacks for Resource-Constrained Network Devices," Axalto invention disclosure, October 2005, an efficient attack detection method for small resource-constrained network devices is disclosed. The approach relies on a system's operational behavior, instead of packets probabilities or signatures. It models expected operational behaviors and attack-suspicious behaviors of the system. Real-time monitors raise an alarm when the system has deviated from the expected operations or when attack-suspicious behaviors occur.

Modern computers typically have built-in firewalls that protect computers from outsider attacks. A firewall typically includes packet filtering that filters incoming or outgoing packets to let good packets pass and to block suspicious packets from coming in. Filter rules specify what packets to pass and what packets to reject, thus controlling the packet filtering behavior. Packet filtering helps to protect local servers from DoS attacks to a certain extend as it can filter out potential DoS attack packets.

It is an aim of the invention to protect local servers from DoS attacks launched from inside and outside of the host computers.

A computer according to the invention comprises a local server and a local client. The local server is arranged to serve the local client. The computer is connectable to a network. The local server comprises a prevention module for preventing DOS attacks, wherein the prevention module is set to forbid network access to the local server from outside the computer.

The invention and its advantages will be explained more in details in the following specification referring to the appended drawings, in which:

Figure 1 represents a framework for protecting local servers from DoS attacks,

Figure 2 represents an information flow diagram wherein the detection module sends feedbacks to the protection module.
According to a preferred embodiment of the invention, the invention comprises a framework comprises methods for prevention, detection, and continuation. This framework focuses on protecting secure servers that requires client authentications. In preferred embodiments, the local server is a local secure web server. The method can apply to other local servers as well.

The framework is particularly suitable for protecting local servers from DoS attacks, especially attacks launched within host computers. The framework, as illustrated in Figure 1, comprises three components: prevention, detection, and continuation.

The prevention module provides uses packet filtering and timeout to prevent attacks. The detection module monitors the system’s operational behavior to detect possible DoS attacks. Once a possible attack is detected, the detection module informs the main control module of the local server, which invokes the continuation module. The latter provides means for the local server to continue operation under DoS attacks.

In preferred embodiments, the local server is an application using sockets for network connection. The local server may be a web server. A socket contains information about a particular communication channel, for example, source IP address and port number, destination IP address and port number.

In preferred embodiments, the socket functionality is a part of the network module of the host computer. A secure web server maintains SSL or TLS context for each secure connection. It might keep multiple active SSL/TLS contexts simultaneously.

The local web server only serves clients, web browsers, running on the same host computer as the server. It does not allow access from outside of the computer.

In preferred embodiments of the invention, a filtering module is included in the local server.
If a local server offers services only to local clients, it can perform filtering at the application (the local server) level. For example, a PC has a local web server that only allows access by local web browsers. The web server waits on port 80 or 443. In this case, the web server can include a filtering module, which rejects all incoming connections that are not coming from the local computer. Many computers firewalls block connections from outside, but some of them do not. Therefore, according to the invention a local server should preferably have its own filtering module. The networking module of the host computer preferably does not dispatch connections targeted to ports other than 80 or 443 to the local web server. Therefore, the local server’s filter does not need to filter based on the port number in preferred embodiments.

One type of DoS attacks is to consume target’s memory resource. For example, a web server normally allocates a finite number of sockets and SSL/TLS context structures for incoming connections. A malicious software can simply consume all the sockets and contexts and hold on to them so that the web server cannot accept any genuine connections.

In preferred embodiments a timer is used on each of the resources. For example, the web server associates a timer to a socket. The timer records the time when the socket established a connection. It resets with each new activity. The web server checks each active socket regularly. If a socket has no activity for a prolonged period that exceeds a predefined threshold, the server closes the socket connection. If a new connection request comes in but the server is out of free socket, the server may close a socket connection that has idled the longest time. It may use a second threshold for this purpose.

DoS attacks may happen at the application level as well. For example, an attacker may start an SSL handshaking but does not complete it. The repetition of such action to a web server consumes all the SSL contexts resource and, hence, results a DoS attack. To alleviate this problem, the web server can use a timer on each of the SSL context. (This might be the same timer as the one for socket connection.) If an SSL client connection started
handshaking but did not continue for a pre-specified time threshold, the server aborts the connection.

In preferred embodiments, an operational behavior-based detection method is used for local servers.

