Ignoring the Great Firewall of China

Richard Clayton, Steven J. Murdoch, and Robert N.M. Watson

University of Cambridge, Computer Laboratory, William Gates Building,
15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom
{richard.clayton, steven.murdoch, robert.watson}@cl.cam.ac.uk

Abstract. The so-called "Great Firewall of China" operates, in part, by inspecting TCP packets for keywords that are to be blocked. If the keyword is present, TCP reset packets (viz: with the RST flag set) are sent to both endpoints of the connection, which then close. However, because the original packets are passed through the firewall unscathed, if the endpoints completely ignore the firewall's resets, then the connection will proceed unhindered. Once one connection has been blocked, the firewall makes further easy-to-evade attempts to block further connections from the same machine. This latter behaviour can be leveraged into a denial-of-service attack on third-party machines.

1 Introduction

The People's Republic of China operates an Internet filtering system which is widely considered to be one of the most sophisticated in the world [9]. It works, in part, by inspecting web (HTTP) traffic to determine if specific keywords are present [8]. These keywords relate to matters such as groups that the Chinese Government has banned, political ideologies that they consider unacceptable and historical events that the regime does not wish to have discussed.

It is straightforward to determine that the keyword-based blocking is occurring within the routers that handle the connections between China and the rest of the world [14]. These routers use devices based upon intrusion detection system (IDS) technology to determine whether the content of packets matches the Chinese Government's filtering rules. If a connection from a client to a webserver is to be blocked then the router injects forged TCP resets (with the RST flag bit set) into the data streams so that the endpoints will abandon the connection. Once blocking has begun, it will remain in place for many minutes and further attempts by the same client to fetch material from same website will immediately be disallowed by the injection of further forged resets.

In Section 2 of this paper we discuss the methods available to countries that wish to prevent their citizens from accessing particular Internet content and the strengths and weaknesses that have been identified by previous investigators. In Section 3 we present the packet traces we obtained from each endpoint of some connections that were blocked by the Chinese firewall system. In Section 4 we propose a model for the operation of this firewall to explain the results we have obtained. Then in Section 5 we show that by ignoring the TCP resets being issued

G. Danezis and P. Golle (Eds.): PET 2006, LNCS 4258, pp. 20–35, 2006.

by the firewall we are able to successfully transfer material that was supposed to be blocked, and discuss why this may prove difficult for the firewall operators to address. In Section 6 we show how the blocking action of the firewall can be leveraged into a denial-of-service attack on third party machines. Finally, in Section 7, we consider how websites outside of China might make their material easier to access despite the blocking, and we discuss the merits and demerits of this method of evading censorship.

2 Content Blocking Systems

Three distinct methods of content blocking – packet dropping, DNS poisoning and content inspection – have been identified in previous papers by Dornseif [5], who studied the blocking of right-wing and Nazi material in Nordrhein-Westfalen and Clayton [3] who studied the hybrid blocking system deployed by BT in the United Kingdom to block access to paedophile websites.

2.1 Packet Dropping Schemes

In a packet dropping scheme, all traffic to specific IP addresses is discarded and the content hosted there becomes inaccessible. This scheme is low cost and easy to deploy – firewalls and routers offer the necessary features as standard.

Packet dropping schemes suffer from two main problems. Firstly, the list of IP addresses must be kept up-to-date, which could pose some difficulties if the content provider wishes to make it hard for an ISP to block their websites (for details of the complexity, see the extensive discussion in [4]). Secondly, the system can suffer from "overblocking" – all of the other websites that share the same IP address will also be blocked. Edelman [6] investigated the potential extent of overblocking and found that 69.8% of the websites for `.com`, `.org` and `.net` domains shared an IP address with 50 or more other websites. Although some of these domain names will have merely been "parked", and providing a generic webpage, the detailed figures show a continuum of differing numbers of websites per IP address, reflecting the prevailing commercial practice of hosting as many websites as possible on every physical machine.

2.2 DNS Poisoning Schemes

In a DNS poisoning scheme, it is arranged that when the Domain Name System (DNS) is consulted to translate a textual hostname into a numeric IP address, no answer is returned; or an incorrect answer is given that leads the user to a generic site that serves up a warning about accessing forbidden content.

These schemes do not suffer from overblocking in that no other websites will be affected when access to a specific host is forbidden. However, it can be difficult to make them work correctly if all that is to be blocked is a website, and email contact is still to be permitted. Dornseif demonstrated that all of the ISPs in his sample had made at least one mistake in implementing DNS poisoning.

2.3 Content Inspection Schemes

Most content inspection schemes work by arranging for all traffic to pass through a proxy which refuses to serve any results for forbidden material. These systems can be made extremely precise, potentially blocking single web pages or single images, and permitting everything else to pass through unhindered.

The reason that proxy-based systems are not universally employed is that a system that can cope with the traffic volumes of a major network – or an entire country – would be extremely expensive. In Pennsylvania USA, a state statute requiring the blocking of sites adjudged to contain child pornography was struck down as unconstitutional in September 2004 [13]. For cost reasons, the Pennsylvanian ISPs had been using a mixture of packet dropping and DNS poisoning. The resultant overblocking and "prior restraint" were significant factors in the court's decision.

