Imperial College London
Department of Computing

SnowWall: A Visual Firewall for the Surveillance society

Madalina-Ioana Sas
ms6413@ic.ac.uk

June 2017

Supervised by Dr. William J. Knottenbelt

Submitted in part fulfillment of the requirements for the degree of
Master of Engineering in Computing of Imperial College London.

Abstract

In the past two decades we have seen a steady increase in the adoption of various technologies that fit into what can be described as a digital lifestyle. Sharing data, experiences and our most intimate thoughts has become second nature to most people connected to the Internet. And as market trends shift so did the approach that companies and governments have in regards to each individual's digital footprint. The uncharted wild west that used to be the internet of yesteryear has become a battleground where everyone is fighting over who knows the user better. Findings such as the Snowden and the Vault7 leaks have deeply shaken the *status quo* in privacy. This project revolves around giving the end user back insight and control over what they involuntarily share with the world.

We present SnowWall: a networking tool designed to provide insights and control into the networking activity on a Windows-based system. SnowWall interacts with the operating system, intercepts every inbound and outbound connection, provides information on the connection's state, lifetime, owning process, and most importantly, remote end point, such as geolocation and ownership information. SnowWall is a powerful tool designed to be user-friendly, which allows anyone to block unwanted connections with high-level firewall rules, such as blocking by country or by owning organization name. With an intuitive visual interface, we strive to shake the public apathy formed around the issue of user tracking. Furthermore, we have analysed the nature of the data being leaked and we bring some intriguing results.

Running SnowWall on multiple hosts and analysing the data has brought up many red flags. We have observed a regular user working on Windows 7, 8.1, and 10, and we have learned that Microsoft collects data on all of these systems, with the amount of connections which open in general growing exponentially as newer versions of Windows are evaluated. It appears that system processes create TCP connections to both Microsoft servers and advertisers, with a large amount of network bandwidth being dedicated to such activities. Furthermore, many popular programs that ship with Windows appear to send data to Microsoft, in the US as well as Ireland, Hong-Kong, Singapore, and the Netherlands. We suspect that other governmental institutions are also involved.

Contents

1 Introduction

1.1 Motivation

Privacy is a growing concern in today's technological landscape. Since the first disclosure of government surveillance programs in 2013, a study by the Pew Research Center [1] has evaluated the public opinion and reaction to the Snowden leaks. The study shows little attempts of the public to improve their confidentiality, and alarming oblivion and public apathy to the problem of privacy. While an amount of 87% of American citizens are expected to be aware of these surveillance programs, and some have labelled themselves concerned about these findings, 54% of them still have not changed their behaviours. This is because they believe that it would be difficult to find the required tools to do so.

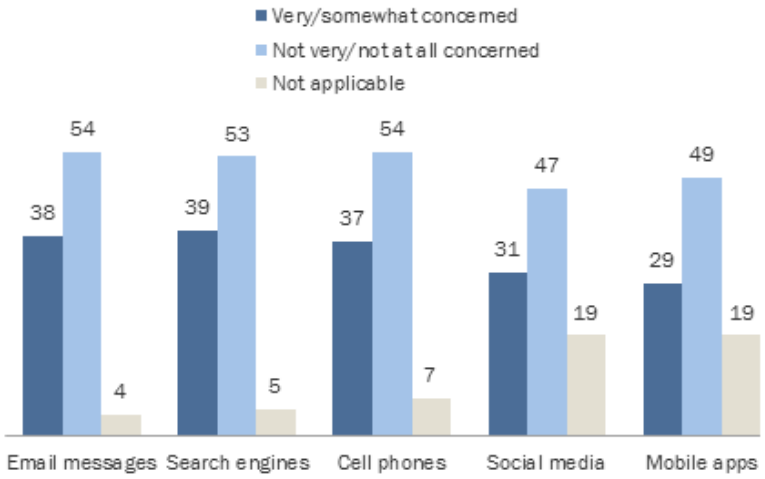


Figure 1: Survey shows less than half of Americans are "very or somewhat concerned" about the government surveillance of their electronic communications and personal data.

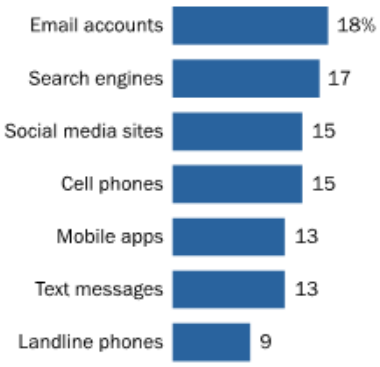


Figure 2: The percentage of users who have changed their behaviour to improve privacy is alarmingly small.

Additionally, other recent disturbing findings [2] make it easy to see how fragile is the confidentiality of our online activity due to vulnerabilities and irresponsible data collection from our software providers, which have become target to governmental espionage and malicious hackers aside.

As we are living in a society increasingly dependent on technology, we need to be aware of the trade-offs implied by the broad availability of free software services. On top of government surveillance, it appears that corporate surveillance is also an imminent problem, which is only aggravated by malicious intrusions into their software platforms. Aside to governmental espionage, data collection for the purpose of improving various services seems to invade, without permission, the users' private lives. Previous research has shown that on smartphones and browsers user settings are bypassed in order to collect data for purposes

such as browser tracking of user's internet activity [4], microphone tracking [5] and location tracking [3] for targeted ads.

Even though many publications have been written about user tracking on mobile platforms, to the best of our knowledge, less work has been focusing on similar intrusions on desktop systems, especially recent versions of Windows. The Windows Phone 8 system, which shares its kernel with Windows 8 Desktop, has been evaluated in previous work [3] to leak location information to third-party apps, even with location services off. This made us suspect that Windows 10, similarly designed as a universal platform which can support both desktop and mobile and maintains some components from it's predecessor, is also susceptible to user tracking. Moreover, many users have complained about personalised or location-specific ads in their Store or calendars, Cortana unexpectedly turning on, and difficult to find and manage privacy settings. The European Digital Rights organization (EDRi) sums up Windows 10's prohibitive 40 pages of terms and conditions as: "Microsoft basically grants itself very broad rights to collect everything you do, say and write with and on your devices in order to sell more targeted advertising or to sell your data to third parties." [6]

Furthermore, since the release of Windows 10, Microsoft turned the Windows Experience Program (dubbed Feedback & Diagnostics) from an optional programme into a service which cannot be fully disabled [7], forcing the users to agree for telemetry to be collected and sent over to Redmond, Washington. Resolving the tensions between a better experience and privacy risks poses a great challenge for software platforms. Windows 10 has only recently introduced more fine-grained privacy settings with it's Creator's Update [8]. Not only is there no guarantee these settings are transparent, but also tasks which may improve privacy, such as creating a firewall rule or monitoring network connections still remain obfuscated to the basic user.

And even though many users have been unsatisfied with Windows 10 and have rolled-back to Windows 7 and 8, rather than allow more flexibility in the data collection, Microsoft have recently introduced similar tracking and telemetry tools in their updates for these systems. [9] [10]

Given the current landscape, it is in our most interest to provide tools that aid the non-expert user into protecting their privacy against both governmental and corporate surveillance. While we are aware data collection is crucial for improving technological services, issues such as imperfect anonymisation of data and bypassing user's permissions need urgent tackling.

1.2 Objectives

The goal of this project is to provide the user a simple and powerful tool to monitor the activity and control their privacy on a Windows operating system, improving on the current state-of-the-art monitoring and management tools that exist for Windows platforms.

Using this tool, we strive to identify where is the user's data being sent, what's the nature of this data, who is it shared with, and how transparent are privacy settings on various Windows systems. We hope this will make users more aware of their privacy as well as unmask unethical tracking attempts, and that the conclusions of this work will aid future research in automatically identifying and stopping possibly malicious privacy breaches.

The main objectives of this work can be summarised as following:

- To allow a user to monitor network traffic, bandwidth and process activity in real-time on a Windows-based PC
- To provide accurate and comprehensive real-time visualisations of the monitored activity, including information about the destination of network traffic
- To build a mechanism which can filter this network traffic according to different dimensions, such as by destination country, organization, or originating application
- To provide easy access to creating and scheduling firewall rules required for the aforementioned policies
- To provide easy access to obfuscated privacy settings
- To demonstrate the applicability of this tool in a various case studies

1.3 Contributions

The aim of this project is to build a monitoring tool for identifying and blocking unwanted network connections, as well as surveying the current threats to privacy on a Windows system. This goal has been achieved by the development of SnowWall, a Microsoft .NET Windows Forms application, compatible with Windows 7 and above. The main contributions brought by the development of this tool are outlined below.

Monitoring. Monitoring tools for Windows do exist[1], but few are user-friendly, mostly being directed to security professionals, and even fewer compile a broad range of tools into the same platform. SnowWall interfaces with the Windows IPHelperAPI to query the kernel for TCP and UDP network connections, statistics and bandwidth, with the Win32API to query running processes, and the FirewallAPI to interface directly with the Windows Firewall. The activity is monitored in real-time and logged to a SQL database, which may prove useful in *a posteriori* forensic analyses. To our knowledge, SnowWall is the first tool to reunite all of these features.

Visualisations. SnowWall uses client-independent IP geolocation and a survey of ISP data to allow the user to visualise the geolocation information of TCP connections on the world map, as well as a conceptual map of organizations and applications. Statistics about bandwidth and connectivity can be accessed through the app, as well as previewed as performance graphs in real-time.

[1] `https://technet.microsoft.com/en-us/sysinternals`

Firewall Control. Straight from the interactive visualisations, SnowWall allows a bespoke interface for communicating with the Windows Firewall. The user can create scheduled, high-level rules, such as to block connections geolocated in a specific part of the world, or belonging to a specific organization, and SnowWall will automatically translate these into Windows Firewall rules.

Additionally, SnowWall can be successfully used as a productivity tool, due to the ability to schedule firewall rules, as well as an out-of-browser ad-blocker and anti-tracking tool, capable of bypassing ad-blocker checks performed by websites [11].

Applicability. SnowWall has been tested by average as well as advanced users of the Windows platforms and it has proved itself of great value in security evaluation work performed on enterprise hosts.

Finally, multiple computers running popular versions of Windows and Windows-based software have been investigated for their privacy status, and the application logs have been analysed to provide insights into the most invasive software and organizations.

1.4 Thesis outline

The rest of this thesis is organized as follows:

Section 2 presents the theoretical knowledge underlying the development of SnowWall, and a survey of existing products and tools similar with ours. A basic introduction into Windows networking and firewall is provided in Section 2.1 and Section 2.2, as well as a quick review of IP geolocation methods in Section 2.3. A survey of existing research work on privacy issues in Windows-based PCs and the results of evaluating various information leaks from both PC and mobile Windows systems can be found in Section 2.4. We include a short description and evaluation of existing tools which offer comparable features with SnowWall and conclude the improvements it brings with respect to the existing options in Section 2.5.

Section 3 presents the design and architecture of the program, including insights into some of our human-centered design research and the main principles which have shaped the existing design of the product.

Section 4 describes the more challenging details of the development work, starting with the back-end in Section 4.1 and database 4.2. We then proceed to discuss front end development. Finally, we conclude this section with the main implementation issues, challenges and risks and how they have been overcome.

Section 5 focuses on the evaluation of the product, from user experience and interaction, compatibility, optimisation and accuracy of geolocation.

Section 6 focuses on displaying and discussing the results of the case studies. We include analyses of the data gathered from running SnowWall on Windows 7, 8.1, and 10

systems and the most popular applications which are at risk of compromising the user's privacy. Using the traffic monitoring test-bed provided by SnowWall, we have evaluated the transparency of privacy settings and the hidden activity and traffic of a Windows system.

Section 7 presents conclusions, possible extensions to the current stack, a legal discussion and future work.

2 Background

This chapter introduces the theoretical background underlying the development work undertaken in this project, as well as existing research related to user privacy on Windows platforms, and existing tools and software which provide similar functionality with ours.

We begin by introducing the main concepts of networking theory required by this work, focusing on Windows and the specific network architectures relevant to this project. We proceed by discussing the notion of a firewall, define and present the current state-of-the-art in application firewalls, as well as specific details about Windows Firewall, its functionality, and relevant implementation details. We then present a survey of techniques for IP geolocation, concentrating on the most relevant ones for the task at hand, specifically latency measurement-based geolocation and data-driven methods, which allow inferring geolocation or IP ownership information from DNS records and ISP data. An analysis of existing literature on monitoring network activity in Windows and research on privacy breaches is also provided. Finally, the chapter concludes with a survey of the state-of-the-art monitoring tools, discussing their advantages and disadvantages and motivating the need for a new, more comprehensive such tool.

2.1 Networking

This section provides a short introduction into the underlying layers on the internet, defines the relevant network protocols and introduces concepts discussed in further sections with respect to firewalls and geolocation.

2.1.1 The OSI Network model

The Open Systems Interconnection (OSI) model [17] is a standardised, conceptual model for the purpose of describing system interconnection. The model applies to telecommunications and computer systems. Its design partitions every communication system into *layers* of abstraction.

Each of the layers serves only the one above and is facilitated by the services provided only by the one beneath. Each layer specifies its own *protocols* for entities at the same layer but in different hosts to communicate.

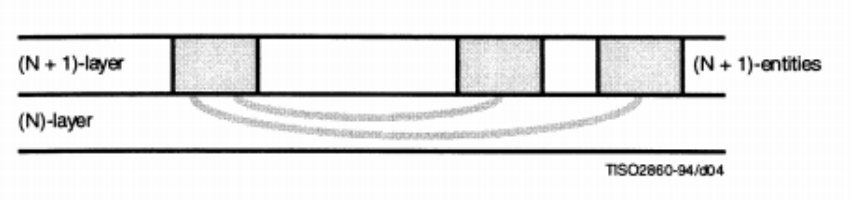


Figure 3: Entities in the (N+1)-layer communicate through the N-layer beneath.

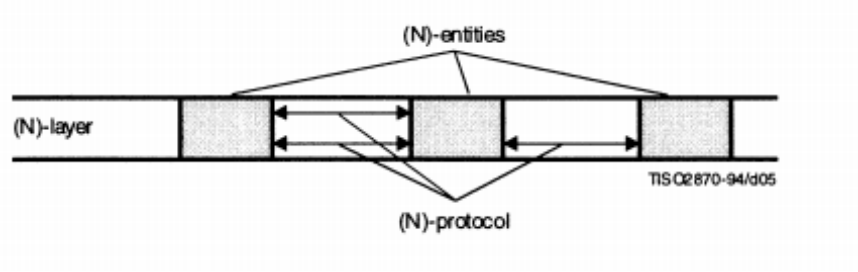


Figure 4: N-entities use the N-protocol to communicate.

The reference model presents 7 layers:

- **Application Layer**: at the top of the layered model, the application layer is the closest to the end user. The application layer is usually responsible with identifying available peers to transmit data from the application, and to verify whether the necessary network resources to fulfill this communication are available. Examples of protocols functioning at the application layer are File Transfer Protocol (FTP) and Simple Mail Transfer Protocol (SMTP).
- **Presentation Layer**: this layer is responsible with formatting the contents of the communication in an unobfuscated way to the application layer. Encryption is performed at this layer, as well as serialisation of data structures to and from XML, processing image and video codecs, e.g. JPEG, PNG, MPEG, GIF.
- **Session Layer**: this layer manages the connections between hosts, allowing the presentation-layer entities to synchronise and manage their dialogue and their data exchange. For this purpose, this layer defines procedures for establishing a communication, exchanging data in a controlled manner, and gracefully terminating or restarting the connection. This layer is concerned with sockets, such as Unix's IPC sockets or Windows's WinSock.
- **Transport Layer**: provides the means of transferring data of variable length across the network from a source to a destination entity in an optimal manner, while maintaining the quality of this transmission and providing with error correction. The Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) operate at this layer.
- **Network Layer**: provides the means of transferring packets, i.e. data sequences of variable length in order to deliver the message from the sender to the destination node by possibly routing it though intermediate nodes in the *network*. If the message is too long for the underlying data link layer, the network layer may send the data split into fragments and reassemble them at destination. The network layer does is not guaranteed to be reliable or to report delivery errors. The Internet Protocol (IP) and Internet Control Message Protocol (ICMP) operate at this layer.
- **Data Link Layer**: provides data transfer from nodes which are directly connected or linked to each other, defining the protocol to establish and terminate connections between them. It provides functionality to detect and potentially correct errors in the

underlying layer. Ethernet and Wi-Fi operate at this layer, and Media Access Control (MAC) addresses are defined at this layer.

- **Physical Layer**: the lowest, or first level of the model, it defines the electrical and physical specification of the connection. This layer generally refers to low-level networking equipment such as network adapters, modems, and repeaters. It is never concerned with protocols.

Following this model, communication between two hosts is done as following:

- the data to be transmitted is composed at the topmost layer, where it is passed to the layer below.
- at each layer below, the data is processed and concatenated with a header and/or footer, until it reaches the lowest layer
- at the lowest layer, the data is transmitted to the physical device meant to receive the communication
- at the receiving end point, the data is propagated up, at leach layer stripped of headers and footers, until reaching the topmost layer
- at the topmost layer, the data has finally reached its destination, and can be used by the receiver

2.1.2 The Internet Protocol Suite

The Internet protocol suite (the TCP/IP model) is a conceptual model and standard set of protocols which define the usage of the Internet [18]. The Internet protocol suite defines the specification of an Internet communication, from addressing, routing, transmission and packet fragmentation. Similar, but simpler than the OSI model, the Internet protocol suite also divides the functionality of an Internet communication system into layers of abstraction.

The model is hardware agnostic and presents 4 layers, which are more flexible than the strongly hierarchic layers of the OSI model:

- **Application Layer**: the topmost layer, closest to the end user. This layer provides user interfaces for exchanging data over the network connection established between two hosts by lower-level protocols. Unlike the OSI model, in the TCP/IP suite there are no layers specifically concernted with presenting the data or managing sessions; most of these features are seen as support protocols or functions programmed into the application. Examples of protocols running at the application layer include HTTP, FTP, SMTP.
- **Transport Layer**: concerned with propagating transport-layer protocol packets produced by the application layer to the target host. This layer provides end-to-end connectivity, running on end hosts only and not on intermediary network nodes e.g. routers. It is only concerned to message transfer, independent of the underlying layers, and it only relies on assumptions about them such as the uniqueness of IP addresses.

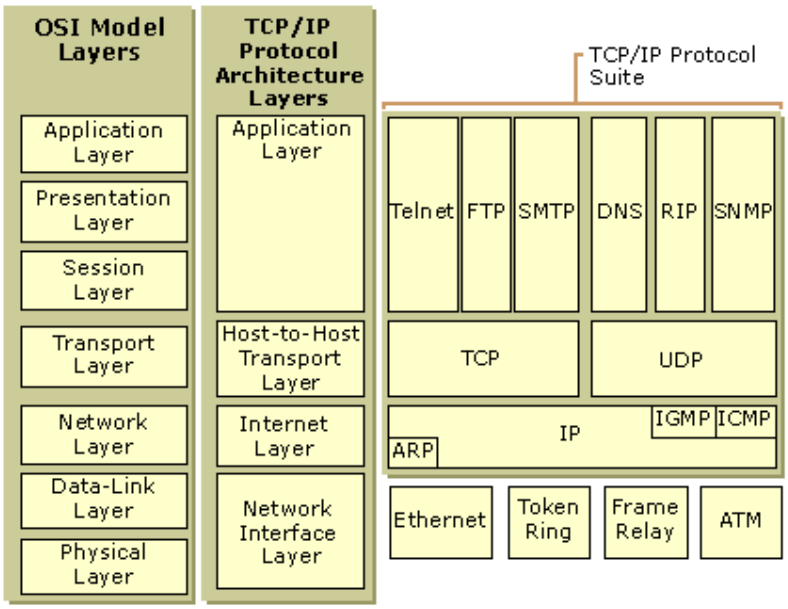


Figure 5: The relationship between the layers defined by the OSI model and the layers defined by TCP/IP, as well as the protocols defined by each layer in the TCP/IP protocol suite (courtesy of Microsoft TechNet).

This layer provides packet fragmentation, error correction, optimisation, ordered delivery when necessary, congestion control, error control and correction, as well as application addressing via *port numbers*. End-to-end message transmission provided by this layer can be either connection-oriented, stateful, e.g. TCP or connection-less, e.g. UDP. The TCP/IP transport layer is equivalent to the Layer 4 in the OSI model, with the addition that session management features defined in Layer 5 of the OSI model such as gracefully closing a connection are supported by protocols at this layer.