The proposed detection mechanism is a part of the local server. Such an embedding provides real-time detection and conserves memory usage. The monitoring code only runs when the code execution passes through there. Therefore, the detection module does not require a separate task; the impact on the system performance is, hence, minimum. Additional advantages of this approach include the ability to detect unknown attacks and simplicity. The effectiveness of the detection does not depend on known packets probabilities or signatures. Furthermore, the method can combine with other existing packets based approaches.

Modeling expected system operational behavior is the first step toward behavior-based DoS attack detection. For example, a local web server operates as the following in normal conditions.

```c
While(true)
{
    socket = accept(...); // waiting for a new connection
    start_new_thread(socket, ...);
}

operation_thread()
{
    ssl_handshaking(socket, sslContext, ...);
    while(true)
    {
        ssl_communication(socket, sslContext, ...);
        if(finished)
```
{
    close(socket);
    reset_ssl_context(sslContext);

    break;
}

In normal operations,

1. The filtering module only drops connections sometimes but infrequently.
2. Timeout happens sometimes but infrequently.
3. SSL (or TLS) handshaking happens immediately after the socket connection, i.e., the client sends the first SSL handshaking message, "Client Hello", as soon as possible after it achieves a socket level connection.
4. The client and the server continue and finish the handshaking as soon as they can.
5. After the handshaking finishes, the client sends its first request immediately. The server answers the request.
6. The server and client may close the connection after one or more rounds of request and response.

Note: With SSL/TLS, a successful handshaking opens a secure channel to carry HTTP requests and responses. The applications (server or client) use the network layer (TCP/IP, socket) to close the communication channel, which results the disconnection of the secure channel as well.

The operational behavior-based detection method proposed in this paper monitors the operations of the local web server. If the operations drift away from expected behaviors for a prolonged time or a higher frequency, the detection module raises an alarm to inform the web server a possible DoS attack.
With examples of normal operational behaviors listed in the previous section, the detection module raises an alarm in one or more following situations.

1. The filtering module continuously drops connections for a prolonged time period or in high frequency that exceeds a predefined threshold.
2. Timeout happens frequently.
3. Established socket connections do not receive SSL handshaking message from clients. Available sockets reach or drop below their low threshold and the server still receives socket connection requests.
4. Many unfinished SSL handshakings hang on their SSL context structures without making any progress. Available context structures reach or drop below their low threshold and the server still receives socket connection requests.
5. Many established connections have not activities but simply hold on resources or receive meaningless or bad requests or messages continuously. The server still receives connection requests and the level of available resources is low.

A continuation method that a local web server can implement in response to DoS attacks according to preferred embodiments is described below. It aims to continue the service.

When a local web server is under a DoS attack, it is still alive. It can do the following to protect itself and its resource.

1. Stop accepting any new connection requests.
2. Close all suspicious connections, such as those with no activity and those with unfinished SSL/TLS handshaking.
3. Inform the user.
4. Apply a continuation procedure (to be described below).

These steps also enable the network module of the host computer to block the DoS packets because their destination could not be reached. If the attack continues, it might trigger the host computer's defense mechanism, if it has one. The question is what the web server can do to continue it operation.
The local web server can inform the user, for example, by sending a message through a shell window or via a web page to the browser that the user is using. The message can warn the user that the local web server is under a DoS attack and provide options and instructions to the user. The user can choose to exit the service (e.g. close the browser) or to continue. The instruction advises the user either to close all web browsers that connect to the local server or to close all except the one that displays the warning page or the one that the user currently uses.

When a local web server is under a DoS attack and the user has decided to continue regardless, the server may be able to continue service by changing to a different TCP port number. After responding to the last http(s) request and closing all outstanding connections, the server waits on a new TCP port. How does the user or the browser know that the server has changed the port number? The following explains how it works.

The warning and instruction web page contains an http (or https) link that points to the new URL (with the new port number) of the server. The user can click the link to get to a new page. The web server has dynamically constructed this new page with all internal links pointing to the new TCP port number. The page might look exactly the same as the previous page displayed before the DoS attack warning. Or the page might request the user to re-authenticate.

Alternatively, the server might display a warning in a shell window. The server chooses this option when it does not have an outstanding browser request. In this case, after detecting a DoS attack and closing all connections, the server does not change the TCP port number immediately. Instead, it searches for the next genuine HTTPS request among incoming messages. During this time, it quickly closes any connections that do not follow normal operations described in See 6.1. (For more information, see the next section.)

Once received a HTTPS request, the server does the following:

1. Close all connections with the old TCP port number;
2. Select a new TCP port number;
3. Dynamically construct a response web page with all internal links pointing to the new TCP port number. The response web page might be the same as if no DoS attacks except a different TCP port number or be a web page that requests the user to re-authenticate.