Nevertheless, proxy-based systems have been deployed in countries such as Saudi Arabia [7], Burma [10] and on specific network providers such as Telenor in Norway [12]. The UK-based BT system studied by Clayton was a hybrid design, utilising a low-cost cache, because only the packets destined for relevant IP addresses would be passed to it. Unfortunately, this permits users to "reverse-engineer" the list of blocked sites. Since these sites provide illegal images of children, this runs counter to the public policy aim of the system.

An alternative method of performing content inspection uses components from an Intrusion Detection System (IDS). The IDS equipment inspects the traffic as it passes by and determines whether or not the content is acceptable. When the content is to be blocked it will arrange for packets to be discarded at a nearby firewall or, in the case of the Chinese system, it will issue TCP reset packets so as to cause the offending connection to be closed.

An IDS-based system is significantly more flexible than the other schemes, and it is much less simple to circumvent. Both Dornseif [5] and Clayton [4] have extensive discussions on how to circumvent the different types of content blocking they identify. However, the IDS approach ought to be able to detect the traffic no matter what evasion scheme is tried, provided that the traffic remains in the clear and is not encrypted or obfuscated in a manner that the IDS cannot convert to a canonical form before coming to a decision.

3 How the Chinese Firewall Blocks Connections

In our experiments we were accessing a website based in China (within the Chinese firewall) from several machines based in Cambridge, England (outside the Chinese firewall). The Chinese firewall system, as currently deployed, is known to work entirely symmetrically[1] – detecting content to be filtered as it passes in both directions – and by issuing all the commands from the Cambridge end we avoided any possibility of infringing Chinese law.

[1] This symmetry is necessarily present because it permits the firewall to block both requests that are deemed to be unacceptable and the return of unacceptable content.

3.1 Blocking with Resets

Initially we accessed a simple web page, which arrived in an entirely normal manner, just as would be expected. As can be seen from the packet dump below, after the initial TCP three-way handshake (SYN, SYN/ACK, ACK) the client (using port 53382 in this instance) issues an HTTP GET command to the server's `http` port (`tcp/80`) for the top level page (/), which is then transferred normally. We were using Netcat (`nc`) to issue the request, rather than a web browser, so that we might avoid extraneous detail. The packet traces were captured by `ethereal`, but we present them in a generic format.

```
cam(53382)   → china(http) [SYN]
china(http) → cam(53382)  [SYN, ACK]
cam(53382)   → china(http) [ACK]
cam(53382)   → china(http) GET / HTTP/1.0<cr><lf><cr><lf>
china(http) → cam(53382)  HTTP/1.1 200 OK (text/html)<cr><lf> etc...
china(http) → cam(53382)  ... more of the web page
cam(53382)   → china(http) [ACK]
              ... and so on until the page was complete
```

We then issued a request which included a small fragment of text that we expected to cause the connection to be blocked, and this promptly occurred:

```
cam(54190)   → china(http) [SYN]
china(http) → cam(54190)  [SYN, ACK] TTL=39
cam(54190)   → china(http) [ACK]
cam(54190)   → china(http) GET /?falun HTTP/1.0<cr><lf><cr><lf>
china(http) → cam(54190)  [RST] TTL=47, seq=1, ack=1
china(http) → cam(54190)  [RST] TTL=47, seq=1461, ack=1
china(http) → cam(54190)  [RST] TTL=47, seq=4381, ack=1
china(http) → cam(54190)  HTTP/1.1 200 OK (text/html)<cr><lf> etc...
cam(54190)   → china(http) [RST] TTL=64, seq=25, ack zeroed
china(http) → cam(54190)  ... more of the web page
cam(54190)   → china(http) [RST] TTL=64, seq=25, ack zeroed
china(http) → cam(54190)  [RST] TTL=47, seq=2921, ack=25
```

The first three reset packets had sequence values that corresponded to the sequence number at the start of the GET packet, that value plus 1460 and that value plus 4380 (3×1460).[2] We believe that the firewall sends three different values to try and ensure that the reset is accepted by the sender, even if the sender has already received ACKs for "full-size" (1460 byte) packets from the destination. Setting the sequence value of the reset packet "correctly" is necessary because many implementations of TCP/IP now apply strict checks that the value is within the expected "window". The vulnerabilities inherent in failing to check for a valid sequence value were first pointed out by Watson in 2004 [15].

[2] When we enabled TCP timestamps, and the packets contained 12 bytes of TCP options, we observed that these values changed to multiples of 1448.

The trace also shows part of the web page arriving from the Chinese machine after the connection had already been aborted (we examine why this occurred below). The Cambridge machine therefore sent its own TCP resets in response to these two (now) unexpected packets. Note that it zeroed the acknowledgement fields, rather than using a value relative to the randomly chosen initial value.

All of the reset packets arrived with a time-to-live (TTL) field value of 47, whereas the packets from the Chinese webserver always had a TTL value of 39, indicating that they were from a different source. If both sources set an initial value of 64, then this would indicate the resets were generated 8 hops away from the webserver, which `traceroute` indicates is the second router within the China Netcom Corporation network (AS9929) after the traffic is passed across from the Sprint network (AS1239).