- **Internet Layer**: sends packets called IP datagrams across the network. This layer is responsible with host addressing and identification and packet routing, both implemented in the Internet Protocol (IP). The Internet layer is unconcerned with the data being sent or the underlying layer, and it does not distinguish between the types of protocols carried by IP e.g. ICMP. The layer provides unreliable transmission by forwarding transport layer packets to the next router until it reaches the destination. The original address system of the internet, known as IPv4, identifies hosts with 32-bit IP addresses. The newer IPv6 uses 128-bit addresses for scalability. The Internet layer is a subset of the 3rd layer in the OSI model, the Network layer.
- **Link Layer**: the lowest layer, supposed to move packets between the IP interfaces of two different hosts. This layer corresponds to the OSI layers 1 and 2, as well as includes some protocols from layer 3, but in its definition it is agnostic to hardware. This is also the layer where packets may be selected to be sent over a networking tunnel e.g. Virtual Private Network (VPN), even though such a tunnel might be a transport or even an application layer protocol. This is one of the reasons why the TCP/IP stack has flexible layers which are not hierarchically encapsulated.

2.1.3 Application layer protocols

Application layer protocols are generally associated with particular client-server software applications, and they run on specific well-known ports.

These protocols can be split in two groups [19], user protocols, which are used by user applications (e.g. FTP, SMTP, HTTP), and support protocols, which provide services to systems such as Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS). For the purposes of this project, we are mostly concerned with DNS and WHOIS.

Domain Name System DNS is a decentralized naming system for resources connected to the Internet. The main purpose of DNS is translating the human-readable internet addresses (hostnames) to IP addresses. DNS assigns each participating entity a domain name, as well as other information about that entity, including the IP addresses associated with domain names, and provides a distributed lookup facility into a database of *resource records* (RR).

DNS delegates responsibility of assigning domain names and mapping those names to Internet resources by designating *authoritative name servers* for each domain.
DNS is a *connection-less* protocol which runs on top of the UDP transport layer protocol. DNS has query and reply messages formatted in the same way, linked by an identifier.

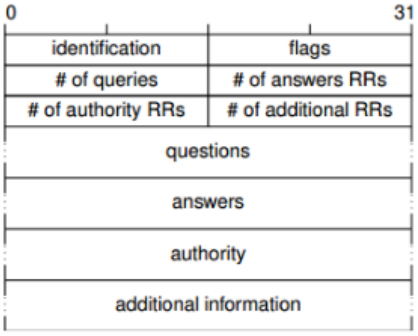


Figure 6: DNS message format

The DNS architecture is hierarchical, with its root at 13 servers known as *root servers*. [20]. These root servers contain several *top-level domain servers*: each one is associated with a top-level domain (e.g. `.com`, `.edu`, `.uk`, `.org`). For each domain, there is an *authoritative DNS server* that holds the map of public hosts within that domain.

A *reverse DNS lookup* is a query to a DNS name server where the IP address is known rather than the domain name. This is relevant to IP geolocation algorithms as they can make use of the hierarchy of domain names of various servers to identify geographical location.

WHOIS WHOIS [21] is a standardised protocol widely used for querying databases that store the registered users of an Internet resource, such as a domain name or an IP address block. WHOIS was originally implemented on the Network Control Program (NCP), but became relevant and widespread with the standard TCP/IP suite.

To query WHOIS, generally it is enough to query the name of the single resource. WHOIS

clients run on top of TCP, and are lightweight applications which simple establish communication channels to the WHOIS server, sending over queries and awaiting responses.

A WHOIS database contains text-based data for each resource. These text records include: resource information, owner or registrant information, and administrative information. Depending on the nature of this information and how it is stored, there are two data models or architectures for the WHOIS protocol:

- A *thick* WHOIS server stores the complete information for the particular resource in the same place, thus a query can be resolved by visiting a single server
- A *thin* WHOIS server stores a reference to the server of the registrar of a specific domain or resource, which is meant to hold the complete information about said resource. This is slower than querying a thick server as more than one server needs to visited. Furthermore, there may be inconsistencies in the data as for example an outdated registrar reference.

The WHOIS protocol has no means of distinguishing between these two models. Some top-level domains such as `.com` operate a thin server, whist others, such as `.org`, operate a thick model.

The WHOIS is not a fully accurate protocol since the data is still in control to the registrant, which is in no way forced to provide accurate information. Furthermore, in some cases privacy mechanisms may be available to allow a registrant to hide their WHOIS information from the public.

A standard WHOIS lookup generally queries the public number registry databases RIPE[2], APNIC[3], AFRINIC[4], LACNIC[5] and ARIN[6].

2.1.4 Transport Layer Protocols

The transport layer, for the purpose of providing specific transmission channels for different types of application, introduces the concept of *port number*.

Each connection is identified by the information about the two end-points, formed of IP address and port. The most popular applications have well-known port numbers established by the Internet Assigned Numbers Authority (IANA). While the application layer has access to the port information, ports are defined at this layer.

Transmission Control Protocol TCP is a connection-oriented protocol and the most used protocol of the Internet. TCP work with packets called *segments*. TCP opens up a

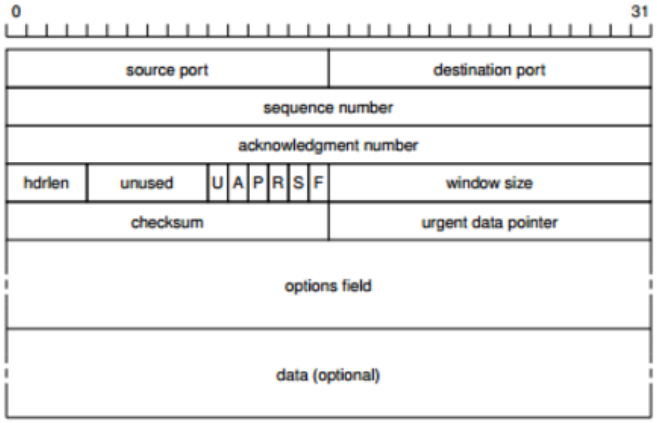
[2]European IP Networks `https://www.ripe.net`
[3]Asia-Pacific Network Information Centre `https://www.apnic.net`
[4]African Network Information Center `https://www.afrinic.net`
[5]Latin America and Caribbean Network Information Centre `https://lacnic.net`
[6]American Registry for Internet Numbers `https://www.arin.net`

Port number	Transport Layer Protocol	Application Layer Protocol
20	TCP	FTP
21	TCP	FTP
22	TCP	SSH
23	TCP	Telnet
25	TCP	SMTP
43	TCP	WHOIS
53	TCP, UDP	DNS
67-68	UDP	DHCP
80	TCP	HTTP
110	TCP	POP3
161	TCP	SNMP
443	TCP	SSL/TLS
16,384-32,767	UDP	RTP-based voice and video protocols

Table 1: Some most popular well-known ports.

connection between two end points, by binding their IP address and port to sockets. A TCP connection allows messages to be sent until it is closed. TCP is a full-duplex service, thus allowing both end-points to send and receive messages at the same time.

A *TCP segment* functions like an envelope for TCP data, such that TCP packets can be sent using the IP protocol. The sequence number and acknowledgement number are used by flow control to identify missing packets and provide ordered transmission. The checksum field is used to verify the integrity of delivered packets.



TCP provides reliable data transfer by implementing a guaranteed delivery mechanism using *flow control* techniques. Flow control determines whether data needs to be resent, and if it does then it stops communication until lost packets are transmitted again, until a packet identical with the original is received at the remote end point. Some of the properties of the TCP protocol are:

- *order*: data segments arrive in the same order as they were sent
- *correctness*: data has minimal error, made possible by checksum mechanisms; flow control deals with lost or discarded packets and duplicate data
- *traffic congestion control*

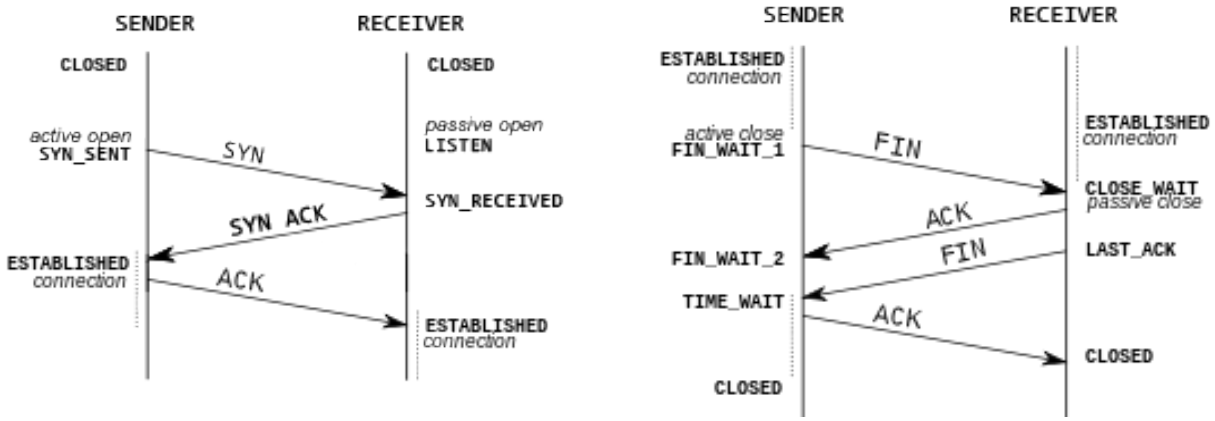


Figure 7: TCP works by sending messages which request acknowledgement of packets being sent. For example, to establish a connection, the connection request `SYN` message is sent, and the connection is established when the receiver confirms with a connection establishment acknowledgement `SYN ACK` which is in turn confirmed by the sender with an acknowledgement `ACK`. This is known as a three-way handshake. A similar flow is used for closing a connection, but with a four-way handshake, implemented with termination request `FIN` messages and `ACK` messages.

Since TCP is a connection-oriented protocol, depending on the current step in communication a connection can be in multiple states.

User Datagram Protocol UDP is a connectionless protocol, less reliable than TCP, but which prioritises efficiency. UDP is commonly used for streaming media, when speed is more important than reliability, or for simple request/response services which do not require the overhead of establishing and closing a connection. UDP works with packets called *datagrams*.

UDP is a best-effort protocol like IP, and uses a weak checksum algorithm to address reliability. The lack of flow control or error correction allows this protocol to focus on timely delivery of data. The *UDP datagram* has a much simpler header than TCP: it includes no flags, only a checksum used to verify if packets have been transmitted.

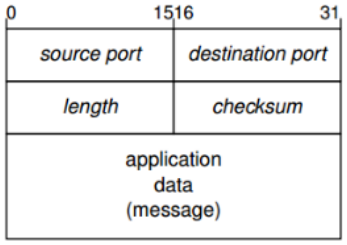


Figure 8: The UDP datagram

2.1.5 Internet Layer Protocols

Internet Protocol IP is the main protocol operating at the Internet Layer. Like UDP, IP is concerned with sending datagrams over the network, and does not guarantee reliability.

LISTEN	waiting for a connection request from any remote endpoint
SYN-SENT	the first step of the three-way handshake has been performed, a connection request was made; waiting for a matching connection request from the receiver
SYN-RECEIVED	the second step was performed; waiting for an acknowledgement from the receiver after having both received and sent back the matching connection request
ESTABLISHED	the three steps of the handshake have been completed: an open connection allows full-duplex data transfer from both sides
FIN-WAIT-1	the first step of the four-way handshake has been performed, i.e. a termination request has been sent; waiting for a maching termination request
CLOSE-WAIT	a connection termination request has been received and an acknowledgement sent - an incomplete handshake called a *passive close* has been performed
FIN-WAIT-2	the second step of the four-way handshake has been performed, the acknowledgement has been received; waiting for a matching termination request
CLOSING	the matching termination request has been sent; waiting to receive acknowledgement
LAST-ACK	the passive close has been completed and a connection termination request is sent to perform an *active close*
TIME-WAIT	after sending the last acknowledgement, wait for twice the maximum segment lifetime to make sure the end-point received the acknowledgement before closing
CLOSED	there is no connection on the given port

Table 2: The states of a TCP connection.

The *IP datagram* wraps messages into headers required by the protocol. It contains the IP version, protocol information (e.g. ICMP), checksum for verifying integrity, the source and destination IP addresses and fragmentation information. The latter allows the network device to split a large data packet into multiple fragments if it exceeds the *maximum transmission unit* (MTU).

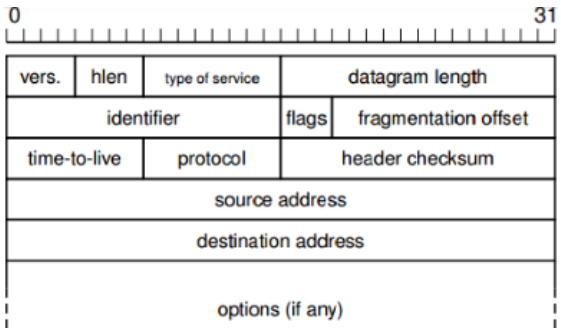


Figure 9: The IP datagram

Internet Control Message Protocol ICMP is a support protocol carried by IP, used to send diagnostic information. ICMP messages are contained inside standard IP packets. It is supported by all network devices and it is used to send error messages or communicate information such as unavailability or unreachability of hosts or routers. ICMP is not used

to exchange data between systems and generally is used by very few user-facing applications outside of diagnostics tools.

ICMP can be used for monitoring the *time-to-live* (TTL) of IP packets. Every node forwarding an IP datagram decrements the TTL field in the IP header, until the TTL reaches the end of its lifespan and the packet is discarded. In this case, ICMP is responsible with sending a notification to the originating host that the packet has exceeded its TTL.

`ping` is a software utility used to test the reachability of a host on the Internet. `ping` operates by measuring the *round-trip time* (RTT) for a ICMP Echo Request sent from the source host to a target and echoed back to the source.

`traceroute` is a network tool which displays the route (path) and transit delays of packets across a network, recorded as the round-trip times of the packets received from each successive host in the route. Each time a packet is forwarded to the next network resource, a *hop* occurs. Unlike `ping`, which only shows the total time spent to establish the connection, the `traceroute` command shows the delays at each hop. `traceroute` can be implemented by transmitting IP packets with specific values in the TTL header fields, then verifying if any "Time to live exceeded in transit" and "Destination unreachable" messages are generated as a response.

On Windows-based systems, `traceroute` and `ping` are both implemented to send ICMP Echo Probes.

2.2 Firewall

Before 1988 and the Morris Worm, the young Internet knew no concept of firewall. The worm started as the project of Robert Tappan Morris Jr., a student at Cornell University and the son of the head of the National Security Agency. The project was meant to exploit vulnerabilities found by Morris throughout his research as a proof of concept of the fragility of the Internet [24]. The worm expanded uncontrollably because of a design mistake which allowed it to install multiple times on a victim machine, eventually leading to a fork bomb [26]. He was the first person to be convicted for felony for computer abuse. Known as the "Internet Virus", this was the first mainstream attack on internet security, and was the attack that inspired the idea to protect a system with a means to filter passing traffic.

This section begins with defining the firewall, and provides a short survey over types of firewalls and how the firewall developed into the software product it is today. We continue by reviewing various firewall architectures and finalise by introducing the Windows Firewall and motivating why interfacing with the Windows Firewall is good enough for the purposes of this project.

2.2.1 Definition

A *firewall* has been defined in the literature [23] as a system that satisfies the following criteria:

- it is situated at the boundary between two networks
- all traffic between the two networks must pass through the firewall
- it must be able to filter the traffic between the two networks by enforcing *policies*, i.e. sets of rules which either allow some traffic to pass, assuming the other traffic is blocked

There are three main types of firewalls, as defined by the National Institute of Standards and Technology [25]: packet filters, stateful firewalls, and proxy-based firewalls.
The development of the three types of firewalls has been an incremental roadmap through security vulnerabilities and network architectures. The following section details their history and features.

2.2.2 History

First-generation firewalls. The first firewalls were *packet filters* [27], network-level or physical-level devices which filter network traffic packet by packet based only on information contained inside each network frame. Generally they were implemented using a *screening router*. Packet filtering (or screening) most frequently uses information of the packet's transport layer protocol and related flags, source and destination address, whether it is inbound or outbound, and, for TCP and UDP traffic, the port number. Therefore, they

could also interact with the transport layer to query port numbers. Work using packet-filtering technology became the first commercially available firewall. [28]

The main disadvantage of packet filters was that they were stateless, thus unable to monitor the subtleties of a TCP connection state or the pseudo-state occuring in some cases with UDP connections. Therefore a malicious remote host could send in spoofed packets, which would appear to be part of an already-established TCP connection (i.e. with the TCP `ACK` flag set). This would give an attacker the possibility to map the local network as if the firewall did not even exist. [29]

Another issue with stateless firewalls was caused by the FTP protocol. FTP needs to open a channel of communication from the client to the server in order to function correctly. This is generally initiated from the server's TCP port 20 to a random, generally high, client port. But there is no guarantee that all traffic originating on port 20 of the server is non-malicious, thus packet filters are not safe to assume that all traffic originating in TCP port 20 is a FTP transfer.

Second-generation firewalls. In order to mitigate these disadvantage, *stateful firewalls* or *dynamic packet filters* have been introduced [30]. Stateful firewalls work at both the network layer and transport layer: the filter monitors for the sequence of TCP packets with the `SYN` flag set and stores them in dynamic state tables until the FIN packet is sent and acknowledged. This allows firewall rules and policies to also use the connection state as a filtering criteria, referred to as *shallow packet inspection*, since it inspects the IP header of the packet, as well as the transport-layer header, but doesn't go deeper into the packet payload. The filter acts on the accumulated data, rather than on discrete packets, thus is capable of intercepting spoofed packets and block them.

Third-generation firewalls. The current standard in firewall technology are *application firewalls*, also known as *reverse-proxy* or *proxy-based* firewalls. They are implemented at the application layer, allowing them to use information from the application and protocol in their filtering, and control applications and services. For example, an application firewall can distinguish between FTP and HTTP traffic. These firewalls can detect if an undesired service or application is attempting to bypass into the network, allowing much finer-grained access control, such as knowing when a protocol is being exploited or a malicious service tries to run from a safe host and port. Application firewalls support network and port-address translation and can be used together with virtual private networks [31].

Next-generation firewalls, considered the state-of-the-art, perform a *deep packet inspection* (DPI) of the application stack, which is capable of examining the entire application stack (from the data link layer up to the application layer) to detect malicious uses of any of the communication protocols [33]. DPI looks for malicious signatures in the entire packet, and can identify viruses, spam, worms, or protocol non-compliance. DPI is effective against denial-of-service and buffer overflow attacks, as well as the few worms which fit within a single packet. One of the common ways of implementing DPI is through *port mirroring*. Port mirroring is a feature implemented in a network switch allowing it to send a copy of

all network packets to another network-monitoring connection running on another port, which then allows for the packet's contents to be analysed and filtered based on complex rules.

Next-generation firewalls combine the features of a traditional third-gen firewall with an *Intrusion Detection System* (IDS) and an *Intrusion Prevention System* (IPS). Techniques other than DPI may be used, such as inspecting encrypted SSL/TLS traffic, working at web-application layer to filter websites, inspecting packets using an antivirus-style filter, and integrating identity management features such as Microsoft's Active Directory services or the open Lightweight Directory Access Protocol (LDAP).

2.2.3 Architecture

Firewalls can be implemented as both hardware and software devices. Firewalls can either be *network firewalls*, implemented as a set of network devices and configurations of hosts which encapsulate an internal network, or be *host-based*, like the firewall distributed with most operating systems. Generally, the first ones are preferred in cloud setups and for enterprise purposes, whilst the latter cater more to the single users.
We briefly introduce network architectures, but in this work we mostly focus on host-based firewalls, since they form the extreme majority of firewalls used by non-professional consumers.

Network firewall architectures. Network firewalls serve the protection of a local trusted network from an external network. Depending on the topology of the two networks, there are three main architectures for network firewalls, which can be previewed in Figure 10.

(a) *Dual-homed host architecture*: revolves around a dual-homed host, which has two network interfaces allowing it to connect to both the local and outside network; this host blocks all IP traffic between the two networks, acting as a proxy between the two and filtering undesired traffic based on the firewall policies

(b) *Screened host architecture*: provides services from a host, called *bastion host* (, that is only attached to the internal network, and is separated from the exterior by a screening router. In this architecture, the router provides the security by packet-filtering.

(c) *Screened subnet architecture*: offers another layer of security to the screened host architecture by adding another network in between the local network from the Internet. This network, known as a perimeter network, is separated by two screening routers. This configuration minimises the risk over a malicious attack over the bastion host, and only allows access inside the perimeter network in case the bastion host has been compromised.

