

## Windows ICF: Can't Live With it, Can't Live Without it

David Wong 2002-08-22

### Windows ICF: Can't Live With it, Can't Live Without it

by David Wong

last updated August 22, 2002

---

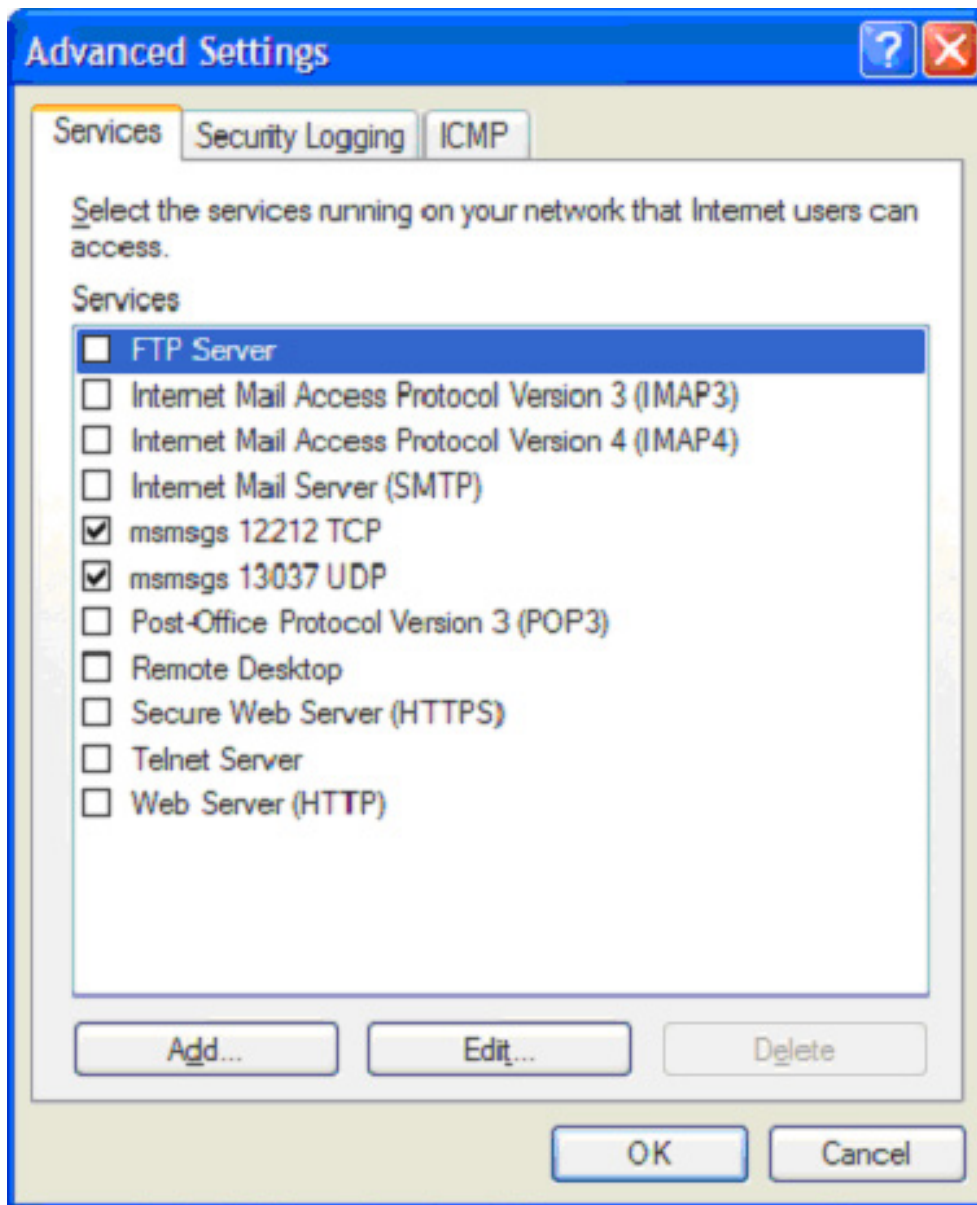
Windows ICF (Internet Connection Firewall) is the built-in firewall in Windows XP, both the Home and Professional editions. ICF is an excellent personal firewall and will prevent most attacks from the Internet. However, the lack of granular control makes ICF much too restrictive for power users. So, as they say, you can't live with it, you can't live without it. For this article, we put ICF into the lab and set our hackers (well, security penetration testers) loose at it to see how good it is. In this article, we will give an overview of ICF, see how ICF performs under a simulated attack, and discuss the pros and cons of ICF.

#### ICF Overview

ICF is a stateful packet filter. The tracking of state allows ICF to make better decisions and support a more comprehensive ruleset than traditional packet filters. By default, the ICF ruleset is very secure and denies all traffic from the Internet, including ICMP echo requests (ping packets). This makes your computer virtually invisible to attackers attempting to probe your machine.