We also examined this blocked connection from the point of view of the Chinese webserver:

```
cam(54190)  → china(http) [SYN] TTL=42
china(http) → cam(54190)  [SYN, ACK]
cam(54190)  → china(http) [ACK] TTL=42
cam(54190)  → china(http) GET /?falun HTTP/1.0<cr><lf><cr><lf>
china(http) → cam(54190)  HTTP/1.1 200 OK (text/html)<cr><lf> etc...
china(http) → cam(54190)  ...more of the web page
cam(54190)  → china(http) [RST] TTL=61, seq=25, ack=1
cam(54190)  → china(http) [RST] TTL=61, seq=1485, ack=1
cam(54190)  → china(http) [RST] TTL=61, seq=4405, ack=1
cam(54190)  → china(http) [RST] TTL=61, seq=25, ack=1
cam(54190)  → china(http) [RST] TTL=61, seq=25, ack=2921
cam(54190)  → china(http) [RST] TTL=42, seq=25, ack zeroed
cam(54190)  → china(http) [RST] TTL=42, seq=25, ack zeroed
```

As can be seen, when the "bad" packet was detected, the firewall also sent resets to the Chinese machine, but these resets arrived after the GET packet (and after the response had commenced). The last two resets (with zeroed ack values), were the ones that were sent by the Cambridge machine.

The other resets (generated because `falun` was present) arrived at the Chinese webserver with a TTL value of 61, which is consistent with them being generated 3 hops away with an initial count of 64. This differs from the 8-hop offset we observed from Cambridge. However, it is possible that there is more than one device that is generating resets – or the initial count may have been adjusted to be different from 64. We do not currently have any definitive explanation for the lack of symmetry that this observation represents.

The first three blocking resets were also set to a range (+25, +1485, +4405) of sequence numbers in an attempt to ensure that at least one was accepted, and in fact the +25 packet will have reset the connection.[3] The fourth and

[3] If the resets had arrived *before* the GET packet, then the resets would *not* have been accepted. The server is running FreeBSD and in this stage of a connection its TCP stack will, to provide protection against denial-of-service attacks, only accept a reset where the sequence number exactly matches the last acknowledgement sent. Before the GET arrives that value is +1, and hence all of the resets would be ineffective.

fifth resets received can be seen, by examining their acknowledgement values, to be responses to the two packets that the server managed to send before the connection was reset.

3.2 Immediate Reset of Connections

The firewall is not just inspecting content but has other blocking rules as well. Having made a "bad" connection we found that, for a short period, all web traffic between the same two hosts was blocked, before any determination could possibly have been made as to the content. This can also be seen in the previous example – but it applies to new connections as well. For example, immediately after the example documented above we saw this:

```
cam(54191)  → china(http) [SYN]
china(http) → cam(54191)  [SYN, ACK] TTL=41
cam(54191)  → china(http) [ACK]
china(http) → cam(54191)  [RST] TTL=49, seq=1
```

Here the reset packet came from the firewall (which sent a reset to the webserver as well). If the client manages to send out its GET packet before the reset arrives from the firewall then multiple resets arrive from the firewall (even if the GET is entirely innocuous). These are then followed by resets from the webserver – which usually receives the resets promptly and so it will have torn down the connection before the GET arrives.

It should be noted that the firewall does not attempt to reset the connection at the SYN stage but waits for the SYN/ACK. Although the client could immediately be sent valid reset packets when the SYN is seen, it is only when the SYN/ACK packet is observed that a reset can be constructed with valid values for the server to act upon.

In our experiments, we found that the length of time for which a pair of endpoints would be prevented from communicating was somewhat variable. Sometimes the blocking would only last for a few minutes, yet at another time the block would be present for most of an hour. The average value was around 20 minutes, but because we saw significant clustering of times around specific values we suspect that different firewall system components may be setting different time delays; and hence a better understanding of which component was to handle our traffic would enable us to predict the blocking period fairly accurately.

3.3 Application to Other Chinese Networks

We obtained a list of Chinese Autonomous Systems (ASs)[4] and from that generated a list of all Chinese subnets that were present in the global routing table. We then used a modified `tcptraceroute` to determine which ASs were handling traffic as it crossed from international networks into China, and from this learnt the identities of the major Chinese border networks. These turned out to be: AS4134, AS4837, AS7497, AS9800, AS9808, AS9929, AS17622, AS24301

[4] `http://bgpview.6test.edu.cn/bgp-view/cur_ana/ipv4cn/china_asnlist.shtml`

and AS24489. We then selected an example web server within each of these ASs and found that similar RST behaviour occurred on all of these networks except AS24489 (Trans-Eurasia Information Network). From this we conclude that our results are extremely typical of the "Great Firewall of China", as it exists in late May 2006, but are not necessarily universally applicable.