Host-based firewall architecture. The motivation for host-based firewalls is that network firewalls protect an internal network, but not the individual hosts inside this network. Therefore, if a worm or malware manages to pass the network-based firewall, the individual hosts cannot protect themselves against this threat. Host-based firewalls protect individual hosts from unauthorised access including from traffic generated inside the trusted local

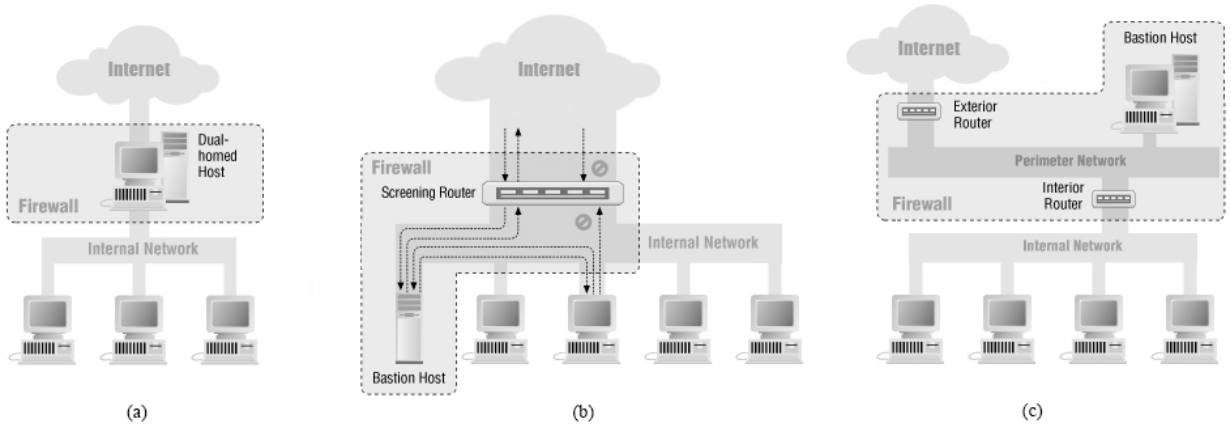


Figure 10: A visual comparison of network firewall architectures, courtesy of [34]. (a) Dual-homed host: the two networks interface via a dual-homed proxy. (b) Screened host: the local network connects to the bastion host, which is separated from the outside by a screening router. (c) Screened subnet: the local network is separated from the outside by a perimeter network.

network. Most host-based firewalls are two-way i.e. support filtering both inbound and outbound traffic.

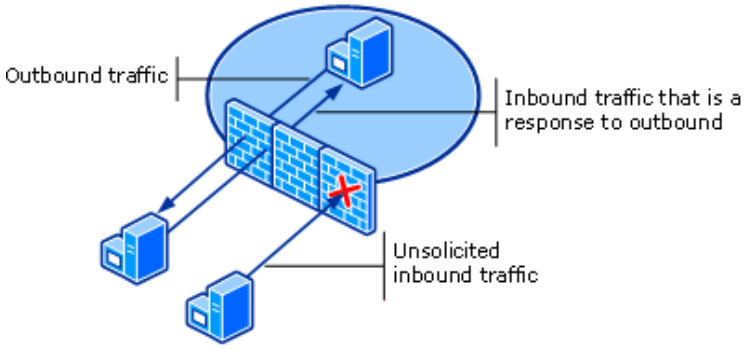


Figure 11: The architecture of a host-based firewall (courtesy of Microsoft TechNet)

Linux systems provide a kernel package for configuring a host-based firewall. The Windows Firewall is provided by default with Windows since Windows XP Service Pack 2.

2.2.4 The Windows Firewall

History. The first firewall solution provided by Microsoft, Internet Connection Firewall, was first shipped with Windows XP Service Pack 1 in 2001 and was, by default, turned off, because of compatibility issues with websites. As a result, it was rarely used.

The Blaster Worm attack in 2003 compromised at least 100,000 Windows machines[35]. Another attack of similar proportions by the Sasser worm followed in 2004[36]. By late 2004, an unpatched Windows machine would become infected, on average, in 40 minutes after it has connected to the internet. [37].
Microsoft delivered the significantly improved Windows Firewall, which was by default turned on, with Windows XP Service Pack 2 in August 2004.

Figure 12: Hex dump of the Blaster worm.

In later versions of Windows, both client and server, the Firewall is turned on by default, with unrequested inbound network traffic blocked, and all outbound traffic allowed. The Firewall supports rules which allow specific inbound connections (preventing intrusions), as well as rules which can control outbound network traffic by blocking unwanted packets from being sent to the network (preventing leaks). Alternatively, the entire outbound traffic can be blocked, and rules can be added to permit only specific outbound connections.
Windows Firewall with Advanced Security also supports allowing or blocking network packets based on whether they use IPsec authentication or encryption.

Architecture. The Windows Firewall relies on a set of networking components and services. Whilst the actual policies operate at user level, the traffic filtering itself is performed at kernel level by the network address translation driver. This architecture has proven efficient throughout the years. To our knowledge, a similar architecture runs on Windows systems until Windows 8 inclusively, with some additional security features for enterprise and deeper packet inspection introduced in Windows 10.
Windows Firewall provides stateful filtering of TCP traffic, pseudo-stateful filtering of UDP traffic, and stateless filtering of ICMP traffic (thus only permitting to block or allow all echo requests). Due to the low-level implementation, the Firewall is tied directly to the TCP/IP architecture of Windows, and performs simple and effective filtering. Windows Firewall does not use Application Layer information unless it is FTP (see Section 2.2.2). Windows Firewall uses the Application Layer Gateway Service to provide dynamic port mapping for the FTP data channel in order to perform stateful filtering of FTP traffic and avoid the problem of dropping legitimate packets.
The main components, drivers and services that implement the Windows Firewall are can be seen in Figure 14 and are presented below.

- The Windows Firewall/Internet Connection Sharing service runs inside `Svchost.exe`. The service processes the Firewall exceptions and relays with the NAT driver, which inspects the traffic.

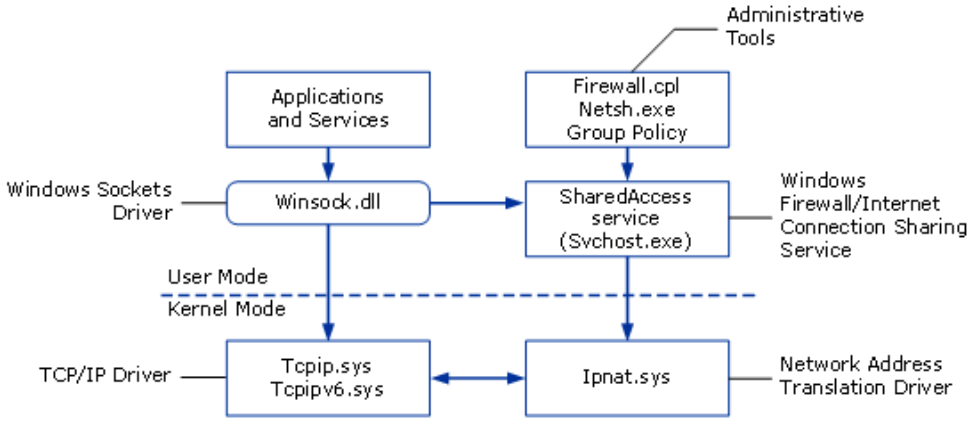


Figure 13: The architecture the Windows firewall (courtesy of Microsoft TechNet). The SharedAccess service running in the Service Host powers the logic of the Firewall, and can be configured by the user via various administrative tools. To enforce its security policy, the Firewall service interfaces with the NAT driver to filter traffic coming through the TCP driver. Allowed connections are forwarded via the sockets driver to the required ports.

- `Ipnat.sys`: The **Network Address Translation driver** provides the NAT Mapping Table which stores the connection state information, including the protocol and source and destination endpoints, with their IP addresses and ports. Even though the Firewall uses the NAT driver, it does not offer network translation features. The NAT driver performs the packet filtering, by inspecting each connection and filtering out the ones which are not included in the exceptions list. Allowed packets are forwarded by the NAT driver to the TCP/IP driver, blocked packets are dropped without notifying the sender (silent discard).
- `Tcpip.sys`: The **IPv4-based TCP/IP driver** controls the flow of TCP traffic between the network adapter and a running application. It defines the TCP/IP protocol allowing a host to lease an IPv4 address from the DHCP server providing internet to the host.
- `Tcpipv6.sys`: The **IPv6-based TCP/IP driver** performs the same tasks as the TCP/IP v4 version, but for IPv6 addresses.
- `Winsock.dll`: The **Windows Sockets Driver** assigns and binds ports to each running program or service who need to listen for incoming traffic.

Security. The Windows Firewall implementation provides a strong bastion for protecting a Windows machine. Consider the recently highly-mediated WannaCry ransomware attack of May 2017. Whilst it indeed exploits vulnerabilities in the Windows system, it has been overlooked that the attacked can be blocked with the Windows Firewall, by simply blocking the port 445 [38]. It is thus of high importance for users to be aware of their Firewall policies and block all open ports which are not frequently used by popular applications.
Only very recently, a strong and sophisticated attack that seems to bypass the Windows Firewall has been detected by Microsoft [39]. This attack exploits the Intel Network controllers, specifically the Intel Active Management Technology (AMT) Serial-over-LAN

(SOL) channel for communication. This channel is independent to the operating system and thus invisible to the firewall or any network monitors on the host. By attacking this channel, the adversaries were able to perform hot-pactching of various system processes. The entry-point of most of these attacks has been spear-phishing. This is a warning of the limitations of software-based firewall technology and a red flag to the human component of every security system.

Interfaces. There are two interfaces related to Windows Firewall: the firewall-hook interface and the Windows Firewall application programming interfaces (APIs). The following figure shows these interfaces and how they relate to Windows Firewall components.

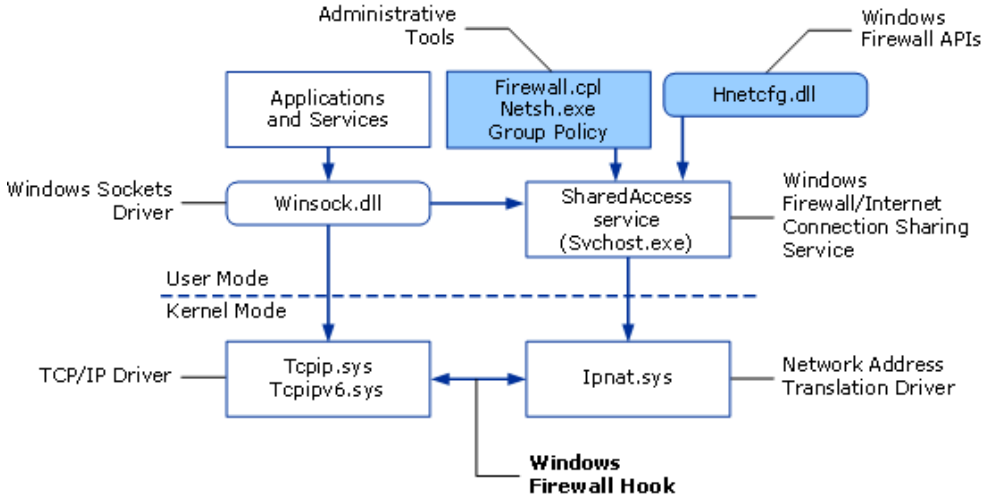


Figure 14: The interfaces related to the Windows firewall (courtesy of Microsoft TechNet).

The interface relevant to this work is the public Windows Firewall API, implemented in `Hnetcfg.dll`. The Windows Firewall API allows direct control to the Firewall Service in `svchost.exe` given administrative permissions. The API allows to enable/disable the Firewall, adding and removing programs, services, ports, and IP hosts to the exceptions list, and configuring logging and ICMP messaging from remote applications.

2.3 IP Geolocation

The word geolocation denotes the latitude and longitude coordinates of a particular location [43]. As the client-independent geolocation of an IP host is crucial for the investigation undertaken in this project, a thorough survey of IP geolocation frameworks has been performed.

Finding the location of an IP host only based on the IP address poses many challenges, since there is no direct mapping between IP address and geographical location. Many of the existing geolocation services rely on commercial databases (e.g. Maxmind[7], IP2Location[8]). Not only are the methods they use unknown, but in many practical situations these datasets are too inaccurate to provide reliable geolocation services. For example a farm in Kansas whose geographic position has been accidentally set as the default site of 600 million IP addresses in the Maxmind IP Geolocation database.[42].

There are several techniques for performing geolocation of an Internet host. Various methods rely on associating a geographic location with the IP address, MAC address, RFID, Wi-Fi positioning system, device fingerprint, GPS. Client-dependent geolocation methods such as GPS-based, Wi-Fi-based and cell tower-based (e.g. Google Maps[9], Skyhook[10], W3C Geolocation API[11]) are usually related to mobile devices, require the user's permission to share their location and generally can have high privacy implications. These types of geolocation have a high overhead when deploying their back-end infrastructure, for example by surveying every street to scan for Wi-Fi access points and cell towers.

For the purpose of this project, because we are looking to geolocate the remote end points of TCP connections formed by the user's machine, we require lightweight methods for client-independent IP geolocation of hosts which are bound to the Internet via a wired network.

The ongoing research field of IP geolocation has produced multiple frameworks of variate efficiency and accuracy. The rest of this subsection will introduce some definitions and a classification of geolocation methods, the proceed to present some of the most important research work on geolocation, discuss the techniques presented in such work, and relate them to the work in this thesis. Figure 15 gives a timeline and graphical overview of how the most popular methods have developed and relate to each other.

2.3.1 Definitions

The research community generally classifies geolocation algorithms into data-driven, measurement-based, and more recently, statistical.

[7] `https://dev.maxmind.com/geoip/geoip2/geolite2`
[8] `http://www.ip2location.com/`
[9] `https://maps.google.com`
[10] `https://skyhookwireless.com`
[11] `https://dev.w3.org/geo/api/spec-source.html`

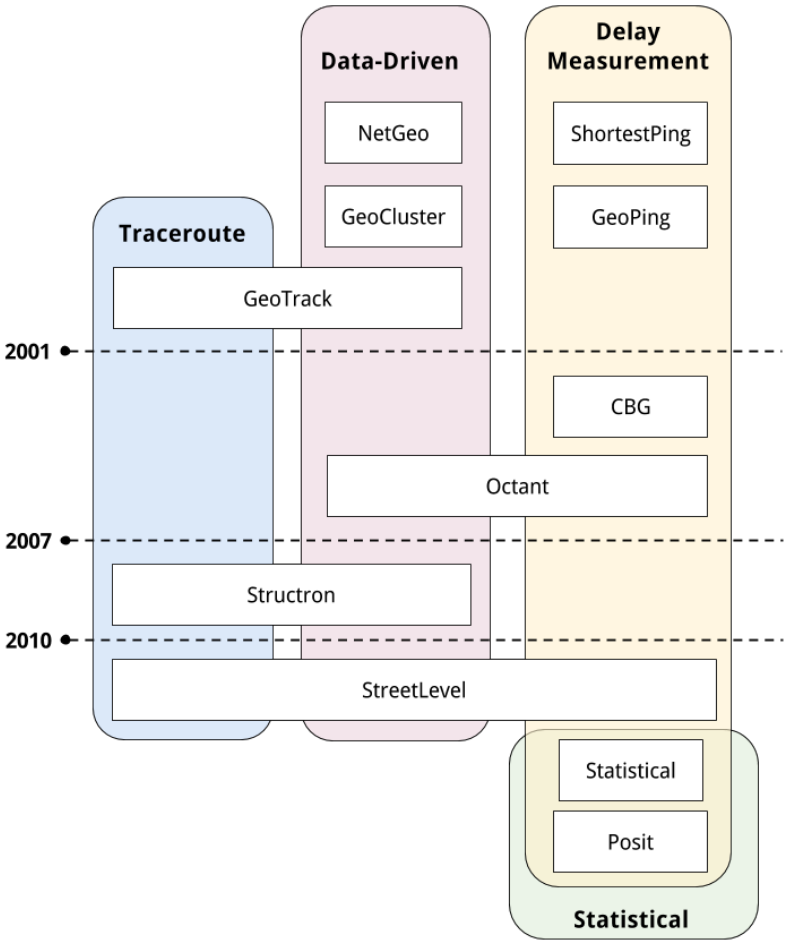


Figure 15: A historical survey of popular geolocation frameworks and methods they employ.

Data-driven geolocation uses protocols such as DNS and WHOIS or online available information such as public databases to infer information about the location of a target host. For example, a domain name may contain naming conventions which indicate the country or city a host belongs to. Similarly, the registrant information in WHOIS records might offer location information.

Measurement-based geolocation uses IMCP probes to identify the latency between nodes in the network. Depending on whether intermediary nodes are targeted, there are two types of such methods:

- **delay-based**, when they use only `ping` measurements to establish latency and approximate distance between two hosts
- **topological**, when they make use of `traceroute` to measure latency and estimate distances between intermediary nodes in the network, or for trying to estimate the way nodes are linked in the network and deduce geolocation information from information found at the nearest known host

Throughout the rest of this thesis, the following concepts are used with respect to measurement-based geolocation:

- The term **target** usually refers to the host of unknown location that needs to be geolocated.
- The term **landmark** signifies an Internet host with precisely known geolocation information. A landmark can either send probes (referred to as an *active landmark* [44], probing landmark, ping server, or a *monitor* [56]), or can be passive, only receiving probes rather than sending them.

Statistical geolocation uses methods of estimation or machine learning to approximate the location of a target, given datasets of geolocation for known hosts.

2.3.2 Data-Driven IP Geolocation

NetGeo [48] mines the location information recorded in the WHOIS database to infer the geographic location of a host. Whilst it may provide successful results, there is no guarantee that the information in the WHOIS database is not stale or inaccurate. Secondly, multiple machines under the same domain name but dispersed in different locations could be all registered as a single entity located at the same address. For example, a WHOIS query would locate all of Google or Facebook's servers from around the world at their headquarters in California.

GeoTrack [44] uses reverse DNS lookups to obtain the names of routers and hosts, and parses the name to infer its location, using the fact it is a popular administrative decision to assign geographical names to router interfaces. This allows inferring location with different levels of granularity. GeoTrack would geolocate the router named `corerouter1.SanFrancisco.cw.net` at the city level, `www.state.ca.us` at state level, `www.un.cm` at the country level. The parsing is done using a database of location codes and naming conventions used by ISPs when naming routers in various cities in Europe and the US. In order to then geolocate an unknown target, GeoTrack uses `traceroute` to get the names and by extension the locations of the routers accessed in the path towards the target, usually choosing the location of the last accessed router as that of the target. This makes GeoTrack both a data-driven and a measurement-based geolocation method.

While this heuristic performs reasonably in practice, there are cases in which it falls short: when multiple domain names are administered by the same administrative domain, the domain boundaries cannot be determined with complete accuracy. Moreover, not all hostnames adhere to this naming convention [45], and furthermore some hostnames are misnamed (e.g. by suggesting a location they are not found at). In [46], by examining data from a large ISP, the authors demonstrate that a while there are not many such errors in practice, these mis-namings induce a large numbers of false links, causing path inflation and routing problems, and making the process of geolocation inaccurate.

A proposal has been made in [47] to include a new Resource Record (RR) in DNS records which would always contain accurate location information. This approach faces deployment issues as it requires changing the DNS record structure and furthermore large amounts of data entry by administrators.

GeoCluster [44] uses various data sources containing network routing information and location information to build a location map for a large subset of the IP address space.

GeoCluster proposes breaking down the IP address space into clusters such that all hosts with IP addresses within a cluster are likely to be geolocated to places which also form a geographic cluster. Then, knowledge about the location of a few hosts in a cluster allows inferring the location on the entire cluster. This framework only makes use of available geolocation data on the web and knowledge of Internet infrastructure to construct a location map for all IP addresses.

For example, suppose that 128.127.126.0/243[12] forms a geographic cluster, and that the prior geolocation data specifies that ten different IP addresses in this cluster are located in Seattle and one is located in Boston. GeoCluster will deduce that the Boston data point is erroneous and that all the IP addresses in the cluster are likely to correspond to hosts located around Seattle.

The clustering algorithm employed by GeoCluster relies on the fact that address allocation and routing in the Internet are hierarchical: routing information is aggregated across hosts which are aggregated under administrative domains (also known as *autonomous systems*). For example, the routes for hosts on a university campus would typically appear as a single aggregate, say as the address prefix 128.127.0.0/16, rather the individual IP addresses. Based on the work in [49], the clustering algorithm uses knowledge of the address prefixes used by the routing protocol to identify topological clusters. The mapping to geographical clusters then appears from combining the knowledge on topological clusters with prior geolocation knowledge.