The ICF ruleset can be modified manually by configuring "services" or programmatically by using the ICF API. Out of the box, ICF contains standard services such as FTP and HTTP. In addition, ICF allows you add custom services for services that use ports that are not provided in the custom list. However, what is lacking from the specification of the ruleset is the ability to restrict access to a particular IP address. If you allow port 80 for HTTP services, you are allowing the whole world to connect to port 80. You can't restrict it a specific IP address or a range of IP addresses. This is a huge shortcoming of ICF.

Besides manual configuration of the ruleset, ICF contains an API that allows applications to temporarily modify the ruleset. In the screenshot below, Windows messenger automatically opened up TCP port 12212 and UDP port 13037 for its own use.



This is both a good and scary feature. It's good because it allows applications like Windows messenger the ability to interoperate with ICF. This is especially useful for applications that open up dynamic ports. With applications that open up dynamic ports, you can't specify a rule that would allow the traffic through, since the port could change. This is great for people who play games that support DirectPlay 8. At the same time, most security professionals get a little wary when applications can change firewall rulesets willy nilly. A big complaint people have about the ICF API is that it requires administrative privileges. If your Windows XP account is a "limited" account, applications you run can't manipulate the ICF ruleset using the ICF API. For more information regarding the ICF API, refer to [About Internet Connection Sharing and Internet Connection Firewall](#).

Finally, ICF has some additional checks besides the standard stateful packet filtering, such as:

1. It enforces a 3-way handshake – this is useful to prevent special scanning techniques.
2. It blocks packets that have impossible flags options (such as a packet with both SYN and FYN bits) – this is useful for preventing attacks against the IP stack with invalid packets
3. Prevention of IP spoofing using Raw Sockets and the IP\_HDRINCL option – this is useful for preventing some forms of distributed denial of service attacks.

For Microsoft's official overview of ICF, refer to Microsoft Technet's [Internet Connection Firewall overview](#). For a more detailed document on ICF, refer to Microsoft's [Internet Connection Firewall Feature Overview](#). Additionally, Chris Weber from Foundstone is about to publish a book on Windows XP security with a chapter on ICF.

## Putting ICF Under Attack

For the purposes of testing, we deployed a single Windows XP network and the Internet with ICF enabled. Note: for the purpose of this test, we did not deploy ICF with ICS (Internet connection sharing) in a home network environment, which is another common configuration. We then launched attacks against ICF from both the Internet as well as from the Windows XP box itself.

## Tools

We used the following tools for testing:

1. [ISIC](#) - IP Stack Integrity Checker
2. [Fscan](#) - Windows-based port scanner.
3. [Nmap](#) - Port scanner.
4. [Foundscan](#) - Vulnerability assessment tool.
5. [Nessus](#) - Vulnerability assessment tool.
6. [fragrouter](#) - IP packet fragmenter.

## Methodology and Results

Our testing methodology was designed to simulate attacks from the hacker community against users of ICF. To simulate attacks, we used publicly available freeware and commercial portscanners and vulnerability assessment tools. In addition, we used tools such as ISIC and fragrouter to attack the stateful packet firewall component of ICF. The testing methodology

incorporated attacks in firewalls previously identified in other firewalls to ensure that Microsoft did not make the same mistake as other vendors, such as handling fragmented packets improperly.

The testing focused on attacks on ICF from the Internet. Since most attacks against a personal firewall will be from the Internet, this is where the risk is highest. In addition, we tested ICF from the perspective of a malicious user on the host in order to identify other potential avenues of attack.

We ran ISIC, the portscanners, and vulnerability assessment tools listed above against the Windows XP box running ICF. Portscanners and vulnerability assessment tools are used by many hackers in order to identify services and potential vulnerabilities. ISIC and Nmap were used to generate non-RFC compliant packets in an attempt to cause ICF to choke. ICF not only blocked all the attacks, but there was no degradation from the user's point of view. A user would not even notice the attacks were being launched. When using ISIC and Nmap to generate non RFC compliant packets, ICF blocked all of the packets since they were invalid packets.

The same attacks were run using fragrouter to fragment the IP packets. Using fragrouter, we were able to break a single TCP packet into multiple packets in attempt to bypass ICF. No success here either. We also tried other standard attacks against firewalls, such as fragment overlay attacks. All in all, the result was that ICF performed very solidly against attacks from the Internet.

Since most stateful firewalls have performance and DoS issues with internal users overloading the state table, we ran the Fscan port scanner from the ICF box. There was a bit of a surprise here. Although no DoS situation was created, we noticed that no matter which host we scanned, ports 21, 389, 1002, and 1720 were always open. This situation scared the hell out of our testers: why would ICF be opening up ports? Is it a backdoor? After much research, we determined that these ports were open due to an application level proxy in the ICF/ICS service. It turns out that in order to support protocols that don't "play nice" with firewalls, Microsoft implemented applications proxies.