4 Design of the Chinese Firewall

Based on the results of our experiments, and descriptions of the type of devices and technologies known to be employed in China – such as Cisco's "Secure Intrusion Detection System" [2] – we propose the following model for the operation of a router that is a part of the Chinese firewall. This model fits our observations well, but it remains speculative because the Chinese network providers do not publish any specifications of their systems.

When a packet arrives at the router it is immediately placed into an appropriate queue for onward transmission. The packets are also passed to an out-of-band IDS device within which their content is inspected. If the packet is considered to be "bad" by the IDS device (because of a keyword match) then three TCP reset packets – with the three different sequence numbers – are generated for each endpoint and given to the router to be transmitted to their destinations.

We do not expect that the IDS, being a logically separate device, will have the capability to remove "bad" packets from the router transmission queue (especially since they might have already been transmitted before a decision is made). Hence it is limited to emitting resets to cause connections to close.

If there is some congestion within the router, and the IDS device is keeping up, then the reset packet will be sent ahead of the "bad" packet; and this is what we mainly observed in our experiments, although sometimes it would lag behind. The values chosen for the reset packets strongly suggest that the designers were concerned that if there is some congestion within the IDS device, compared with the router, then several "bad" packets may have already been transmitted and so the reset packets will reach the destination after these have arrived.

Once the IDS system has detected behaviour it wishes to block, it might add a simple discard rule to the main router, rather than issuing resets. We strongly suspect that this does not scale well within major, high-speed, routers, but that scaling the blocking within the IDS systems is cheaper and easier.

We have already observed, from the time periods for which connections were blocked, that there seemed to be several devices providing the firewall functionality. We ran a further experiment which sent 256 packets containing the offending string through the firewall. Although these packets came from a single machine, we set their source addresses to 256 consecutive IP address values, viz: the Chinese firewall would believe that 256 different, albeit related, machines were sending content that was to be blocked. We observed that the reset packets that were returned to us would sometimes arrive "out of order".

The modern Internet generally arranges for packets to be processed in FIFO (first-in, first-out) queues, so the simplest explanation for the lack of ordering

was that different packets had been passed to different IDS systems, whose own FIFO queues were not equally loaded at the moment they issued the resets. Unfortunately, we found that the experiment engendered so much packet loss (not all of the resets were returned for all of the connections) that it was not possible to form a view as to how far out of order packets could come – and hence establish a lower bound on the number of parallel IDS devices. We intend to return to this experiment at a later time.

4.1 Firewall "State"

There is no evidence that the out-of-band IDS devices communicate with each other so as to create a shared notion of the "state" of connections that pass through the firewall. Experiments demonstrate that triggering a firewall in one border network did not affect the traffic passing through another.

Even where "state" might be expected to be preserved – within the IDS devices – there is no stateful TCP inspection: splitting the `?falun` query across packets is sufficient to avoid detection. Furthermore, the devices are unaware of whether an open connection exists, so that for many of our tests we did not perform the three-way handshake to open a connection but just sent the packet containing the HTTP `GET` request. In fact, apart from the ongoing blocking of traffic after the initial detection occurs, there is no evidence for the IDS devices doing anything other than acting upon one packet at a time.

5 Deliberately Ignoring Resets

The firewall relies entirely upon the endpoints implementing the TCP protocol [11] in a standards-compliant manner and aborting the connection when a reset packet is received. The firewall could sometimes be slightly caught out, as we noted above, when the resets beat the GET packet to the destination and so they were ignored by the careful validation that was applied. Nevertheless, the connection was successfully torn down as soon as the next packet transited the firewall, and hence this didn't make much overall difference.

But now consider what happens if the endpoints do *not* conform to the standards and the TCP resets are entirely ignored. We might expect the firewall to have no impact on HTTP transfers, despite them triggering the IDS system.

We therefore conducted a further experiment with both of the endpoints ignoring TCP resets. We could have achieved this in a number of different ways, but we chose to set appropriate rules within packet filtering firewalls. Within Linux we installed `iptables` and gave the command:

```
iptables -A INPUT -p tcp --tcp-flags RST RST -j DROP
```

which specifies that incoming TCP packets with the RST flag set are to be discarded. If we had been using FreeBSD's `ipfw` the command would have been:

```
ipfw add 1000 drop tcp from any to me tcpflags rst in
```

Once we were discarding TCP resets we found that we could indeed transfer a web page without any blocking occurring. Examining the traffic at the Cambridge end of the connection we saw the results:

```
cam(55817)  → china(http) [SYN]
china(http) → cam(55817)  [SYN, ACK] TTL=41
cam(55817)  → china(http) [ACK]
cam(55817)  → china(http) GET /?falun HTTP/1.0<cr><lf><cr><lf>
china(http) → cam(55817)  [RST] TTL=49, seq=1
china(http) → cam(55817)  [RST] TTL=49, seq=1
china(http) → cam(55817)  [RST] TTL=49, seq=1
china(http) → cam(55817)  HTTP/1.1 200 OK (text/html)<cr><lf> etc
china(http) → cam(55817)  ... more of the web page
cam(55817)  → china(http) [ACK] seq=25, ack=2921
china(http) → cam(55817)  ... more of the web page
china(http) → cam(55817)  [RST] TTL=49, seq=1461
china(http) → cam(55817)  [RST] TTL=49, seq=2921
china(http) → cam(55817)  [RST] TTL=49, seq=4381
cam(55817)  → china(http) [ACK] seq=25, ack=4381
china(http) → cam(55817)  [RST] TTL=49, seq=2921
china(http) → cam(55817)  ... more of the web page
china(http) → cam(55817)  ... more of the web page
cam(55817)  → china(http) [ACK] seq=25, ack=7301
china(http) → cam(55817)  [RST] TTL=49, seq=5841
china(http) → cam(55817)  [RST] TTL=49, seq=7301
china(http) → cam(55817)  [RST] TTL=49, seq=4381
china(http) → cam(55817)  ... more of the web page
china(http) → cam(55817)  [RST] TTL=49, seq=8761
              ... and so on until the page was complete
```