A similar idea of clustering based on Internet topology is also used with good results by Structon [50], which relies on clustering DNS names.

GeoCluster suffers from many sources of data inconsistency, which make it incapable of achieving fine-grained or reliable geolocation. First, the system relies on the correctness of user-provided data which is difficult to verify. Moreover, geographic accuracy is lost when information mapping an IP to a ZIP code is reduced to a mapping to geographical coordinates by taking the ZIP-center of an area (calculated by various averaging measures) rather than considering the entire area itself. The drawbacks of this kind of approach are discussed at large in [51], where Constraint-Based Geolocation is introduced to allow IP addresses to be geolocated to areas rather than specific points.

Street-Level [58] compensates for the disadvantages occurring when data is collected from the users or from websites directly by using data from location mapping services. These services, most of the proprietary, are usually the result of large content distribution networks

[12]The notation *a.b.c.d/n* denotes an address slice with a prefix (or subnet mask) of n bits

collaborating with multiple ISPs, which in some cases requires an extensive tabulation of IP address ranges and corresponding location. The authors have mined publicly available mapping services for addresses of locally hosted websites (and have developed means of filtering out uninformative cases such as shared hosting or CDN servers), thus having created a dataset of domain name, IP address, and postal address mappings, which they are further used as landmarks in their delay measurement-based methods.

2.3.3 Delay-based IP Geolocation

The goal of delay measurement-driven geolocation is to estimate the location of the target host from an end-to-end delay measurement from landmarks to targets, by exploiting the relationship between network delay and geographic distance. Most such methods rely on a ping server with known geolocation to send probes to the target, and then various techniques are used to process the packet delay to infer geographical location.

Shortest Ping [44] is the simplest method to perform delay-based geolocation. Each target is mapped to the landmark that is 'nearest' with respect to the measured RTT from the probing landmark to the target and back.

GeoPing [44], rather than focusing on the shortest delay from the ping server to the target, probes a target from all ping servers in order to build a *delay vector*. Furthermore, it makes use not only of the *active* landmarks (i.e. ping servers with known geolocation), but also of a database of *passive* landmarks (hosts with known location but from which it is not possible to send probes). Specifically, these passive landmarks are probed from all ping servers to construct their delay vectors, and then stored with their location as well as their delay vectors in a *delay map*. The delay map constitutes a conceptual *training dataset* and is constructed prior to geolocation.

For example, consider a set of N active landmarks (ping servers) and a host, either a target or a passive landmark. The passive landmark will have a corresponding delay vector

$$DV = (d_1, d_2, ..., d_N) \tag{1}$$

which contains the smallest measured round-trip-time (RTT) to the host from each ping server to the target and back. This delay vector acts as a descriptor of each landmark. The GeoPing heuristic then searches for the 'nearest' landmark in the delay map and selects its location as the location of the target. The measure of distance introduced in the paper is the Nearest Neighbour in Delay Space (NNDS), defined as the minimal Euclidean distance between delay vectors. Thus, for two delay vectors DV and DV', the distance between them is defined as

$$\sqrt{(d_1 - d'_1)^2 + ... + (d_N - d'_N)^2} \tag{2}$$

The robustness of NNDS is given by an increasing number of active landmarks and the accuracy of the results depends on abundance of passive landmarks. Figure 33 shows the error versus the number of probes based on the authors' experiments. Another contribution to the robustness of the method is the fact that delay is measured multiple times and only the minimal delay is used in the delay vector.

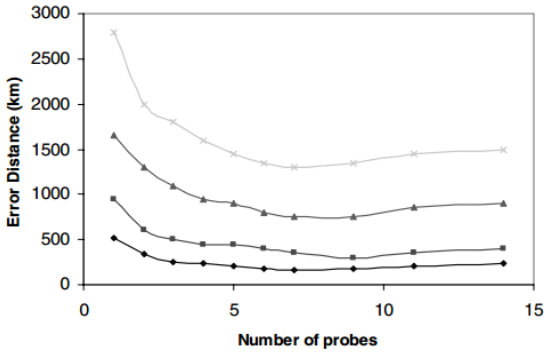


Figure 16: Geolocation error vs number of active landmarks.

The advantages in using passive landmarks consist in allowing the technique to perform better as the dataset without having to increase the density of probing landmarks, which is not always possible in practice. But this method's accuracy is strongly dependent on the proximity of landmarks around the target.

Further work on delay measurements considers a factor of conversion between delay and geographical distance. Research on Internet performance [54] has shown that packets travel through fiber optic cables at $m = 2/3$ speed of light in vacuum. However, this is more like a loose upper bound, as in practice we need to account for queuing delay, transmission delay, packet delay, and other non-propagation delay which may occur. Experiments in [51] suggest the correlation between geographic distance and network delay (see Figure 17). The paper also refutes the concept of a *baseline* when the measurements are only subject to the propagation delay of the medium, and proposes a statistical, data-driven method instead to determine a *bestline* which approximates the converting factor between delay and distance. This solution is then employed to implement Constraint-Based Geolocation.

It is obvious that such techniques can have inaccuracies, which are directly influenced by various factors, hardware-based or topological. A more accurate estimate of $m = 4/9$ as the converting factor between delay and geographical distance has been proposed in [53], yet the authors stress that pure delay-methods are generally prone to inaccuracy. Note that the measure for geographical distance used in most frameworks is the great-circle distance [55], the shortest distance between two points on the surface of the earth taking into account the Earth's curvature, which can itself introduce measurement errors.

Constraint-based Geolocation (CBG). Geolocation methods previous to **CBG** [51] use active landmarks to provide the location estimation of the target host, but in the form of a discrete set of answers. In contrast, CBG uses multilateration to establish, based on the distance constraints induced by the landmark estimates, a continuous space or a region where the target must be located.

Multilateration is the process of estimating the physical position of a given point using a sufficient number of distance measurements to some fixed points whose positions are known.

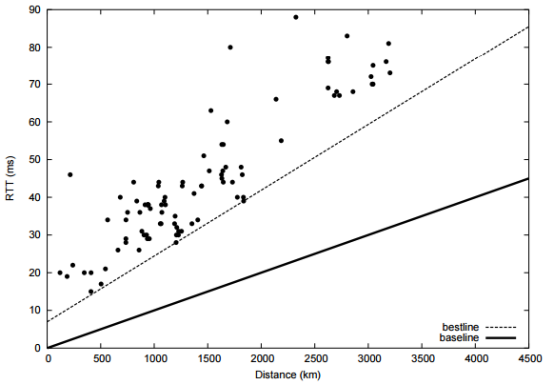


Figure 17: A scatter plot of geographic distance and network delay from the original CBG work. The solid line represents the baseline (estimated best-case) and the dotted one the bestline (the results inferred from data by the paper).

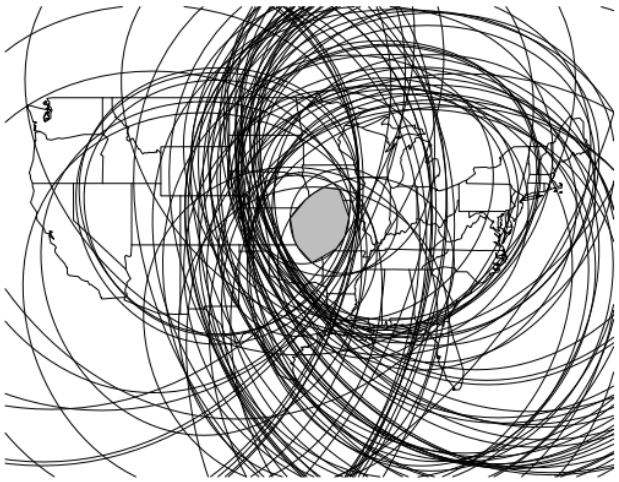


Figure 18: An example of multilateration from the original CBG work. For each ping server, we draw a circle centered at its location with the radius estimated by the delay measurement, and we take their intersection as the area where the target must be located.

For example, the Global Positioning System (GPS) uses multilateration to satellites to estimate the position of a GPS receiver [52]. In particular, CBG uses multilateration to build an intersection that covers the target as follows: for each ping server, we draw a ring centered at its location with the radius estimated by the delay measurement. The intersection of these circles will always find a region that covers the target IP. Figure 18 shows an example of this method.

To estimate the distance from a ping server with known location to the target, CBG first sends probes to the target from the ping servers, measuring delay from all the ping servers to the target. This delay is then converted into a geographical distance using their *bestline* delay-to-distance approximation.

Currently, CBG is the state-of-the-art in geolocation based only on delay measurements. The main disadvantage of pure delay-based algorithms is the error induced by the distance to the nearest vantage point or landmark. The work in [53] shows that generally such methods provide worst-case errors of over 1000km when the target is away from any landmarks, and while CBG *bestline* constraints attempt to compensate for path inflation or circular paths in Internet topology, a single conversion factor for the entire network is not enough to capture the intricacies of network topology and routing policies.

2.3.4 Topological geolocation

As far as topology is concerned, nodes in the internet layer plane are *autonomous systems*, and links between nodes can suggest relationships between the Internet Service Providers

(ISPs) or other organizations in charge. The topology of the Internet layer can be used to infer information about geolocation when making `traceroute` measurements. A possible way of gathering this information is from IP mapping tables belonging to Internet Service Providers (ISPs).

Classical approaches on `traceroute`, such as GeoTrack [44], Structon [50] and Street-Level [58] make use of `traceroute` to identify the routers on the path from the active landmark to the target, and then select the location of the last router in the path as the location of the target.

Octant. Similar methods have been employed in Octant [57]. Building on top of CBG, Octant uses geographic information from the DNS name of routers along the path to the target, as well as geographic constraints, for example, it considers feasible and unfeasible regions where a router could be located by making use of population density and geographical data. Octant proposes a framework which can be trained with this additional data in order to perform more accurately at geolocation.

2.3.5 Statistical geolocation

Another way of improving delay based measurements is to make use of statistical embedding and modelling to compensate for the likely errors inherent to sending probes over the Internet. Recent work in IP geolocation [61], [62], [60] has been focusing on maximising the likelihood probability that an IP address can be geolocated at a specific position, after having learned probability distributions from training sets of geolocated hosts. All three frameworks assume conditional independence between measurements and perform maximum likelihood estimation to calculate the geographic location.

Learning-based Geolocation [60] is a framework which proposes that IP geolocation be expressed as a machine learning classification problem. Using only ping-based measurements from a set of known monitors to the target, the location of that target is classified based on these measurements such that it lies in the most probable geographic region. Unlike other frameworks, which have been tested on small data sets, Learning-Based has been tested for many data points and shows very promising results.

The paper introduces a Naive Bayes estimation method that assigns a given IP address to a geographic region out of multiple partitions, based on a set of measurements associated to that target. The parameters taken into account when classifying the IP address to a location consist of latency and hop counts from multiple active landmarks to the target, as well as population density data. The classifier has been designed with the geography of the US in mind, and the geographic regions it classifies to are all the counties in the Unites States.

Consider a measurement $\mathcal{M} = \{m_1, m_2, ...m_N\}$ which represents both latency and hop count values to the target from the set of monitors, and a set of counties $\mathcal{C}$ such that the

target with measurement $\mathcal{M}$ is located in some county $c \in \mathcal{C}$. Then, to estimate the county c^* in which the target is located, one should maximize the probability of the target being located in county c given measurement $\mathcal{M}$. Then, using Bayes theorem:

$$c^* = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\, P(c|\mathcal{M}) = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\, \frac{P(\mathcal{M}|c)P(c)}{P(\mathcal{M})} = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\, P(\mathcal{M}|c)P(c) \tag{3}$$

where the evidence is discarded due to being constant across counties.
The authors express $P(c)$ as a measure of population density

$$P(c_i) = \frac{pop(c_i)}{\sum_{c_j \in \mathcal{C}} pop(c_j)} \tag{4}$$

and decompose the measurement $\mathcal{M}$ as a product of independent measurements

$$P(\mathcal{M}|c) = P(\{m_1, ...m_N\}|c) = \prod_{m_i \in \mathcal{M}} P(m_i|c) \tag{5}$$

To estimate the one-dimensional densities $P(m_i|c)$ the framework uses distance vectors $d = \{d_1, ...d_m\}$ such that d_i is the distance between the monitor i and county c, to learn the probability of observing measurement m_i given that the target is located d_i away from monitor i.

Furthermore, weights are integrated into this estimation methods as the authors argue that a latency-based measurement is likely to be more accurate and of more use than hop-counts or population data. Thus the parameters λ_{hop}, λ_{pop}, γ_{hop} and γ_{pop} are used as regularizers for the two quantities, and the training phase is dedicated to estimating these parameters. Figure 19 visually describes the steps taken by the Learning-Based algorithm.

The authors propose this model as a proof of concept rather than a definitive solution, so it is worth noting that machine learning approaches are the newest point of view on geolocation. While as they settle for a coarse-grained (county-level) geolocation in this paper, the framework is feasible to be extended to finer-grained areas, such as post-codes or blocks. Further work [56], [61] proposes different models which are discussed below.

Posit [56] is a state-of-the-art geolocation framework which proposes an adaptive behaviour in order to geolocate machines which refuse probes. The Posit framework is divided into two components:

1. Hop-based geolocation, inspired from the methods in GeoPing, which uses network topology structure and hop count vectors to a set of passive landmarks to infer geolocation. Note that hop-based Posit is effective in geolocating adversarial targets which generally refuse probes
2. Latency-based geolocation, which uses delay measurements from the monitors to both targets and landmarks in a CBG fashion, and performs maximum likelihood estimation to convert the delay into a distance measure

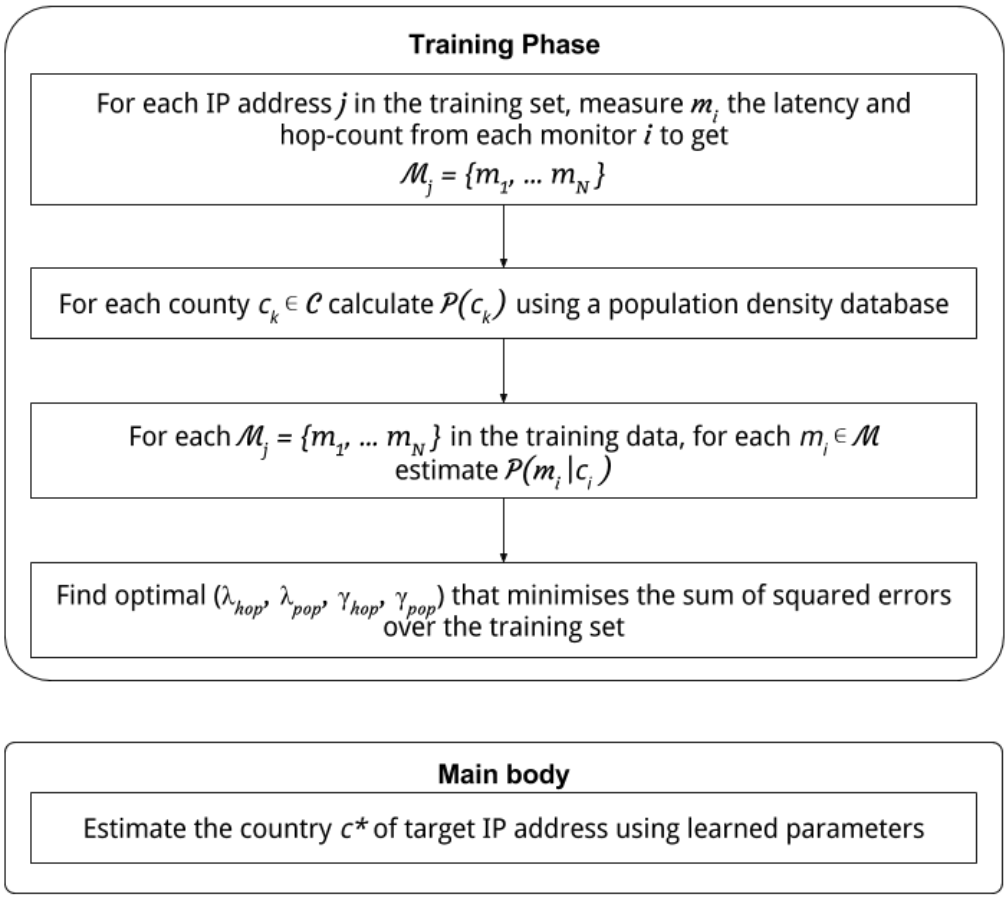


Figure 19: Learning-based Naive Bayes Geolocation Algorithm

Hop-based Posit is remarkable in its design due to its advantages against adversarial targets. The framework constructs hop-count vectors for each target $i = \{1, 2, ...N\}$ and for each landmark $j = \{1, 2, ...T\}$ by measuring the hop count from each of the M monitors.

$$h_i^{target} = \left[h_{i,1}^{target},\ h_{i,2}^{target}, ...,\ h_{i,M}^{target}\right] \tag{6}$$

$$h_i^{land} = \left[h_{j,1}^{land},\ h_{j,2}^{land}, ...,\ h_{j,M}^{land}\right] \tag{7}$$

Using the hypothesis that topologically close targets are also geographically close, each target is geolocated to the location of the topologically closest landmark. The closes landmark is defined as the landmark with the smallest variance in the hop count vector difference.

$$c^* = \underset{j}{\operatorname{argmin}}\, \sigma^2(h_i^{target} - h_j^{land}) \tag{8}$$

The drawback of this method is that when no landmarks are topologically close to the target, then there are very inaccurate results. Thankfully, the variance measure can suggest

The work in [63] shows that hop-count vectors contain enough information to deduce network topology such as to cluster targets in a significant way. In network layer topology, an *egress router* is a router which monitors and potentially restricts network flow from a network to another.
Thus, if all monitors are in the same network, then the paths towards two targets which share the egress router will be shared past the egress point. Given this shared path, the authors have determined that two targets i and j are topologically close in the network if they have the hop count property:

$$h_{i,k} = h_{j,k} + C$$

for every monitor k located past the shared router, and some constant C representing the number of common hops.

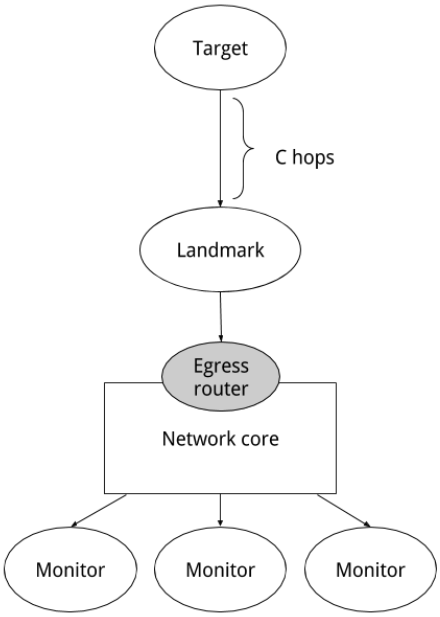


Figure 20: A network where the target is C hops away from a landmark, with both sharing the same path.

whether this is the case. On the other hand, this framework can be used for target which were, until the work in this paper, impossible to geolocate. Specifically, hop counts can still be derived from hosts which refuse probes by monitoring packets and collecting time-to-live (TTL) counts at each monitor, which can be used to infer the number of hops. [64]

In its latency component, Posit improves the work in Learning-Based by introducing likelihood distributions to passive landmarks as well as monitors.

2.3.6 Evaluation

By studying previous work, one can observe two trends in recent IP geolocation: first, a growing concern with accuracy in the detriment of scalability, displayed by frameworks like Street-Level or Octant, and secondly, an interest in large-scale IP geolocation, such as the idea of a global geolocation map proposed by GeoCluster, or the initiative to geolocate all IPs in China proposed by Structon.

While the evaluation of an IP geolocation framework sounds like a very concrete task, in practice there are many issues that need to be considered. Specifically, one has to evaluate the precision of the framework, by using a dataset of IP addresses with known location and comparing the test results against the real data. The problem is that the properties of this dataset affect the conclusions that can be drawn from the evaluation phase.

Recent work [65] attempts to evaluate the success of simple frameworks such as Shortest Ping and CBG on large-scale datasets and argue that this compensates for the greater lack

of accuracy of these methods. But this work only achieves its goal because it focuses on scalability rather than accuracy. On the other hand, most popular recent work does not test their framework against large datasets. Some effort has been put in evaluating the performance of frameworks on datasets consisting of different types of hosts (residential, university, commercial). Furthermore, work in [56] and [58] runs geolocation experiments to observe the correlation between the density of monitors and landmarks versus the accuracy of the results, and [58] also researches correlation with population density.