4. Send HTTPS responses;

5. Wait connection requests on the new TCP port.

Once the web server closes the old TCP port that it used, the network module of the host computer automatically drops all the connection requests to that TCP port. From the web server perspective, the host computer filters out the attack packets.

The detection module preferably not only triggers the continuation module but also sends feedbacks to the prevention module. Such feedback enables the prevention module to adjust parameters. See Figure 2.

For example, after a detection of a DoS attack and before the web server can change the TCP port number, the detection module can inform the prevention module to adjust timeout values. For example, the prevention module can set the following time thresholds to very small values to aid the web server’s search for a genuine request.

1. Between a socket connection and starting of SSL/TLS handshaking;
2. The gaps during SSL/TLS handshaking;
3. Between finishing SSL/TLS handshaking and before the first HTTPS request.

After changing to a new TCP port, the continuation module re-adjusts these thresholds, for example, through the detection module.

To prevent further DoS attacks, the web server may choose to change its TCP port number regularly if it has only one browser connected to it at the time. The procedure is the same as the one listed above.
CLAIMS

1. A computer comprising a local server and a local client, wherein the local server is arranged to serve the local client, wherein the computer is connectable to a network, the computer being characterized in that the local server comprises a prevention module for preventing DOS attacks, wherein the prevention module is set to forbid network access to the local server from outside the computer.
2. The computer according to claim 1, wherein the prevention module comprises a filtering module.
3. The computer according to claim 1, wherein the prevention module comprises a timeout module, wherein the timeout module is set
   a. to record the time when a connection is established with the local server,
   b. to reset said recorded time to current time whenever there is some activity on the connection, and
   c. to close the connection if there is no activity for a period that exceeds a predefined threshold.
4. The computer according to claim 1, wherein the local server is set to manage user credentials in order to securely authenticate a user of the computer to a remote server through the network.
5. The computer according to claim 1, wherein the local server is a web server.
6. The computer according to claim 1, wherein the local client is a web browser.
7. The computer according to claim 1, wherein the local server comprises a detection module for detecting local DOS attacks, the detection module being set to associate a connection between the local server and the local client with a timer in order to monitor this connection.
8. The computer according to claim 1, wherein the connection between the local client and the local server is established using sockets.
9. The computer according to claim 1, wherein the local server uses a secure protocol for connection with the local client.
10. The computer according to claim 9, wherein the secure protocol is SSL or TLS.

11. The computer according to claim 10, wherein the detection module is set to use the timer in order to verify if SSL or TLS handshaking happens quickly after the connection.

12. The computer according to claim 10, wherein the detection module is set to verify if the connection is dropped.

13. The computer according to claim 12, wherein the detection module is set to verify if the frequency of connections being dropped exceeds a predefined threshold.

14. The computer according to claim 10, wherein the detection module is set to verify if the local server receives an SSL handshaking message from the local client.

15. The computer according to claim 14, wherein the detection module is set to verify if the number of SSL handshakes hanging on their SSL context structures without making any progress exceeds a predefined threshold.

16. The computer according to claim 10, wherein the detection module is set to verify if established connections have no activity.

17. The computer according to claim 16, wherein the detection module is set to verify if established connections hold on resources continuously.

18. The computer according to claim 16, wherein the detection module is set to verify if established connections receive meaningless or bad requests or messages continuously.

19. The computer according to claim 10, wherein the local server further comprises a continuation module, wherein the continuation defines how to continue the connection between the local client and the local server once a DOS attack is detected.

20. The computer according to claim 19, wherein the continuation module is set to instruct the local server to inform the user of the DOS attack.

21. The computer according to claim 19, wherein the continuation module is set to instruct the local server to change the TCP port number used for the connection between the local server and the local client.
22. The computer according to claim 7, wherein the detection module is set to send feedback to the prevention module in order to adjust the prevention module parameters.

23. The computer according to claim 22 and 3, wherein the feedback is set to adjust the threshold of the timeout module.

24. System comprising a resource constrained device and a computer according to claim 1, wherein the local server is loaded on the computer from a resource constrained in order to be executed on the computer.

25. System according to claim 24, wherein the resource constrained device is a USB mass storage device connected to a USB port of the computer.

26. System according to claim 25, wherein the USB mass storage device is a smart card.

27. System according to claim 26, wherein the local server is set to have the smart card execute sensitive parts of the local server.