viz: the web page was transferred in a normal manner except for the TCP reset packets generated by the firewall. Since these were all ignored (there were 28 resets sent in total), they had no effect on the client's TCP/IP stack – which continued to accept the incoming web page, issuing ACKs as appropriate. A similar pattern of RSTs mixed in amongst the real traffic could also be seen at the Chinese end.

Hence, by simply ignoring the packets sent by the "Great Firewall", we made it entirely ineffective! This will doubtless disappoint its implementers.

5.1 Blocking with Confusion

As well as blocking further connections by issuing TCP resets once the connection was established, we observed that parts of the firewall occasionally used an additional strategy. On some pairs of endpoints (apparently at random), we saw a forged SYN/ACK packet arrive from the firewall. This contained an apparently random (and hence invalid) sequence number.

If the SYN/ACK packet generated at the firewall arrives at the client before the real SYN/ACK then the connection fails. The sequence of events is that the

client records the random sequence number from the specious SYN/ACK and returns what the server considers to be an incorrect ACK value. This triggers a reset packet and the client closes. In practice, there are a number of other packets in a typical trace when the client is prompt in sending its GET, causing both the firewall and the server to respond with further resets:

```
cam(38104)   → china(http) [SYN]
china(http) → cam(38104)  [SYN, ACK] TTL=105
cam(38104)   → china(http) [ACK]
cam(38104)   → china(http) GET / HTTP/1.0<cr><lf><cr><lf>
china(http) → cam(38104)  [RST] TTL=45, seq=1
china(http) → cam(38104)  [RST] TTL=45, seq=1
china(http) → cam(38104)  [SYN, ACK] TTL=37
cam(38104)   → china(http) [RST] TTL=64, seq=1
china(http) → cam(38104)  [RST] TTL=49, seq=1
china(http) → cam(38104)  [RST] TTL=45, seq=3770952438
china(http) → cam(38104)  [RST] TTL=45, seq=1
china(http) → cam(38104)  [RST] TTL=45, seq=1
china(http) → cam(38104)  [RST] TTL=37, seq=1
china(http) → cam(38104)  [RST] TTL=37, seq=1
```

Dealing with this new firewall strategy is more difficult than dealing with the forged reset packets. The problem is that even if the client ignores the (entirely valid) reset from the server, it continues to have an incorrect understanding of the server's sequence number, and it cannot "synchronise" with the server to complete the three-way handshake and connect.

Of course if, as occasionally happens, the specious SYN/ACK from the firewall arrives *after* the SYN/ACK from the webserver, it will be ignored by the client and will not cause any confusion. The firewall still attempts to tear down the connection with forged reset packets but, just as before, ignoring these resets means that a blocked web page can still be viewed.

Deciding which of two incoming SYN/ACK packets is genuine is clearly essential. In the examples we saw they were easy to distinguish, the firewall version had a distinctive TTL value, no DF flag, and no TCP options were set. They are therefore, at present, just as easy to filter as resets and the Chinese firewall is once again ineffective. Moreover, this strategy is only used once an attempt has been made to block a previous connection, and hence the expected TTL value for the server could be remembered by the client, whereas the firewall will not know what value to place into its forged packet.

However, with increasing sophistication in the firewall, it might manage to forge SYN/ACK packets with no detectable differences. The client could simply take the view that the firewall packet was the one arriving first. However, if the firewall countered this by sometimes delaying its SYN/ACK packet (allowing a naïve system to get access, but defeating a more sophisticated system!) then a complex "game" could result with ever more abstruse strategies. It should be noted that webpage fetching often involves multiple connections and so the

firewall operators might feel that they had "won" the game by blocking a proportion of accesses, rather than all of them.

An effective client strategy (with the prerequisite that both client and server are discarding resets) is to arrange to treat all incoming SYN/ACK packets (the firewall might in future send more than one) as valid. The client should then record their sequence values and ACK all of them. The client then continues to consider all values to be potentially correct (holding appropriate state within the TCP stack) until it receives an ACK from the server that confirms which value is actually correct. This is somewhat complex to achieve and beyond the capabilities of simple packet-filtering systems such as `iptables` or `ipfw`.