The drawback in the work above comes from having tested the frameworks on less than a thousand data points. From a statistical perspective, this is an extremely weak result compared to millions of IPs that might need to be geolocated in a real-world application. By surveying the literature, one can observe the following phenomenon: when researchers attempt to test a competing framework on a different dataset than the one originally used in the framework's evaluation, great discrepancies occur. For example, the original Street-Level [58] framework claims to achieve less than 1km accuracy, but the authors of Posit [56] refute this claim by achieving way less encouraging results on different datasets.

The tables below presents a survey of geolocation accuracy aggregated from the experiments in multiple papers. From these tables we can see the conflict in performance depending on the data set.

Method	Mean error (km)	Median error (km)
Shortest Ping	393.13	352.124
GeoPing	332.941	274.602
CBG	328	299.305
Octant	260.037	153.434
Street-Level	416.192	352.124
Learning-Based	271.528	194.939
Posit	187.585	74.2551

Table 3: Statistics of measurement-based techniques collected from [56] on 431 commercial hosts with known coordinates belonging to Akamai Technologies in North America.

Method	Median error (km)
GeoPing	68
CBG	89
TBG	68
Octant	22
Street-Level	0.69

Table 4: Statistics of measurement-based techniques collected from [58] evaluated on PlanetLab nodes.

2.4 Privacy

It is no surprise that many technology providers have succumbed to data collection such as location information, Internet activity and browsing history to offer users a personalised experience. But removing the user the choice and control over the forms of data collection is what has become problematic. More and more software providers have started to obfuscate privacy settings from users, or force them to accept permissions needed to use various services, without giving them the choice to opt-out of the service and keep data collection off. Microsoft's Windows is not an exception. But Windows is not a phone application or a website that we can choose not to use - it is an operating system used by the majority of people in the world. This section outlines the main known privacy issues on Windows systems and discusses various research work on these issues.

2.4.1 Windows telemetry

The Customer Experience Improvement Program (CEIP) allows Windows users to send over their telemetry data to improve Microsoft services. Users enrolled in this programme will automatically enable the Telemetry / Feedback & Diagnostics Windows services to collect user data.

From Microsoft's privacy policy: "Finally, we will access, disclose and preserve personal data, including your content (such as the content of your emails, other private communications or files in private folders), when we have a good faith belief that doing so is necessary."

In August 2015, Microsoft delivered the first updates to Windows 7 and Windows 8 users (KB3075249, KB3080149 and KB3068708) that would start sending telemetry data to Microsoft[12]. While the first two are optional, the third is a *recommended update*, meaning that users who use Automatic Updates would have it installed without asking for their confirmation. Many users were disappointed that the updates did not describe in detail what the privacy implications are, forcing many people to unwarily join a data collection service. The data collection settings can be seen in the Windows Registry[13].

Furthermore, in October 2016 the quality roll-ups KB3192403 and KB3192404 are proposed, both giving even more permissions to the Telemetry services. In the meantime Microsoft has changed their update policy pushing updates as monthly packages rather than individually. The two roll-ups above include both relevant security updates and the upgrades to Telemetry in the same package, removing fine-grained control from the user.

Microsoft lists two host names in each of these updates that data is received from and sent to: `vortex-win.data.microsoft.com` and `settings-win.data.microsoft.com` citeprivacy-win-update-1 [10]. The two and potentially many others are hard-coded to bypass the hosts file in a system DLL[14]. It has been argued that only a manually configurable firewall is able

[13]From the registry subkey `HKEY_LOCAL_MACHINE/SOFTWARE/Policies/Microsoft/Windows/DataCollection`

[14]Found in the Windows install directory, in `/system32/dnsapi.dll`

to stop this data from being sent after users have installed updates. Multiple other hostnames have been suspected to do the same, and there are user-led attempts to identify all of them. [14] [**?**]

Windows Update	Details
KB3015249	Adds telemetry capabilities
KB3022345	Installs diagnostic tracking service
KB3068708	Introduces the Diagnostics and Telemetry tracking service
KB3075249	Adds telemetry points to the User Account Control
KB3080149	Updates the Diagnostics and Telemetry tracking service
KB3192403	Updates Telemetry to upload data by authenticated proxies
KB3192404	Updates Telemetry to upload data by authenticated proxies

Table 5: The Windows updates which affect privacy on Windows 7 and 8. CEIP comes by default with Windows 10.

2.4.2 Windows 7

Customer Experience Improvement Program is the main adversary when discussing the user's privacy on Windows 7. After a user installs the updates mentioned above, it is very difficult to roll-back or configure Telemetry settings. On Windows 7, unlike the in 8, 8.1 or 10, there are no settings screens where people could change their privacy settings. In order to disable CEIP one would have to check individual settings for each application included in the programme. Microsoft states that "most programs make CEIP options available from the Help menu, although for some products, you might need to check settings, options, or preferences menus."[13]

2.4.3 Windows 8.1

On top of the updates discussed above, when we talk about Windows 8 and 8.1 talk about a modern operating system, with support for personalisation and mobile. Thus unlike in Windows 7, Windows 8 and 8.1 provide specific control panels for controlling privacy settings such as sharing location, voice and text data with apps.

On the other hand, users do not have any offline options to regulate advertising. Instead, they have to use their Microsoft account to opt out of receiving personalized ads or prevent apps from sending unique device or user identifiers to 3rd parties.

Research work done in [3] has attempted to evaluate the privacy leaks on Windows Phone 8.1. To test whether the Windows Phone device leaks location information when the location setting is turned off, the researches monitor the use of MAC address, IP address and GPS coordinates by installed applications using a man-in-the-middle proxy running packet sniffing software. Each of the 40 applications they have evaluated were tested for two minutes.

Name	Location	Advertising
BBC iPlayer	✓	
Bible		✓
Combo Pic		✓
Copter		✓
Wizards Choice		✓
Falldowm		✓
Hangman		✓
Logo Quiz		✓
Millionaire		✓
PGA Tour	✓	
Poynt	✓	
Robotek	✓	
Chelsea FC News		✓

Table 6: Applications which either leak location data or advertising information.

The table below presents a list of applications which have been found to leak information in [3]. Many of them are games or news applications. No application has been identified to leak both location and advertising information, yet we are aware their dataset was very small.

While we are aware there are differences between the Windows Phone and Desktop architectures, they both run on the same kernel and they provide identical Privacy Setting screens. The issue we believe here is not that there are few unknown applications leaking data, but that this is possible at all. We are inclined to suspect that using devices which allow sharing location e.g. tablets running Windows Desktop may be at risk of leaking location information in the same way as phones, but we were unable to find research to support this claim.

2.4.4 Windows 10

Windows 10 provides a suite of privacy settings but also comes with Telemetry enabled by default. When setting up Windows, it is possible to configure initial privacy settings such as not to share your advertiser information, but users have still complained about privacy on Windows 10. Unfortunately w to the best of our knowledge there is little peer-reviewed literature analyzing which data is being sent to Microsoft, but the online community has been eager to compensate[**?**]. Users have claimed that Microsoft collects telemetry data through Cortana, Explorer, and the Office suite, as well as sharing data with advertisers. Multiple online resources are available e.g. in [**?**] to suggest how to improve your privacy settings on Windows 10.

The Microsoft website lists which applications collect data on Windows 10[16], examples include Bing, Cortana, Microsoft Office, Groove Music and TV, MSN, OneDrive, Outlook, and Skype.

2.5 Related Tools

This section presents some state-of-the-art tools used for network monitoring and firewall management.

2.5.1 SysInternals

The SysInternals[15] suite of tools is provided by a group of developers supported by Microsoft. The tools have been developed since the 9x versions of Windows, and have been continuously improved for newer versions of Windows.

Out of the tools part of this suite, the most comprehensive are Process Explorer, Process Monitor and TCP View.

Process Explorer augments the Windows Task Manager, allowing the user to preview information about the running processes and the attached DLLs, kill these processes and close open handles, and preview how resources are used on the system. But Process Explorer does not provide any logging features, and it does not interfere with the network.

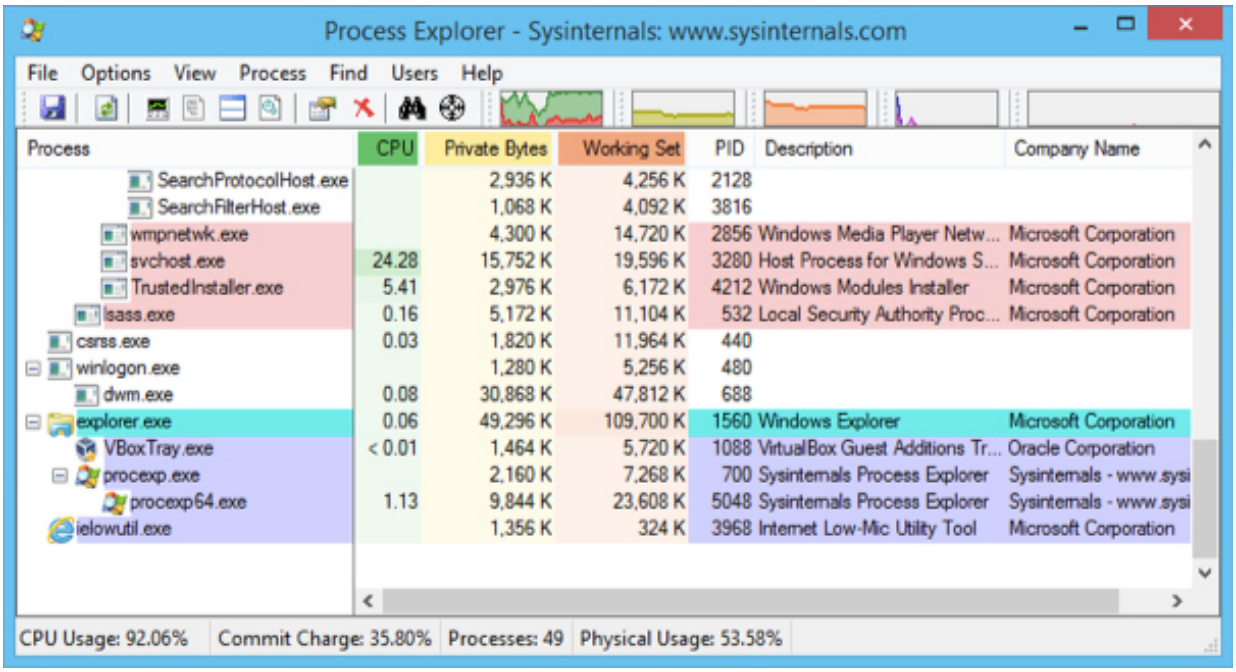


Figure 21: Process Explorer allows the user to interact with running processes.

Process Monitor is an event logger and monitoring tool. It allows real-time preview of the running processes, including extra details about the process such as the path, the owning user, registry activity, file system activity and network activity for a specific process. Process Monitor does not allow user interaction such as killing processes, but it does provide

[15] `https://technet.microsoft.com/en-gb/sysinternals/bb545021.aspx`

logs. Even though it contains large amounts of useful information in one place, the user interface is cluttered and heavy.

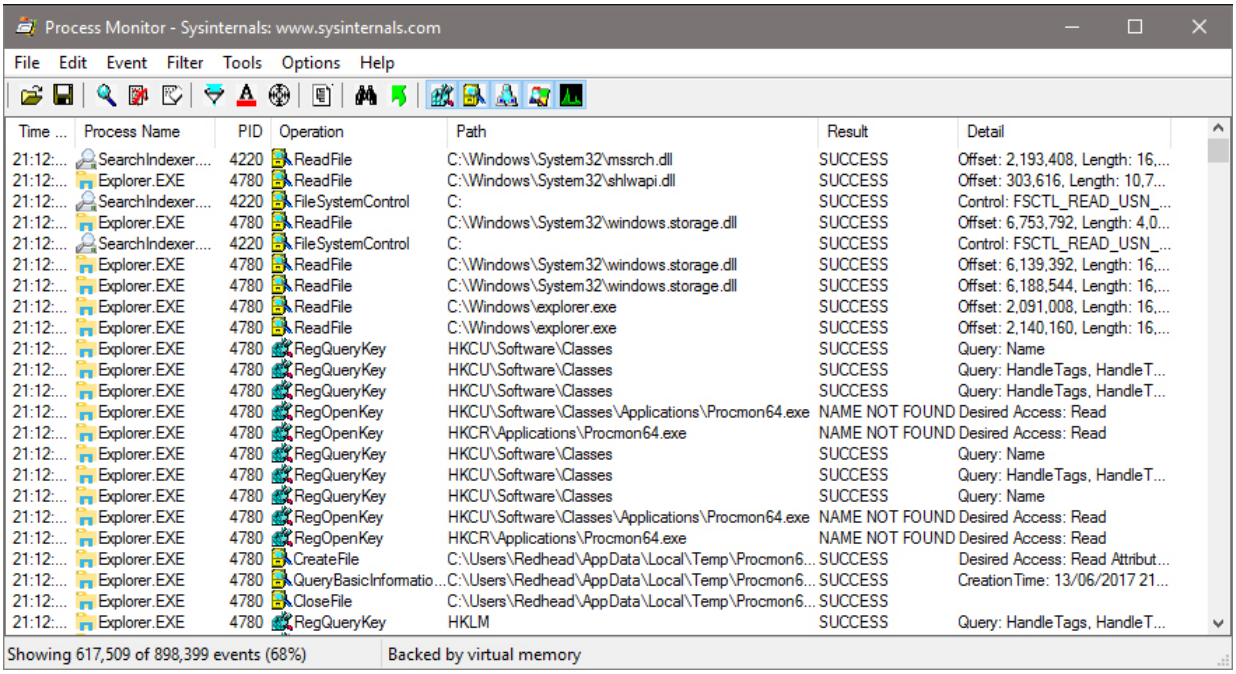


Figure 22: Process Monitor allows real time preview of running process.

TCP View is the closest to our desired platform. It is a networking tool which allows the user to see in real time all the TCP and UDP connections that open through and from the localhost, as well as querying WHOIS information and terminating these connections or the owner processes. The disadvantage is that TCP View neither does log these connections, nor does it interface with the Windows Firewall, so it only provides a temporary solution for the problem of unwanted connections.

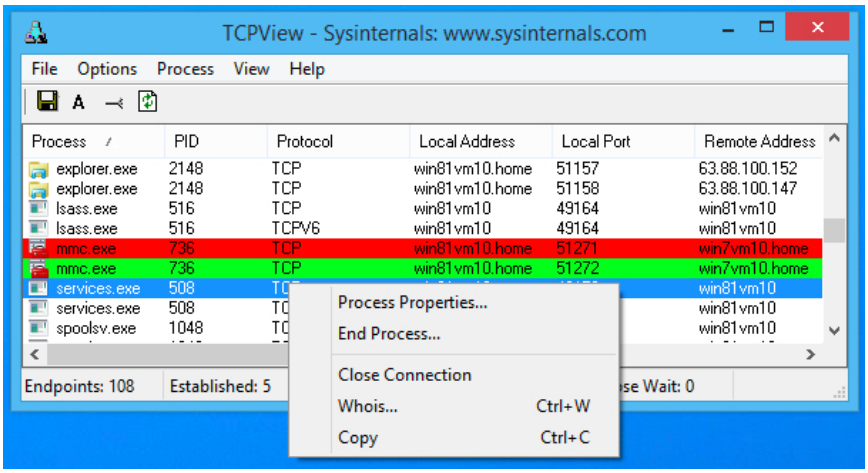


Figure 23: TCP View allows inspecting and closing open TCP or UDP connections.

2.5.2 GlassWire

GlassWire[16] is a commercially available firewall software which augments the Windows firewall by providing visualisations of network activity and bandwidth statistics. Furthermore, it provides a large suite of intrusion detection features, such as detecting changes in system files and device lists.

Figure 24: GlassWire is an augmented firewall which offers graphs for bandwidth statistics.

2.5.3 WFN

The Windows Firewall Notifier[17] allows the user to see, in real-time, whether an outbound connection has been established at the localhost, and allows the user to block it with firewall rules. The user interface also allows viewing a list of connections with basic geolocation features, bandwidth statistics, as well as a map. But WFN does not allow the user to create higher-level rules, it does not run in the background, and their logging feature is not functional.

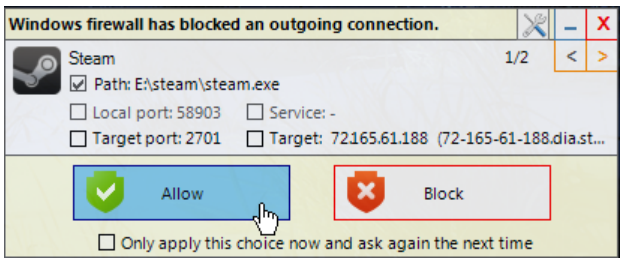


Figure 25: WFN allows the user to block unwanted outbound connections.

[16] https://www.glasswire.com/
[17] https://wfn.codeplex.com/

2.5.4 Windows Privacy Tools

The article at [15] presents a comprehensive comparison and discussion about various privacy tools available for Windows. These tools allow the users to disable Telemetry and Cortana, edit specific registry entries to block the upgrade to Windows 10, disable services and remove applications via either a front-end or scripts. While these tools give the user access to the obfuscated privacy and many under-the-hood settings, most of them are seldom updated, not all have friendly front-ends, and none provide monitoring and logging.

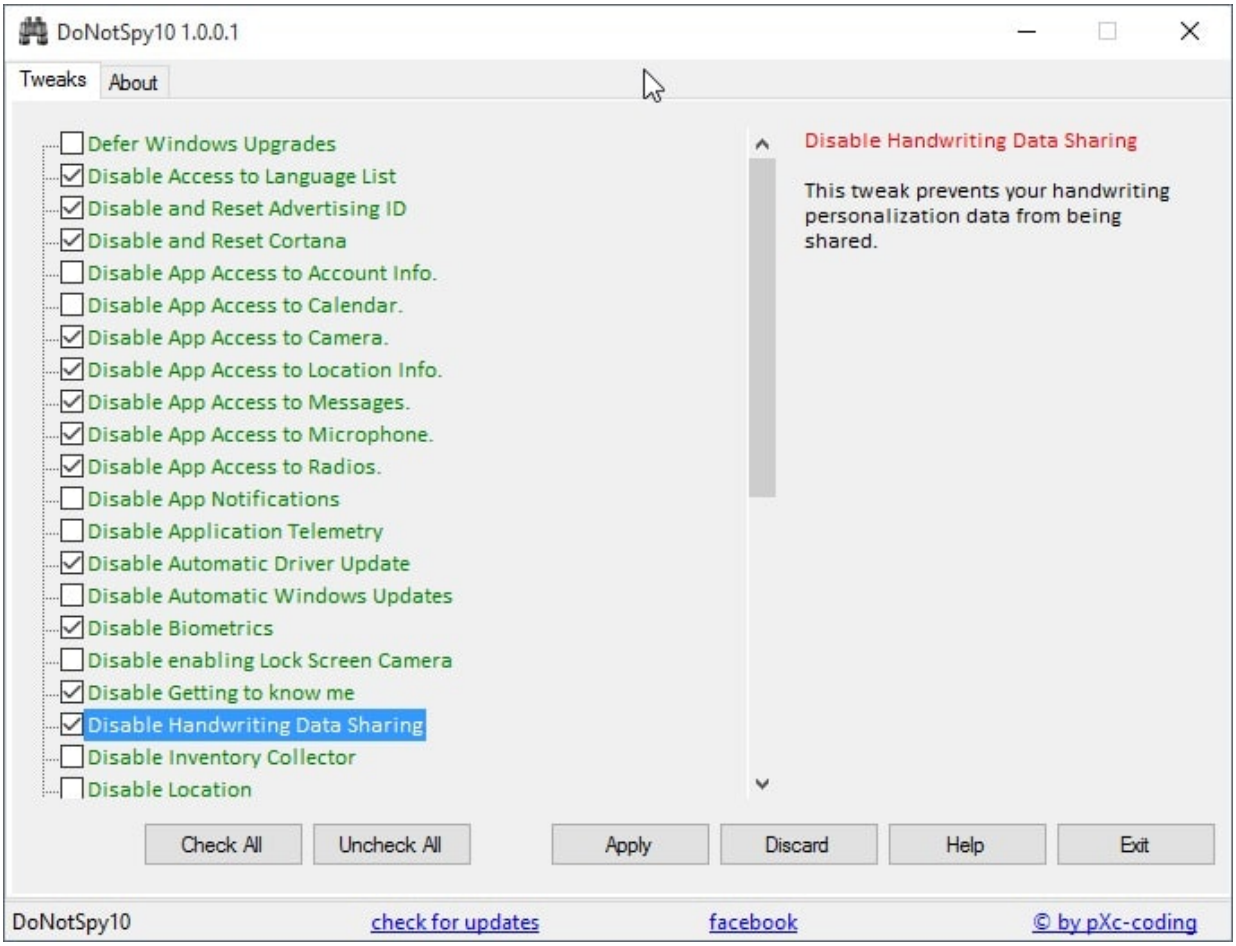


Figure 26: DoNotSpy 10 provides an interface where a user can disable application access to sensors or system features, speech or handwriting data sharing, and Cortana, all in one place. DoNotSpy 10 supports creating restore points before performing these tweaks, but ships with adware in its installer.