What do I mean by "play nice"? Well, protocols that dynamically open up incoming ports makes building a simple stateful firewall impossible. Let's use FTP and H.323 as examples. Under normal FTP behavior (not PASV), the FTP client will connect to the FTP server on port 21. This is all fine and dandy until the client issues the FTP PORT command. The FTP PORT command tells

the FTP server to connect back to the FTP client on an arbitrary incoming port to send data. If the firewall doesn't detect and process the PORT command, it would normally drop the packet. Hence, ICF implemented an FTP proxy to address the PORT command.

Similarly, H.323 is used for voice-over IP call set-up. This is the call set-up only. The actually voice and video RTP streams use dynamic ports that are generated during call set-up.

Therefore, in order for ICF to allow the traffic through, it needs to listen in on the call set-up.

Ports 389 and 1002 and used for LDAP and ILS (Internet Locator Service) for Netmeeting to work.

The existence of the application proxies were developed for ICS. However, since ICS and ICF are implemented as one service, the proxies will be there even if you don't have ICS enabled. Although there may be security bugs in the proxies, our testing stopped here. Why you ask? The answer is if a hacker was already on the box or on the internal network in the case of ICS, you probably have bigger problems. However, the application proxies are probably good areas of research for you bug hunters out there.

The following table illustrates the ICF/ICS application proxy ports:

Port	Service	Reason
21	FTP	NTP PORT command
398	LDAP	Netmeeting
1002	ILS	Netmeeting
1720	H.323	Netmeeting

In addition, ICF listens in on PPTP connections to allow GRE packets through as well as T.120 connections used for Voice over IP.

Although ICF performed well under attacks from the Internet, we contrived an e-mail Trojan attack that obviously passed right through the firewall. Admittedly, ICF is not a virus/trojan scanner and should not detect a trojan; however, the lack of egress filtering allowed our Trojan to "phone home" and open up a connection for an attacker. In addition, although ICF protects against DDoS attacks by preventing manipulation of the IP source address, we were still able to modify IP source addresses by generating Ethernet packets by using NDIS driver calls. See the

SecurityFocus article [When Dreamcasts Attack](#) for more information on "phone home" attacks.

To sum up, although we had a bit of a shock with the application proxies, ICF performed well under attacks. However, the lack of egress filtering is a huge flaw in the design of ICF.

## Pros

Windows ICF has a lot of features and advantages going for it. It's difficult to overlook some of these features, particularly the following:

- ICF is free. You can't argue with free and integrated into the operating system. Well, unless you are Netscape or the Department of Justice.
- ICF performed robustly under attack as well as under high utilization.
- ICF will prevent most attacks from the Internet. By default, the firewall ruleset is very restrictive, preventing most attacks. I say most, because it won't prevent against virus and other attacks that initiate from your computer.
- ICF is a stateful firewall. Stateful firewalls are generally more secure than packet filters.
- ICF may already be installed and active. If you've used the Network Setup Wizard, it may have already turned on ICF for you.
- The ICF programmatic API allows ICF aware applications to open up "holes" in the ICF to allow incoming traffic. This is useful for programs like Windows Messenger and games like Warcraft 3 that require inbound connections.
- Application proxies allow ICF to work with firewall unfriendly protocols

## Cons

Despite all of these advantages, Windows ICF may cause some problems, especially for corporate users and power users who need more control of their firewall. Some of the problems listed below may be difficult for some users to accept.

- ICF breaks a lot of applications. This is arguably a good feature for a firewall, after all, firewalls are designed to stop traffic. However, the inability to create granular access rules and specify "trusted" hosts encourages users to just shut it off. ICF doesn't support RPC, so message notification in Outlook will not function properly. In addition, services such as file sharing won't work.
- ICF does not perform any outbound filtering. This is by design; however, it makes ICF

useless against Trojans and other malicious applications that "phone home".

- ICF lacks real-time notification of attacks. ICF can be configured to log allowed or denied traffic in a file; however, no real time notification is available.
- The ability of applications to dynamically open ports requires administrative-level privileges. If you're applying the "Principle of Least Privileges" and your account is only a "limited" account, the applications you run will not be able to take advantage of the ICF programmatic API and open up ports dynamically.

## Conclusion

ICF is an excellent security tool for most people. It blocks most attacks from the Internet. And it works *automagically* with applications that are aware of the ICF API. It has support for industry standard protocols like FTP, H.323, and PPTP. However, if you are a corporate user and/or a power user, you'll probably want another personal firewall that allows more granular control. Without the granular control, power users are forced to disable the firewall in order to get some applications to work. My advice: use ICF if it works for you; if you have been forced to turn off the firewall (like I have), go out and buy a personal firewall with more buttons that you can tweak. For more information on alternative firewalls, the following SecurityFocus article is an excellent discussion: [Securing Privacy, Part Two: Software Issues](#).

*About the author: [David Wong](#), CISSP, is a principal consultant at Foundstone, a provider of security assessment services. David is a lead member of Foundstone's product assessment group where he has performed security assessments of shrink-wrapped software and Web application assessments for several years.*

[Privacy Statement](#)

Copyright 2006, SecurityFocus