A further round of this new "game" would be for the firewall to forge an ACK for all of the client's packets. It should be possible for the client to see through this subterfuge by discarding values for which a genuine looking RST is received from the server, so the firewall would need to forge these – and once again the strategies can become arbitrarily complex. The endpoints do have an advantage in that they can eventually conclude whether packets are being generated by other, stateful, endpoint or by a stateless firewall. However, should the firewall start to keep "state" then this major architectural change (albeit almost certainly at significant cost) would open up many other strategies, and the advantage would swing decisively to the firewall.

Unfortunately, it must be noted that firewall generated SYN/ACK packets cannot be securely dealt with by a change to the TCP/IP stack at the server end of the connection. The server is in a position to work out that the client is continually responding with the "wrong" ACK value and retrospectively alter its own state to correspond with the value from the forged SYN/ACK packet. However, doing this would permit access by systems that forged the source IP addresses so as to pretend to be another machine [1].

Making secure connections in the presence of adversaries that can "sniff" packets and add forged packets of their own has of course been well studied in the context of cryptographic key exchange protocols. The open question is to what extent fairly simple modifications to existing TCP/IP stacks will continue to be sufficient to overcome the strategies available to the Chinese firewall operators, given the architectural limitations of their current design.

6 Denial-of-Service Attacks

As we have already noted, a single TCP packet containing a request such as `?falun` is sufficient to trigger blocking between the destination address and source address for periods of up to an hour. If the source of the packet is forged, this permits a (somewhat limited) denial-of-service attack which will prevent a particular pair of endpoints from communicating. However, depending upon their motives, this might be sufficient for some attackers. For example, it might be possible to identify the machines used by regional government offices and prevent them accessing "Windows Update"; or prevent a particular ministry accessing specific UN websites; or prevent access by Chinese embassies abroad to particular Chinese websites "back home".

Our calculations suggest that the denial-of-service could be reasonably effective even if operated by a lone individual on a dial-up connection. Such an individual could generate approximately 100 triggering packets per second, and hence – assuming that blocking was in place for the average period of 20 minutes – some 120 000 pairs of end-points could be permanently prevented from communicating.

Of course, current denial-of-service attacks are seldom instantiated by single dial-up machines, but by large numbers of machines on much faster connections. Hence the 120 000 value can be multiplied up to taste. However, it may well be that the IDS components of the firewall do not have the ability to record substantial numbers of blocked connections – so the actual impact is likely to be limited by this type of resource consideration. It should also be noted that while the IDS is handling an attempted denial-of-service attack it will have fewer resources to devote to recording information about other connections – thereby temporarily reducing its effectiveness.

6.1 Limitations on the Denial-of-Service Attack

Further experiments showed that the firewall's blocking was somewhat more complex than we have explained so far; and hence a denial-of-service attack would not necessarily be quite as effective as it initially seemed.

Firstly, the blocking is only applied to further connections with similar port numbers. The algorithm being used by the firewall only blocks the 128 TCP port numbers whose most significant 7 bits of value match the connection that triggered the blocking. For a system such as Windows that uses ephemeral port numbers sequentially this would mean that an average of 64 further connections would be blocked (therefore occasionally, if a port number such as 4095 was used in a triggering connection, there would be no further blocking). Conversely on a system such as OpenBSD which uses ephemeral port numbers pseudo-randomly, then the chance of another connection being blocked is only about 1 in 500.

We do not have a definitive explanation as to why the firewall behaves this way. It would seem much simpler are more effective to just block every connection to the same endpoints, without worrying about the port number.[5] It is possible that the aim is to avoid penalising other users of Network Address Translation (NAT) devices, when just one user has been blocked, or it may that the port number helps determine which particular IDS machine is given the packet. However, it may just be that the behaviour is meant to appear mysterious – and hence more menacing.

From the point of view of a denial-of-service attacker, the consequence is that all possible port number ranges must be blocked, unless there are special circumstances which allow the attacker to guess which ephemeral port numbers will be used in the near future. This increases, by a factor of about 500, the number of packets that must be sent to ensure one machine is blocked.

Secondly, not all IP addresses had their traffic inspected. Every hour we sent a rapid burst of requests containing "`?falun`", one packet from each of a block

[5] HTTP traffic was blocked not only on `tcp/http` port 80, but also on other port numbers. However, only a single server port was ever blocked – no adjacent ports were affected – nor was `tcp/https` (port 443) blocked when port 80 was.