2.5.5 Conclusions

From the above survey, we have observed a lack of tools which provide process monitoring, network analysis and firewall control inside the same tool.

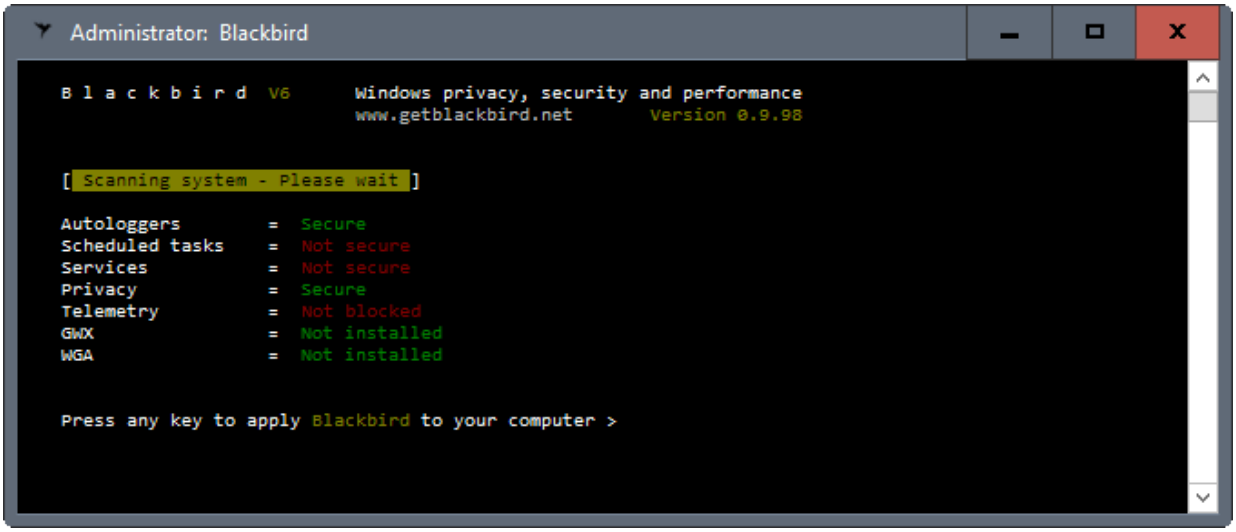


Figure 27: Blackbird is a program for Windows Vista and up which can be run from the command line. It provides a very extensive list of features, allowing tweaks ranging from blocking updates, telemetry and ad servers and disabling services.

The tools which focus on networking seem to lack in details, in contrast with the large amount of information the process monitoring tools provide.

Moreover, not all of these tools provide logging, and the user interfaces are very unfriendly, more directed to the IT professional rather than the average user.

Whilst most of these tools offer, individually, many features supported in SnowWall, the average user is still lacking a user-friendly but powerful tool for blocking intrusive network connections, logging the network activity, and viewing statistics. Furthermore, none of these tools focuses on providing information to the user such as where their data is going or what organization is involved.

3 System Design

This section presents the main design decisions undertaken in the development of this project, from technologies to architecture and user interface design. It concludes with a list of features supported by the application.

3.1 Technologies

3.1.1 Supported Platforms

The choice of technologies for developing SnowWall is dictated by the desire to incorporate as many versions of Windows as possible. It is worth noting that backwards compatibility has been considered more of an issue than forwards-compatibility, as Windows 10 has been announced to be delivered as a service, and thus we expect that software compatible with the current version Creator's Update will maintain compatibility in the future.

SnowWall supports Windows 7, Windows 8, Windows 8.1, and Windows 10 as long as the .NET Framework 4.6 is installed.

3.1.2 Development Framework

The **.NET framework**[18] has been chosen for this project as it is the go-to for developing Windows applications, because it provides support and interfaces for the Windows operating system, and because it allows the flexibility of choosing your preferred programming language.

All .NET applications run in a virtual machine known as the Common Language Runtime. The Framework provides a set of class libraries and features for developing on Windows known as the Framework Class Library. The Framework Class Library contains classes for user interface (such as Windows Forms and Windows Presentation Foundation), data access (such as the Entity Framework), web development (ASP.NET), cryptography, network services, and libraries for asynchronous programming and parallelisation (Parallel LINQ, Task Parallel Library).

The front-end is built with **Windows Forms**[19], the simplest front-end technology provided by Microsoft for development in Windows Desktop. While we are aware that Windows Forms is a dated framework which does not support features such as e.g. animations, it is the only front-end framework that can be built using only back-end code, without the overhead of learning specific technologies and markup-languages. Due to the size and complexity of this project, while the front-end is of great importance, we decided to priori-

[18] `https://www.microsoft.com/net`

[19] https://docs.microsoft.com/en-us/dotnet/framework/winforms/

tise the back-end and functionality, and thus chose the simplest solution that satisfies the front-end design.

3.1.3 Programming language

The programming language chosen for this project is **C#**. C# is a language developed as Microsoft and standardised by ECMA[69] and ISO[70]. It is a multi-paradigm language which supports strong typing, generic object-oriented programming, as well as imperative, declarative, and functional features. Moreover, C# supports asynchronous programming, multi-threading and parallel programming, required by the data processing necessary to identify and geolocate each host.

3.1.4 Database management

With large amounts of information flowing in and out of the database, the necessary asynchronous processing, and a great need for speed, integrity and reliability, a flexible solution for data management is required. We cannot afford to either miss logging connections or to add duplicates, as this will mislead the user and affect any security evaluation that may be done based on these logs.

Therefore, a **SQL** database which supports atomicity, consistency, isolation and durability properties is required. For this purpose, we have chosen **SQLite**[20]. SQLite is free to use and the SQLite Server can be embedded into the installer such that the user does not need to be concerned with any prerequisites during the installation.

The project uses **Entity Framework**[21] to interact with its data sources. Entity Framework is an open-source object relational mapping framework provided by Microsoft. It allows direct mappings between model classes defined in application code and database objects, with a choice of developing code-first, model-first or database-first. It has been chosen due to its flexibility and versatility, because it avoids the difficulties of managing data sources and data connections, and because it provides migration techniques for modifying the structure of a database automatically whenever the code-based model is modified inside the application without losing existing data.

3.2 Architecture

Because of the need to interact with many system components, SnowWall features a layered architecture formed of multiple components. The design is highly modular, allowing for each of the components to be replaced at any time. We have taken this design decision to

[20]SQLite is an open, lightweight SQL server `https://www.sqlite.org/`

[21]Entity Framework is Microsoft's recommended data access technology. `https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx`

allow different geolocation frameworks to be used and because we are aware the front-end can be improved.

The back-end is formed of multiple components and libraries which can access various APIs such as for retrieving a list of active TCP and UDP connections, a list of running processes or details for a specific process and bandwidth statistics. The back-end includes a component which issues web requests to query online geolocation services, DNS and WHOIS records. It also contains functions which perform tasks such as user impersonation and adding registry keys, and interfaces with the Windows Firewall.

The front-end displays multiple views to the user, each tailored to a specific data visualisation. The front-end views are chosen by a principle of separating concerns: one focuses on the geolocation of active connections' end points, one on the owners of these end-points, one on network statistics and processes and one on firewall rules.

The flow of information through the program is designed in such a way as to allow logging of all detected activity into the database. For example, after intercepting a list of network connections active at the current moment in time, the back-end makes asynchronous calls to DNS records and geolocation servers to identify information about connection end points, and the results are stored into the database.

Another example of control flow is when the user creates a blocking rule: the rule is saved into the database, and subsequently, all active connections are filtered by this rule. If a connection is encountered to satisfy the rule, a firewall rule is automatically generated to block it.

A comprehensive description of each component in the SnowWall architecture is provided in the implementation section 4.1.

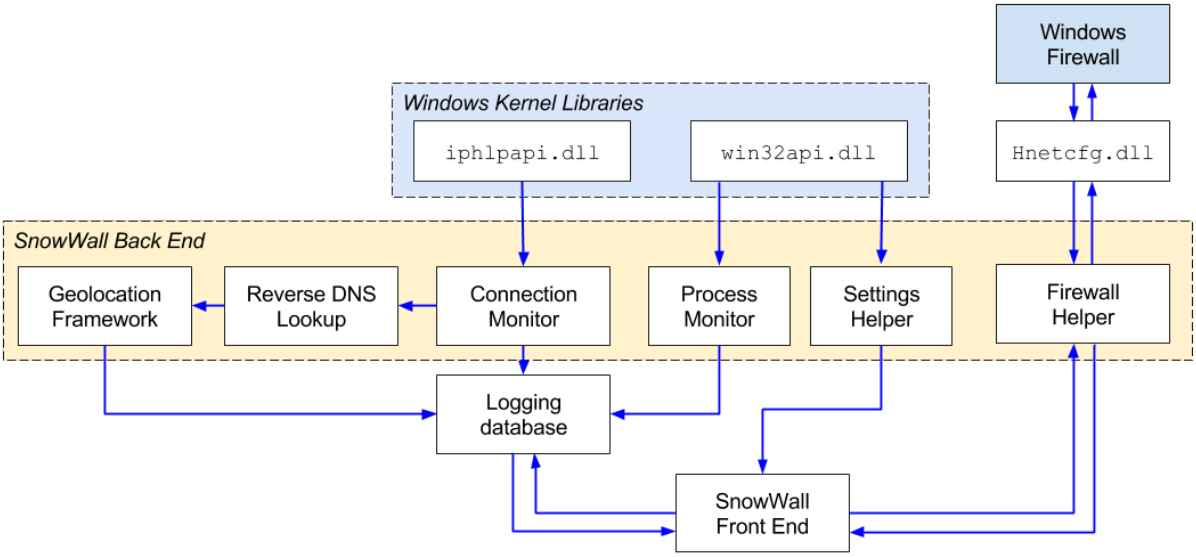


Figure 28: SnowWall interacts with Windows Libraries to monitor network connections and processes, manage settings and control the firewall.

3.3 Database

We have chosen a simple design for SnowWall's database. The model consists of five tables. The main table is the `ConnectionModel` table, which records all the connections that open on a user's machine, with adjacent information. The `ProcessModel` and `EndPointModel` store auxiliary information about processes and remote hosts. A connection can belong to only one process and link at most one endpoints.

The `ProgramRuleModel` table stores the custom rules created by the user. They store the name of the entity to be blocked, be it a country, an organization, or an application. The `ScheduleModel` stores user preferences about days of the week and times of the day when the rules should be enabled. Section 4.2 presents the design into more detail.

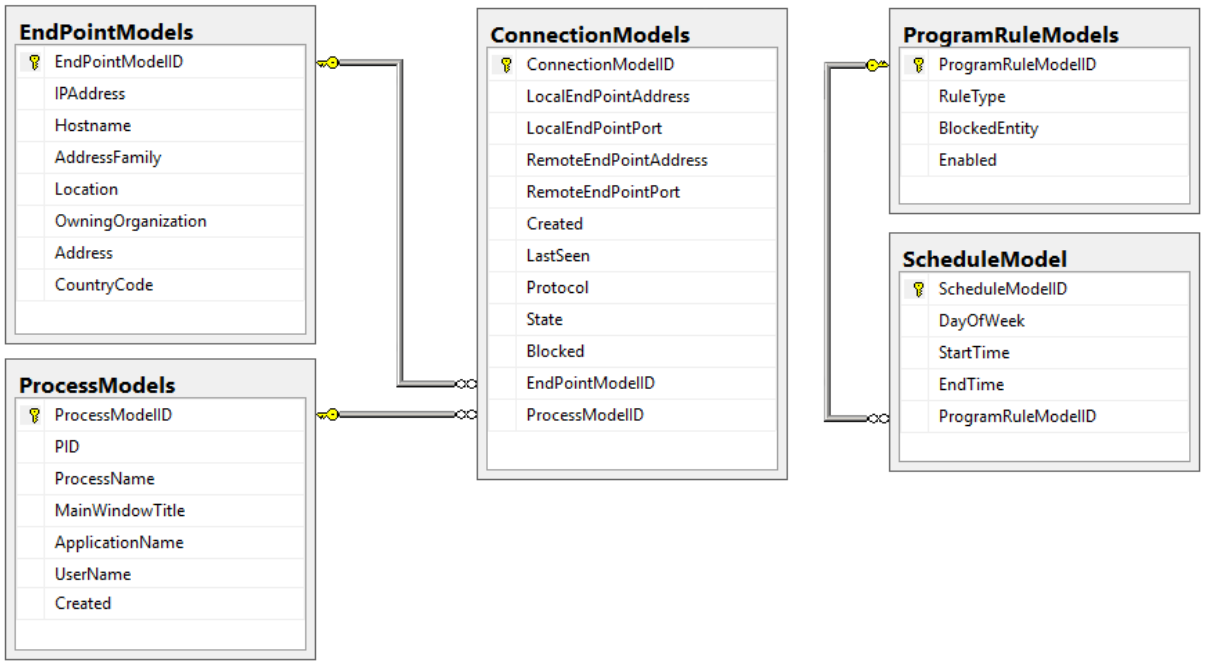


Figure 29: Class diagram of the SnowWall database exported from SQL Server.

3.4 Front-end Design

One of the main motivations for this project is to provide the users with a comprehensive front-end which is easy to use and understand. Furthermore, we wish that the front-end has a strong impact and motivates the user to take action for protecting their privacy. Because of this reason, we have chosen to use multiple sorts of data visualisations to better suggest the state of a user's privacy.

3.4.1 Human-Centered Design

Human-centered design[71] is a creative design and management approach which takes into account the human component at every step of developing a technology. This is to broaden the connection with the users of computer systems by focusing on the individual, asking for feedback from real-world possible users rather than guiding the design process by statistics.

To reiterate one of our objectives, we wish that SnowWall is able to shake the public apathy surrounding privacy. In order to achieve that, we have to focus on the user's psychology - why are they not interested in their privacy? What could change their opinion?

A starting point has been the survey on American citizen's privacy strategies [1] discussed in Section 1. The statistics in this study have allowed us to shape the problem and to launch a human-centered investigation by asking various Windows users about *what means they take to improve their privacy on their desktop, and what they like or dislike about them?*. We reiterate this was not a statistical investigation: we asked a small number of users for detailed answers, rather than the opposite. Table 7 presents the some of the answers and ideas we have found in this initial research.

How do you improve your privacy?	*What do you like about this method?*	*What do you dislike?*
Using Privacy Badger to avoid browser tracking	Ad-blocking	Does not stop tracking in Windows apps
Setting up a metered connection to avoid forced updates	Nothing	Missing security updates
Using TCP View and Windows Firewall to block	It reduced tracking	Too many steps
Disabling Telemetry with PowerShell scripts	It reduces tracking	Tedious, advanced and time-consuming
Using Ad-Block Plus	Ad-blocking	Blocks websites which detect ad-blocking
Creating firewall rules for everything	It stops tracking	Time-consuming and difficult to do
Using Tor browser	Anonymity	Slow access to the Internet
Never send diagnostics to any application	Nothing	Too many menus, I want all privacy settings in one place
Disabling Cortana	It silences Cortana	It still shows in Task Manager, if I try to kill it, it starts back again
Erasing history periodically, licensed anti-virus	Convenient and easy to remember	I'm not sure if they are the best, but I didn't research into this issue actively either

Table 7: Short survey about improving your privacy on a Windows system.

The front-end is split into multiple views. The rest of this section discusses the design process of each and concludes with a set of wireframes which describe the final design.

Map View Since the outset of this project we have been visualising a map-style main view which will locate each connection on their approximate latitude and longitude on a minimalist map of the world.

To confirm this we have shown a few users tools such as TCP view or WFN (discussed in Section 2.5) and asked them to comment on the interface. All of them have said that simply displaying a table with every open connection is not enough to stimulate the user into understanding the flow of data and where the network connections on their computer are directed, even if a flag or a country indicator is attached.

Furthermore, we have produced a quick prototype in Adobe Photoshop for the Map View and asked users to comment. We have learned that colour-coding is not sufficient for them to understand which organization is collecting most of their data. Thus, we have established colour-coding to rather describe whether a country is available or blocked, and we have moved on to follow the principle of simplicity and design a separate visualisation for the Organization View.

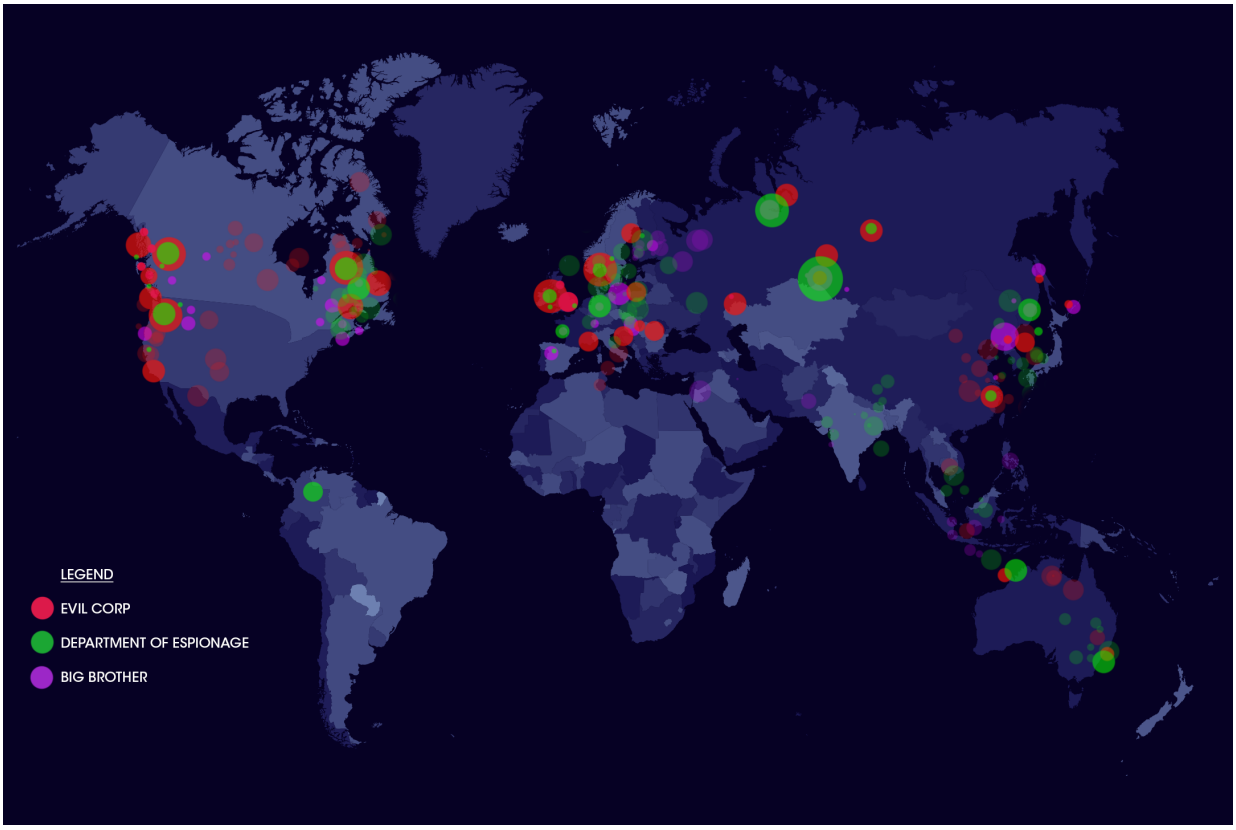


Figure 30: Initial prototype built in Adobe Photoshop for the Map View.

Organization View After surveying various visualisation techniques and asking Windows users to choose their preferred visualisation from the ones shown in Figure 35, and using inspiration from the Map of the Internet[22], we have settled on the *bubble graph*.