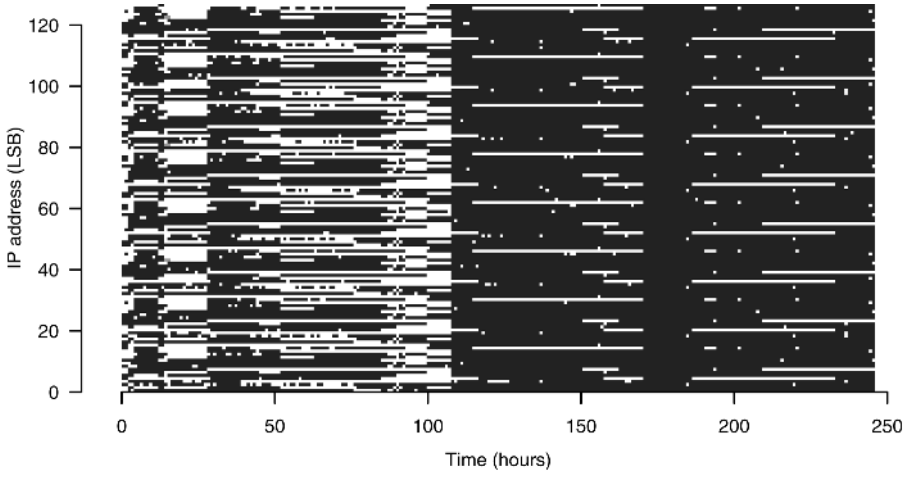

Fig. 1. Blocking of "bad" strings by the Chinese firewall. We tested from 256 adjacent IP addresses once an hour for 10 days in early February 2006. Results for the first 128 are shown; the pattern was very similar for the others. The dark blobs indicate that the access was blocked, and white that there was no blocking. When the result was indeterminate (no response at all) the colour is a mid-gray. An obvious change in firewall configuration (to block more IP addresses) is visible after 110 hours.

of 256 consecutive IP addresses. Initially, about two-thirds of each set of packets were blocked, with the address selection varying over time. However, after a few days, almost all packets caused blocking behaviour. We were unable to reverse-engineer the algorithm that determined which IP addresses had their packets scanned, although distinctive patterns (see Figure 1) within the IP address selections strongly suggest that quite a simple mechanism has been deployed. The most likely explanation is a lack of resources – two-thirds of the traffic may be all that the content scanning system can handle. Clearly, if a proportion of machines are being excused packet inspection at a particular time, then at that time, it will not be possible to mount a denial-of-service attack on them.

Finally, we observe that these experiments, as is the case with all the experiments we made, were performed using a small number of endpoints both outside and within China. Although we saw reasonably consistent results, with a system as complex as the "Great Firewall of China" it is entirely possible that we failed to observe significant aspects of its behaviour. Hence, although we believe that a denial-of-service attack may succeed in many circumstances, we cannot say that an attack on an arbitrary pair of endpoints would succeed.

7 Strategic Considerations

In order for traffic to pass unhindered through the Chinese firewall machines it is necessary for *both* endpoints to ignore resets. Machines in the "rest of the world" that wish to be accessed from China should have no difficulty in arranging for

a reconfiguration. However, the individual at the Chinese end of the connection may not wish to install special software. Their difficulty is that the firewall may not only be blocking connections but also logging what it has done. This might lead to an investigation, and the specially installed software would be discovered and an unenlightened view might be taken of the motives for installing it.

The packet inspection capabilities of the Chinese firewall can also be evaded by the use of encryption. If the authorities detected encrypted traffic, perhaps by statistical analysis of the content, then the same problem of specially installed software would arise when the endpoint was visited. However, since encryption systems typically discard session keys, it might not be possible to demonstrate that the traffic had been, say, pornography rather than political speech. In the case where the firewall is breached by discarding resets, the content will be available to the firewall in the clear, so that the authorities could consult their logs and treat the two types of access differently. As a result, some might view discarding reset packets as having an advantage over the use of encryption.

The Chinese authorities might be forced to take a more tolerant view of the use of reset discarding software by their citizens if this was to become universally deployed, and the resets were discarded for completely unrelated reasons.

Other work on "software firewalls" has shown that TCP resets are routinely discarded with few side effects (see Section 4.7 of [4]). Their main purpose is to provide a rapid way of reporting that incoming traffic is unwelcome. However, if the remote machine is well-behaved then very little more traffic will arrive if the packets are simply ignored, rather than responded to with a reset.

Nevertheless, some people may not wish to discard every TCP reset, and an alternative strategy is possible.[6] At present, inspection of the TTL values provides a simple method of distinguishing the resets generated by the firewall from any resets sent by the other end of the connection. In particular, we note that Watson's reset attack [15], whereby third parties forge resets to close down connections, is usually resisted by careful validation of the sequence numbers of reset packets. Validating the TTL value in the reset packet to ensure that it is similar to the TTL value seen for the rest of the connection would improve the chances of spotting forged resets generally. One of the present authors has developed a 20-line patch for FreeBSD [16] that discards resets whose TTL radically differs from other incoming packets on the connection. Experience so far has been very positive. It is unlikely that other operating systems or "personal firewalls" would find it onerous to provide the same facility.

Of course, the Chinese firewall can be adapted to make the proposed method of circumvention harder to achieve – in particular, it could trivially ensure that the TTL value was correct on reset packets sent in the same direction as triggering packets, although getting it correct for resets sent in the other direction would be difficult because Internet routing is often asymmetric and so the firewall cannot expect to see both directions of traffic.

[6] In future the Chinese firewall might block connections with FIN packets rather than resets. Ignoring all FIN packets would upset normal operations; this alternative strategy would then be the more appropriate.

However, it will continue to be complex to arrange to remove packets from router queues (or even delay them until a decision on their content has been made). Unless packets can be prevented from reaching their destination, our basic method – of ignoring everything the firewall says – will continue to work.