[22] `http://internet-map.net/`

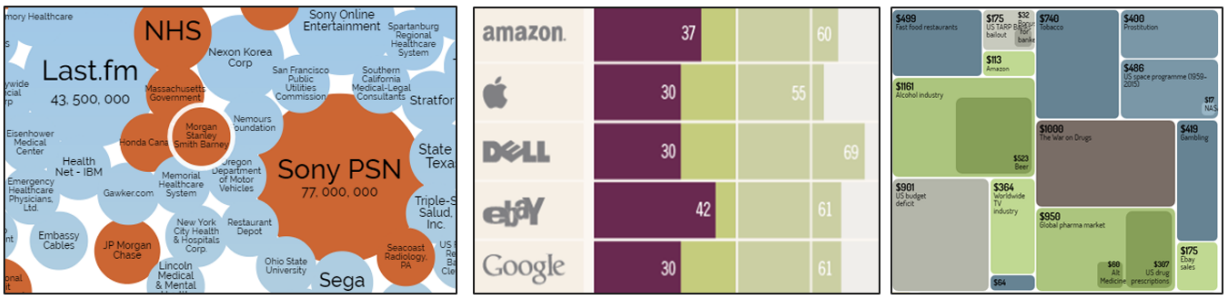


Figure 31: Three candidate types of visualisation for the Organization View (Courtesy of `www.informationisbeautiful.com`).

A bubble graph allows three-dimensional data visualisation thanks to the position on the grid and the bubble size. It is specifically due to the strong visual identity of the differently-coloured circles that we believe it will have an impact.

Process, Scheduler, Settings View The rest of the views on this application have been developed as a result of continuous feedback from various users on what features they would desire from a privacy application on Windows. The front-end for these views has been built in concordance with similar applications.

3.4.2 User Experience

Another crucial point in designing a powerful front-end is how the user interacts with the UI. The most important principle we have followed is to allow the user to create blocking rules as simple as possible.

The hierarchy of views follows this principle. The views are ordered from the most simple and quick to use to the most advanced:

- The first two views provide minimal amounts of information, but maximum efficiency in blocking connections. In the Map View and the Organization view, creating a blocking rule is instant: simply click on the country or the organization to block.
- A the user progresses through the Process View and the Scheduler, more advanced features and longer use-paths are involved. To block a specific process, a user has to select it from a list and use a blocking button. The Scheduler then allows the user to modify existing rules to be enabled or disabled at specific times, using setting forms.
- Finally, the Settings View contains a panel of administrative settings similar to the Windows Settings panel, as well as the option to export the data into a portable spreadsheet file for further analysis.

3.4.3 User Interface

The results of our design research have been aggregated in the wireframes presented below.

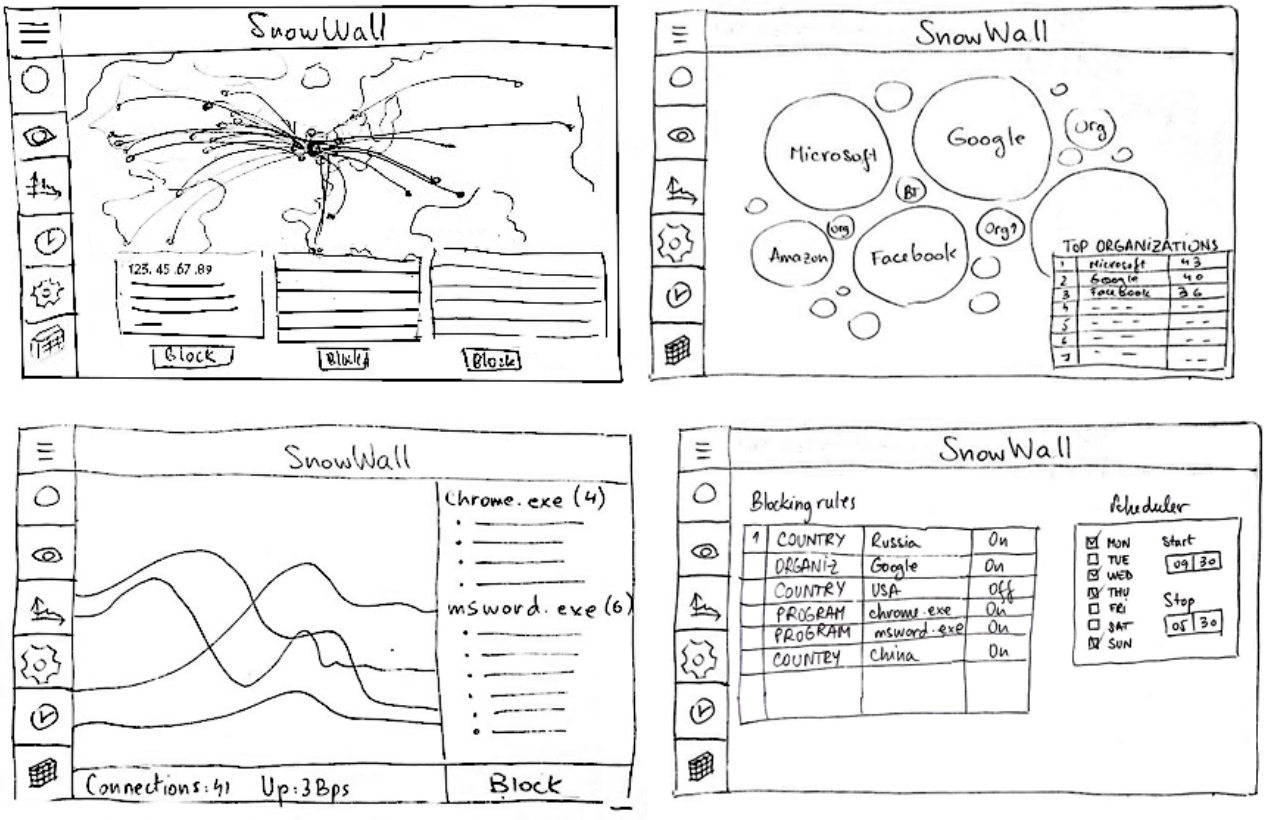


Figure 32: The SnowWall front-end. (a) The **Map View** shows the world map, with the connections marked as points on the map, and panels for connection details, the list of connections, and country statistics. (b) The **Organization View** shows a bubble graph of connections towards hosts owned by a specific organization, and organization statistics. (c) The **Process View** shows a performance graph of network bandwidth and a list of connections for each process. (d) The **Scheduler** shows a list of user-defined blocking rules and an interface to schedule them.

3.5 Product Outline

3.5.1 Target User

We have begun this project with the Windows 10 user in mind, but as we have learned about the newest privacy updates on Windows 7 and 8 we have reconsidered this decision. We have made sure that SnowWall is not only on Windows 10, but on Windows 7, Windows 8, and Windows 8.1 as well.

Given the large market share of Windows[72], which has been estimated to at least 80%, the problem we are trying to tackle is a universal problem. Windows is used by youths and seniors aside, by professionals, by developers, by creatives, and by gamers. All these users are under attack for their privacy, and all could benefit from a tool such as SnowWall.

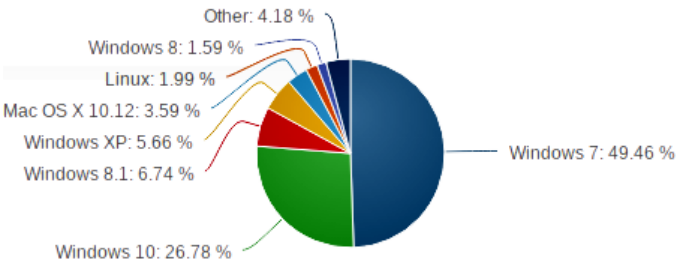


Figure 33: Pie chart of the market share of most popular operating systems for desktop. Courtesy of `www.netmarketshare.com`

In principle, we focus on the regular user, because we have learned that advanced users have a greater chance of protecting their privacy by creating firewall rules or monitoring connections themselves. But the fact that SnowWall is so simple to use yet so powerful it can be useful to advanced users too, even security professionals who need to make assessments about leaked information and log activity on client machines.

Additionally, we believe this tool can help users which have to work on Windows daily in a corporate environment, such as office workers, secretaries, analysts. SnowWall is easy enough to use for them to introduce it in their work-flow, for example to block access to content and ads which might interfere with their productivity, keeping themselves private at work, as well as protecting the integrity of their employer's company.

Last, but not least, the content creator and the gamer can both benefit from the advantages of monitoring and controlling their network bandwidth.

While we are aware this discussions is by no means an exhaustive survey of Windows users, we have concentrated on what we have considered to be the most urgent problems and desired tools, while keeping a clear view of the bigger picture in mind.

3.5.2 Identity

The name SnowWall has been chosen to refer to a firewall as well as a tribute to Edward Snowden. We have also designed a logo for the program, inspired by the skeumorphic design of logos for various security and anti-virus apps. Figure 34 shows the logo on the program's splash screen.

3.5.3 Features

The main features supported by SnowWall are listed below:

- List TCP and UDP connections currently active on localhost
- Log these connections in the database

- Preview a selected connection's details
- Show geolocated remote end points on the map of the world
- Save information about geolocated end points in the database
- Block all connections outgoing to a specific country by clicking on the country in the map interface
- Show a bubble graph of known owners of geolocated remote end points
- Block all connections outgoing to a specific organization by clicking on the bubble representing that organization
- Show a line graph of bandwidth usage
- Show a list of running processes with connections opened by each process
- Log running processes into the database
- Block all connections established by a process
- Export database logs
- Show a list of user-created blocking rules, such as country-specific, organization-specific, and program-specific
- Schedule the rules to be enabled or disabled at specific times of day
- Create firewall rules which mirror user-created blocking rules
- Impersonate an user with administrative permissions in order to gain access to the firewall and registry
- Customise less accessible privacy settings on Windows
- Customise SnowWall to run at startup and/or run in the background

Figure 34: The SnowWall splash screen.

4 Implementation

This section details the implementation process for this application. We begin by introducing the back-end components, which aggregate the entire functionality of the application. We proceed to discuss the most important design decisions and patterns used in designing database access. We then introduce a skeleton application which makes use of all the features already implemented, then a few implementation details for the front-end. We conclude with the most important risks and challenges undertaken throughout development.

4.1 Stage I - Back-end Components

The back-end is formed of a set of components implemented as Portable Class Libraries. These libraries can be attached to console applications as well as any type of front-end. The libraries define multiple classes and methods which interface with various APIs internally, as well as an external API with adapters to model classes used in the database.

4.1.1 Process Library

The Process Library provides the Process Monitor and the Process Utilities.

Unlike for network connections, event handlers can be attached to a process. The **Process Monitor** defines a static object which attaches two functions of choice to the event handlers of a process starting and exiting. In order to attach these handlers, we have interfaced with the **Windows Management Instrumentation** (WMI). We have actively chosen not to support querying individual threads, as there are already multiple process monitors supporting this feature.

WMI is a robust, reliable infrastructure for managing operations on Windows operating systems. Using WMI provides developers access to system management tools and services built into the operating systems.

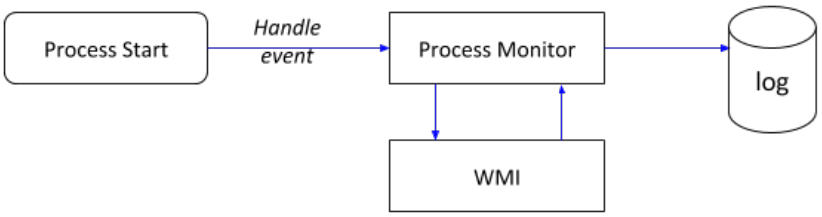


Figure 35: The Process Monitor intercepts process start or stop events and handles them with logging functions.

Whenever a process starts or ends, the system maintains a trace of these processes. WMI allows access to the trace via SQL queries. A query which retrieves the caller process is

attached to each event, and then the process details are passed to the delegate function which implements the handler. In our case, these delegate functions connect to the database to log the event.

The **Process Utilities** augment the existing C# process library with the ability to get the process's host application file path. The file path is required for the Windows firewall to create a rule for the process. Because a 32-bit process cannot query a 64-bit process and vice-versa, it is necessary to use WMI to query the running processes. Unfortunately, .NET has no means of detecting whether a running process is 32 or 64-bit. In order to identify whether a process runs on 64-bit, one needs to use the Windows Kernel API, which is accessible through `win32api.dll`.

Querying a system Dynamic Link Library is performed with a technique known as **PInvoke**[23]. The Platform Invocation Services allows managed code to call unmanaged functions implemented in DLLs. The DLL is imported into the application and with the `DllImport` attribute one can call functions defined in the DLL. Generally, these functions return byte arrays or C structures. These are unmanaged objects which cannot be used directly into a C# application. In order to build these into the corresponding class type, a technique known as **marshalling** is invoked.

For every .NET class type there is a default unmanaged type, which the Common Language Runtime uses to translate parameters when executing an unmanaged function call. In our case, we require a pointer to the running process which we retrieve by querying the process handle from the Process .NET class. The output is by default marshalled into a `bool`.

4.1.2 Connection Library

The Connection Library is formed of multiple classes and functions, whose purpose is to invoke the IP Helper API found in `iphlpapi.dll`, which contains the interface to Windows' TCP/IP network functions. The IP Helper API contains unmanaged functions which retrieve the TCP or UDP tables of active connections, and allow enabling and retrieving bandwidth statistics.

The Connection Monitor component polls for connections every second, in order to accurately detect the time when a connection changes state or closes. Whilst we are aware that polling is generally not desired for performance reasons, unfortunately the Windows networking APIs do not provide support for adding connection establishment or state change handlers due to the low level at which they operate.

The .NET stack permits querying for active connections in a simpler, managed way, but in order to retrieve information such as owning process ID and creation time one has to use the IP Helper. Because of the complex unmanaged types returned by IP Helper functions, in this case we had to perform non-default marshalling by declaring structures for each unmanaged type. Most of the functions both use as parameters and return integer pointers which link to byte buffers. To transform these byte buffers into the classes and objects

[23] `http://www.pinvoke.net/`

required for the application, we have defined enumerations and structures which describe the extended TCP and UDP tables and table entries and other auxiliary types.

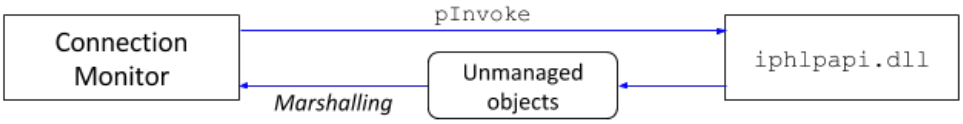


Figure 36: The public API for the Connection Library is the Connection Monitor, which returns lists of Connection objects used throughout the entire application.

4.1.3 Firewall Library

The Firewall Library interfaces with the Windows Firewall through its API found in `hnetcfg.dll` and defines required enumerations and classes for defining firewall rules. Out of all libraries used by SnowWall, its implementation was the most straightforward.

The Firewall Library is used by SnowWall to create and manage its firewall rules. The Firewall Manager, given a set of constraints such as a program rule blocking a country and a list of connections will automatically generate firewall rules for each connection which satisfies the constraints. The information about the program rule such as the country or organization name is stored in the rule description, such that when the user decides to delete or disable the higher-level program rule the individual firewall rules it manages are also automatically deleted or disabled.

4.1.4 Geolocation Library

The Geolocation Library contains .NET functions which connect to public DNS servers to query hostnames for IP addresses, as well as it interfaces with the device's location services (if enabled) to retrieve the current location to be displayed on the map.

It also includes a list of countries in the world[24], with their name, continent, and country code, which are used throughout the application and are required by the map front-end component to keep track of blocked countries.

The most important feature of this library is issuing HTTP web requests to various geolocation providers. The results are returned as JSON and processed inside the library into relevant information. The geolocation providers are discussed at large in Section **??**, where we discuss and evaluate the performance of geolocation.

Asynchronous programming is a technique of parallel programming in which a work runs on a separate thread from the main thread and notifies the calling thread of its completion, failure or progress. It is common practice that in applications with a user interface, in order

[24]Provided by SyncFusion.

to avoid blocking the UI thread, asynchronous calls are used to perform time-consuming processing tasks in the background. Asynchronous programming was required for this application because querying external services such as DNS or WHOIS involves the overhead of establishing a connection to the service.

4.1.5 Settings Library

This component provides both settings specific to running SnowWall as well as an interface to some Windows privacy settings. For example, it allows the user to specify whether they prefer to have SnowWall run at start-up, to give their permission for the process to run as administrator in order to have access to the Firewall and registry. The settings library also contains functions which add specific registry keys or run PowerShell scripts, for example for removing Metro apps or disabling Cortana. The implementation for these tasks has been mostly done with standard methods, because Microsoft provides managed classes and functions for editing registry or running PowerShell as a .NET framework component.

In order to give administrative permission we have used a technique known as **Impersonation**. Impersonation is performed by pInvoke of unmanaged functions found in two Windows kernel libraries: the Advanced API `advapi32.dll` and User Environment API `userenv.dll`. The main idea is to use the functions in `userenv.dll` to create an *environment block* owned by a specific authenticated user and use the `advapi32.dll` functions to launch programs only inside this block. The Windows Kernel API `win32api.dll` is used to open and close the process handle of these programs.

4.2 Stage II - Database

4.2.1 Tables and Views

The database tables for the five main entities used by SnowWall have been introduced in Section 3.3.

But a relevant implementation detail is the use of SQL Views to support the statistical visualisation such as counting the connections for each country. While Entity Framework with its adaptive query language LINQ should make redundant the use of SQL views, after some experimentation we have discovered that using SQL views is faster and more efficient than implementing these counters and features in code. Thus, we have designed class entities which map to these views and retrieve the statistics calculated by the SQL engine.

The views we have defined are Country Statistics, Organization Statistics, and Process statistics, which show an all-time count of connections opened towards each country, organization, and the active connections opened by each application respectively.

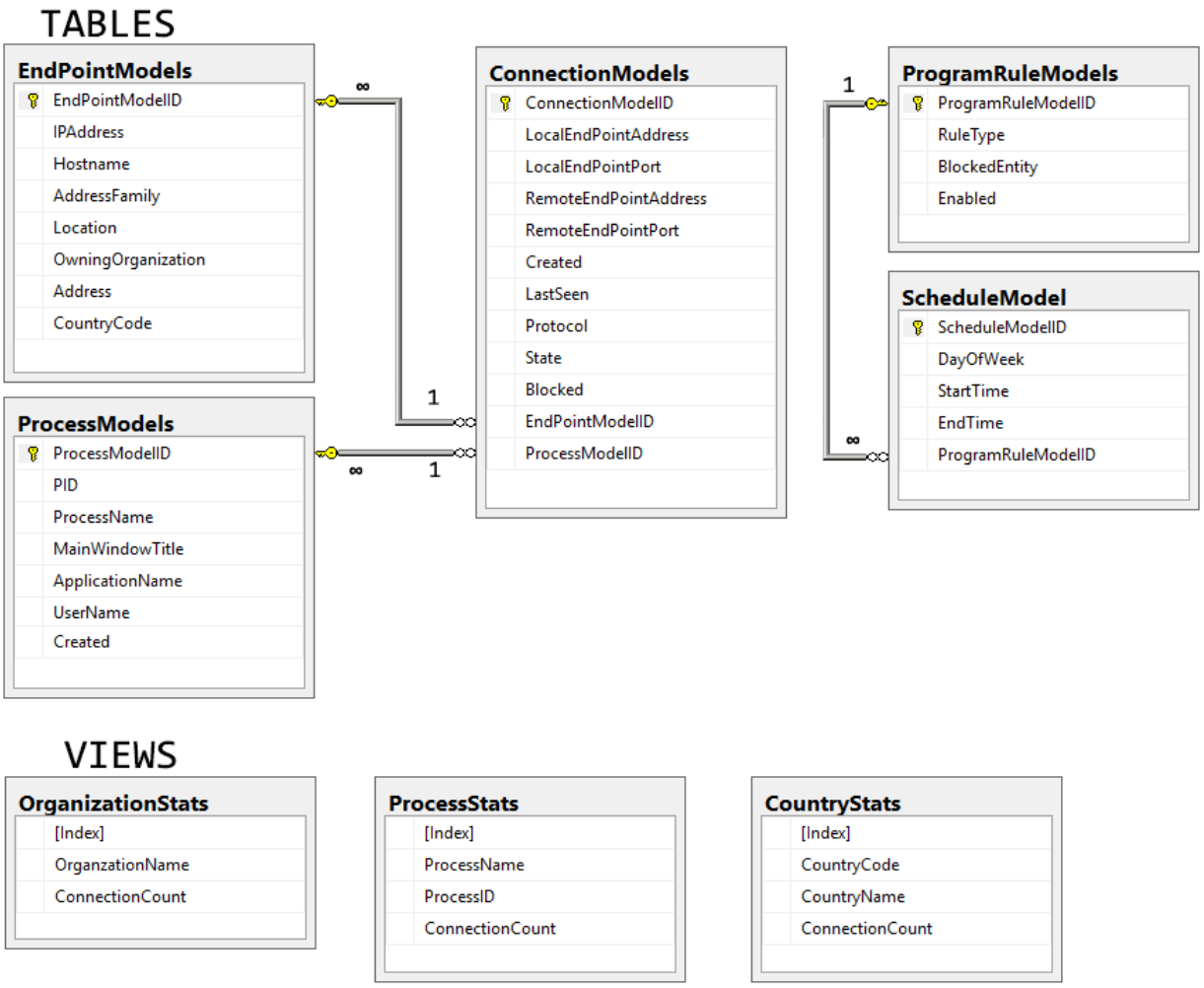


Figure 37: Tables and Views used by SnowWall.