A completely different firewall strategy would be to refuse to route any further packets to sites that have triggered the blocking behaviour. However, we have already noted that this may scale very badly, because it must be done "in-line" with the fast path through the routers – and of course, full-scale blocking would increase the effectiveness of the denial-of-service attacks we discussed above.

8 Conclusions

We have demonstrated that the "Great Firewall of China" relies on inspecting packets for specific content. When filtering rules are triggered, forged reset packets are sent to each endpoint of the TCP connection. However, the genuine packets traverse the firewall unchanged, and hence by ignoring the resets, traffic can be exchanged unhindered. Further connections to the same destination are also blocked (although only if closely related port numbers are used), but ignoring resets will continue to permit unhindered access.

This result will be of considerable significance to the Chinese authorities, who will presumably wish to strengthen their systems to fix the holes in their firewall, although as we have noted, this may not be especially easy to achieve.

However, the result may be of less significance to Chinese residents who wish to access content unhindered, because their activity can still be logged and investigated. Only if the ignoring of reset packets becomes commonplace will residents be able to claim that their firewall evasion was inadvertent. This is not entirely far-fetched because validating TCP resets to see if they have been forged is a reasonable precaution for TCP/IP stack vendors to take.

We have also shown that a side-effect of the blocking is the potential for a denial-of-service attack, albeit one that can only be used to attack particular pairs of endpoints. It is perhaps unsurprising that a blocking mechanism can be used to block things – but without adding significant amounts of "state" to the firewall we do not see an easy way to prevent attacks.

The results we have demonstrated are also relevant to other countries, institutions and enterprises that use similar reset mechanisms to protect their interests. They should carefully note that the blocking entirely relies upon the acquiescence of those who are being blocked. Smaller countries than China may run a greater risk of denial-of-service, because they are likely to have fewer endpoints within their borders, so the firewall may not run out of resources to store details of blocked connections before the effect becomes significant.

Acknowledgments

We wish to acknowledge the assistance of a Chinese national we will not name (and who was entirely unaware of the nature of our experiment, and whose web

pages contain no illicit material) in providing an extremely convincing practical demonstration of a theoretical idea. Richard Clayton is currently working on the spamHINTS project, funded by Intel Research.

References

1. Bellovin, S.: Defending Against Sequence Number Attacks. RFC1948, IETF, May 1996.
2. Carter, E.: Secure Intrusion Detection Systems. Cisco Press (2001)
3. Clayton, R.: Failures in a Hybrid Content Blocking System. Fifth Privacy Enhancing Technologies Workshop, PET 2005, Dubrovnik, Croatia, 30 May–1 June 2005.
4. Clayton, R.: Anonymity and Traceability in Cyberspace. Tech Report UCAM-CL-TR-653, Computer Laboratory, University of Cambridge (2005)
5. Dornseif, M.: Government mandated blocking of foreign Web content. In von Knop, J., Haverkamp, W. and Jessen, E. (ed): Security, E-Learning, E-Services: Proceedings of the 17. DFN-Arbeitstagung über Kommunikationsnetze, Düsseldorf 2003, Lecture Notes in Informatics, pp. 617–648.
6. Edelman, B.: Web Sites Sharing IP Addresses: Prevalence and Significance. Berkman Center for Internet and Society (February 2003) `http://cyber.law.harvard.edu/people/edelman/ip-sharing/`
7. King Abdulaziz City for Science and Technology: Local content filtering Procedure. Internet Services Unit, KACST (2004). `http://www.isu.net.sa/saudi-internet/contenet-filtring/filtring-mechanism.htm`
8. The OpenNet Initiative: Probing Chinese search engine filtering. Bulletin 005, (August 2004) `http://www.opennetinitiative.net/bulletins/005/`
9. The OpenNet Initiative: Internet Filtering in China in 2004–2005: A Country Study (June 2004) `http://www.opennetinitiative.net/studies/china/ONI_China_Country_Study.pdf`
10. The OpenNet Initiative: Internet Filtering in Burma in 2005: A Country Study (October 2004) `http://www.opennetinitiative.net/burma/ONI_Burma_Country_Study.pdf`
11. Postel, J. (ed.): Transmission Control Protocol: DARPA Internet Program Protocol Specification. RFC 793, IETF (1981)
12. Telenor Norge: Telenor and KRIPOS introduce Internet child pornography filter. Telenor Press Release, 21 September 2004. `http://presse.telenor.no/PR/200409/961319_5.html`
13. US District Court for the Eastern District of Pennsylvania: CDT, ACLU, Plantagenet Inc. v Pappert, 337 F.Supp.2d 606, 10 September 2004.
14. Villeneuve, N.: Censorship Is In the Router. (3 June 2005) `http://ice.citizenlab.org/?p=113`
15. Watson, P.: Slipping in the Window: TCP Reset Attacks. CanSecWest/core04 (2004)
16. Watson, R.: 20060607-tcp-ttl.diff. June 2006, `http://www.cl.cam.ac.uk/~rnw24/patches/`