4.2.2 Data access

Entity Framework allows creating a means of data access known as a **data context**. A data context contains collections of entities and allows CRUD (create-read-update-delete) operations on these collections.

A **repository** is a design pattern which provides an abstraction layer between the data access layer and the business logic of an application. It mediates between these layers and acts like an in-memory object collection. The business layer of an application constructs queries which are issued to the repository, and the repository satisfies the queries by either returning the relevant objects or modifying objects from its collection. Behind the scenes, the repository performs the actual data access to save the changes into the database. The repository allows a separation between the data layer and the domain, which can facilitate unit testing.

A repository can save changes on-the-run or persist the changes and batch save them as

specific times. The latter is known as a **unit of work** pattern, which allows to commit multiple changes as a single database transaction. A unit of work generally encapsulates the repositories and the database context, allowing flexibility to change implementation of the data layer, as well as packaging a set of changes into a single access to the database. This is of crucial importance for data integrity: suppose an inconsistent entry is added to the database, whose primary key is referenced by another object. If the two objects are saved into the same batch, then failing to add the first will rollback all the other changes, thus avoiding an inconsistent foreign key.

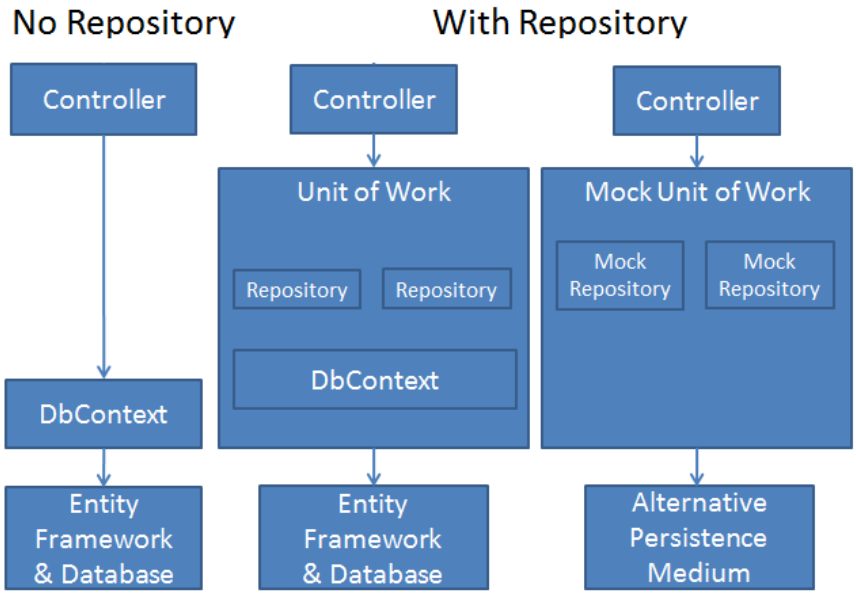


Figure 38: A unit of work containing a repository for each entity model separates program logic from the data access layer.

From an implementation perspective, for each entity type we have used a repository interface and a repository class implementing that interface. The unit of work class declares a repository for each entity type and is used as a model in the front-end. Figure 39 shows a diagram of the three main repositories implemented in SnowWall.

For simplicity, the most basic business logic such as queries which allow filtering the results of a select query are also part of the repositories. These methods are implemented as parallel queries using lambda functions. Since the database may hold thousands of connection objects and process logs, filtering or mapping them in parallel is a large performance improvement.

Figure 39: UML Diagram of repositories used by SnowWall.

4.3 Stage III - Simple monitoring app

After successfully building the back-end libraries and data access, the first test was to produce an application which will use these components and show connections and firewall rules in real-time. A screen capture of this application at work can be previewed in Figure 41. We have chosen a simple list interface, with a button which allows creating a firewall rule to block a selected connection. This intermediary step in development has allowed catching many bugs which have not been identified in individual unit-testing of the libraries and data access layer, as well as finding solutions for performance issues such as caused by the need to perform asynchronous tasks at the same time with displaying real-time results.

This application uses a timer which queries for active connections every second. While we are aware this is a very small interval, which is not necessarily required since no network connection can be established this quickly, this choice has been made to allow the closest possible to a live feed. The Connection Monitor and the Geolocation Helper calls are issued once every second, and after completed they update the database model. Then the view, having its controls bound to the unit of work, displays the updated data.

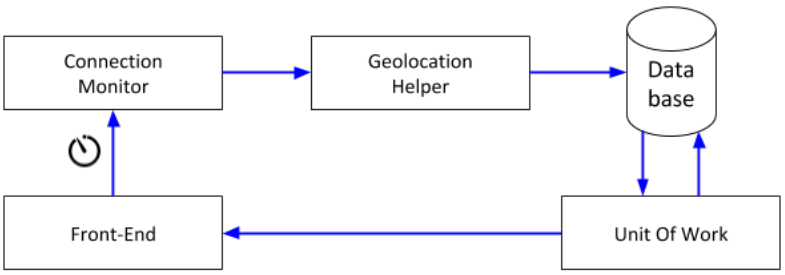


Figure 40: The timer triggers the connection monitor, which in turns grabs geolocation information, and saves the connections in the database.

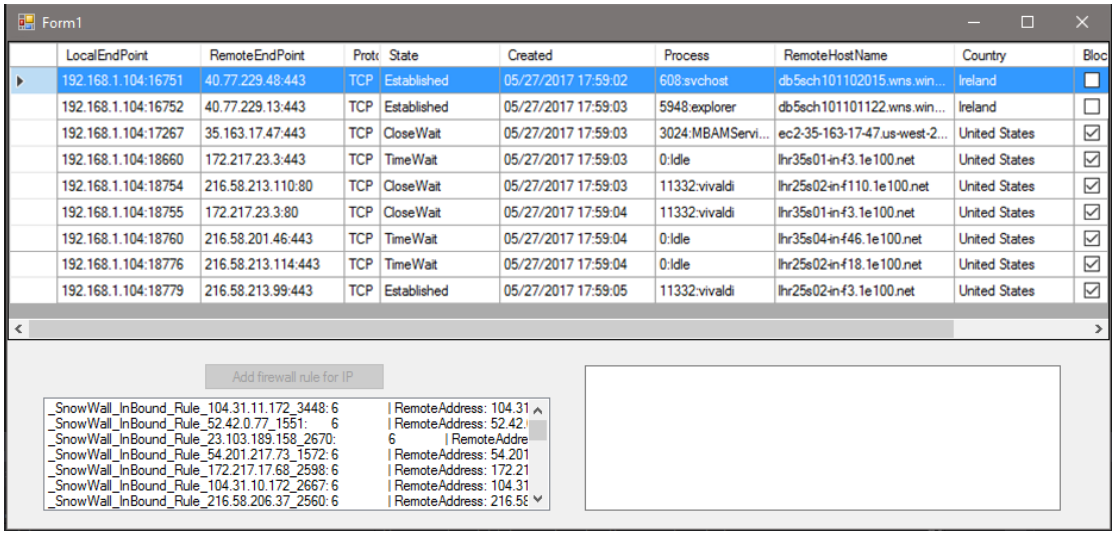


Figure 41: A simple application which tests back-end and database.

4.4 Stage IV - Building the front-end

The final front-end implements the different views we have discussed in Section 3.4. Only the Map View and the Process View function in real time, thus they make use of an architecture similar to the one in Figure 40.

The design pattern generally implemented in Windows Forms applications is known as **Model-View-Presenter**. MVP is a derivation of the Model-View–Controller (MVC) pattern, and is commonly used for building user interfaces. In MVP the presenter assumes a functionality similar to the controller, interacting with the model and updating it as a result of user input, but also includes presentation logic such as defining view interactions. The views are the Windows forms, which may bind models to UI components directly, without the aid of the presenter - such as the list of connections in our case.

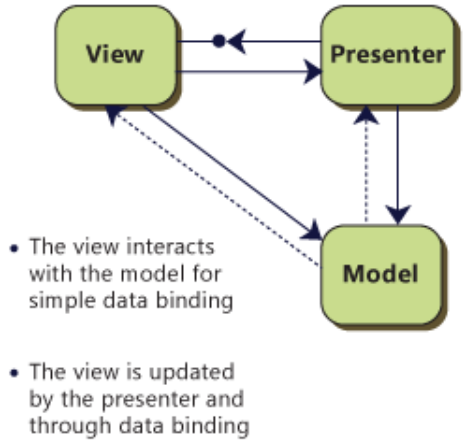


Figure 42: The model-view-presenter architecture (courtesy of MSDN)

The Map View is implemented with the use of a Map control provided by SyncFusion[25]. The Map control supports multiple layers, which can display the contour of countries, colour-code countries whether they are blocked or allowed, and show annotations at specific coordinates which mark the geolocation of active connections. The contour of countries is locally generated as a graphical entity based on an offline shape file `.shp`, the same file type used by GPS. We are aware this is not a resilient design choice as the political map of the world may change, but at the same time this solution removes the overhead of contacting a maps provider such as Google or Bing, as well as embeds the contour information of each country required to highlight it in a specific colour, which is not supported by any map provider with a .NET control.

The Organization view is implemented with with the SyncFusion Bubble Graph control, which is bound directly to the database view of organization statistics. We hope that in a future version of SnowWall we would be able to improve this visualisation with

The something m Procesore dynamic.s view shows a performance graph of statistics, which has been implemented by plotting the usage function over time on a WinForms Panel control using the system drawing functions. The plot updates every 2 seconds by default but allows the user to change the speed of updates from 2 times a second to up to once per hour.

[25] `https://www.syncfusion.com`

4.5 Risks and challenges

4.5.1 Risk Assessment

Using unmanaged functions The process of transforming unmanaged structures and types into managed types is notoriously error-prone and difficult. We had to thoroughly inspect the Microsoft documentation of these data types as the main risks involved in this case are possible buffer overflows or underflows which maintain a high security risk.

Polling Polled operations refer to actively sampling the status of an external device by a client program as a synchronous task. This poses various risks such as delaying the UI thread if too many connections are open. In order to mitigate this issue, we have separated the polling operations onto separate threads, using a timer which dispatches asynchronous tasks. The UI thread reloads on the main thread and grabs available information straight from the database. This might incur some delay in presenting the live feed of connection, but it is the best solution we have found to solve this issue.

Registry editing Editing the Windows registry is notoriously known as the main cause of multiple exploits and security risks. We are aware of this and we only implement functions which perform very strict tasks without interfering with user input, i.e. all the details of these operations are encapsulated.

Geolocation Research discussed in Section 2.3 shows clearly how most client-independent techniques for geolocation are prone to errors and inaccuracy. This poses the risk of incorrectly geolocating a host in such a way that may alert an user unnecessarily, or even worse, conceal vital information about the location and owner of a host. We have attempted to mitigate this risk by providing a high-level geographical means of blocking, which can only block a country. While this comes at the cost of flexibility, existing geolocation is mostly resilient to country identification errors, as each continent has its own research community which maps the IP address blocks to countries, but cannot guarantee accuracy for cities and regions.

Unfortunately, when it comes to identifying the owner of an IP address, we are aware that services such as WHOIS would rather direct to the address of the Internet Service Provider which lease the IP addresses to private entities rather than to the entities themselves. In Section 5.4.2 where we explain how we tried to evaluate and improve the accuracy of this information.

4.5.2 Challenges

Building a geolocation framework After performing comprehensive research on the matter of geolocation, we have found ourselves interested in providing an implementation of a state-of-the-art model. Unfortunately, the quality of a geolocation framework is dictated by the resources at hand. A technique such as multi-lateration requires a large amount of active landmarks (ping servers) which unfortunately we did not have access to, as many research projects for such purposes, for example PlanetLab [68] and Geant [66] have now ended. Thus we have only used available services, mostly data-driven, to aggregate existing data. We are aware that these methods impose high inaccuracies but we hope that for the task at hand they are sufficient for delivering a product with our requirements.

The development stack One of the main challenges to overcome in this project has been to choose the right software stack to develop this application. At the end of the first three stages, an important decision about the front-end had to be made. Whilst we would have desired a more flexible front-end, e.g. JavaScript-based, it would have been impossible to pair such a front-end with the current back-end, because of restrictions in the Microsoft Development stack.

In March 2017, Microsoft has released the Universal Windows Platform development tools. At this time the first 3 stages of implementation have already been completed. We have considered the front-end solutions provided by this framework and even built a mock implementation which can be previewed in Figure 43. While this platform allows modern features such as building seamless, interactive user interfaces which support animations and flexible screen sizes, we realised that this platform is not compatible with back-ends built in .NET. A large amount of extra development work would have been necessary to build the application with UWP, so unfortunately we had to discard this option.

The remaining choice was between the Windows Forms and the Windows Presentation Foundation (WPF) technologies. Initially, it seemed that Windows Forms is a framework too limited to produce the kind of front-end we desired. We have attempted to build a WPF application, but unfortunately the overhead of getting acquainted with this framework would have highly limited the output of development. Finally, the solution came with the SyncFusion[26] stack of tools, which provided the front-end components `Map` and `Bubble Graph` necessary for this application.

[26] `https://www.syncfusion.com`

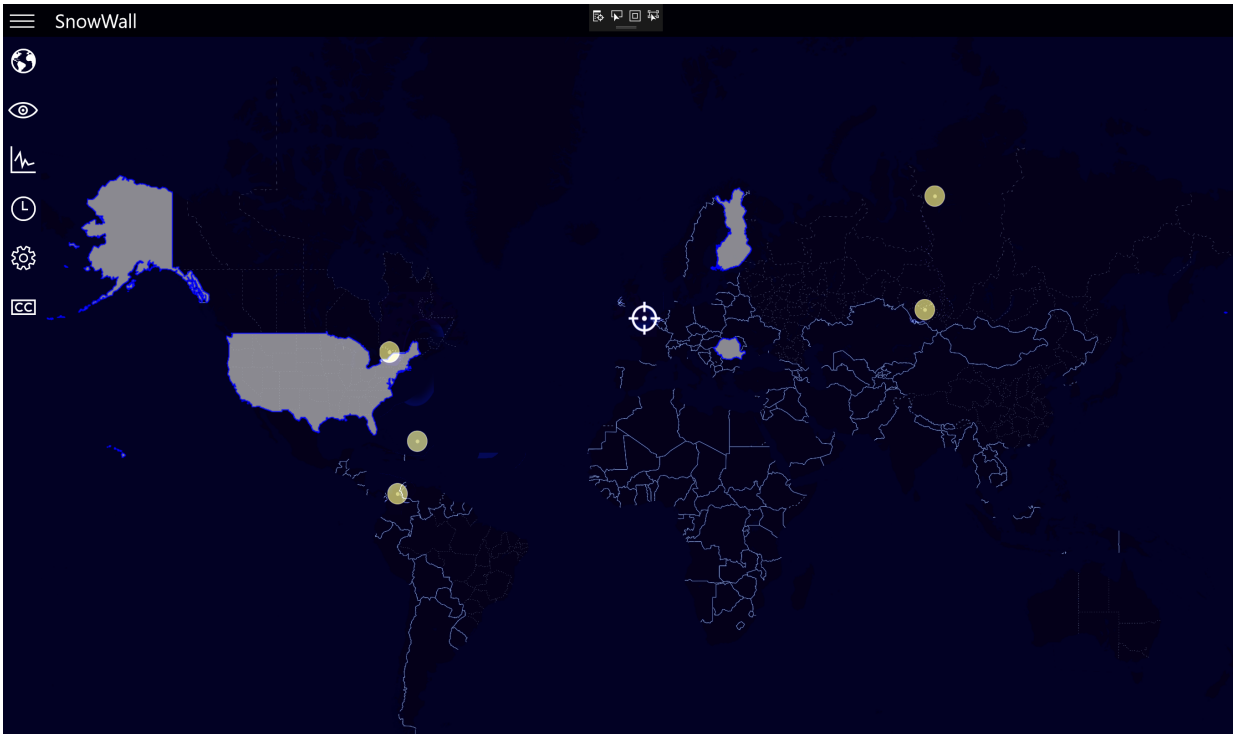


Figure 43: An implementation of the Map View as an UWP app.

5 Program Evaluation

This section outlines the main methods we have undertaken to evaluate this product. We begin with an demonstration of the user interface, and we show that the functionality has been correctly implemented. Then we show the results of the final stage of human-centered design research, namely alpha testing by different users, and discuss the quality of the final product from their perspective as well as from the perspective of the list of features presented in Section 3.5.3.

We then proceed to discuss the geolocation methods we have used and how we have evaluated their accuracy and aggregated the data, as well as some quantitative analysis about the geolocated data.

5.1 Overall Functionality

5.1.1 Experimental setup and participants

The experimental setup for the qualitative evaluation is as follows: SnowWall has been run with Administrator Permissions on a Windows 10 machine. A tester has been interacting with SnowWall and documenting the results of each action.

The test was executed twice by two participants: the lead developer and a third-party Windows user.

5.1.2 Methodology

We have taken the actions described in each of the tests below, and repeated them a few times in order to verify program resilience.

1. **Rules are displayed in real time** To test the Map View, we began by allowing SnowWall to run for a number of minutes while using the Internet, such that we could observe that the connections and locations on the Map View are updated in real-time. We have compared our results with the results of TCP View.
2. **Blocking a country creates firewall rules** We invited the tester to block and unblock countries. We have observed that the outbound connections from the blocked countries disappear from the map, and also from TCP View. We have used the Windows Firewall with Advanced Security to verify that the required outbound rules have been created.
3. **Unblocking a country deletes firewall rules** The tester has proceeded to unblock the previously blocked country, and we have then verified that the firewall rules have been removed.

4. **Blocking an organization creates firewall rules** The tester has clicked on a bubble representing an organization to block that organization, then we have checked the Windows Firewall to see the rules have been aded.
5. **Scheduling a rule** To test effective scheduling, we have asked the user to set up a schedule for one of the previously created rules such as all the rules would be disabled a few minutes in the future and then re-enabled again. We have verified with the Windows Firewall that this was the case.
6. **Changing settings** We have asked the user to use the settings to disable Cortana and to set SnowWall to run at start-up. We have then used PowerShell scripts and the Registry Editor respectively to verify this was the case.

5.1.3 Results

The figures below show screen captures taken throughout the test methodology which describe successful completion of tasks.

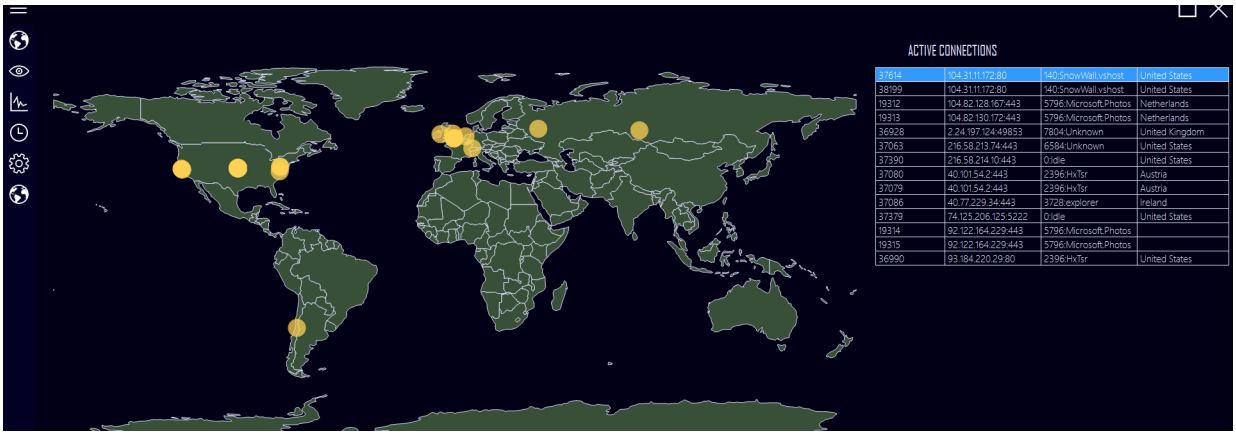


Figure 44: An initial preview of the map and connection list in Map View. No country is blocked.

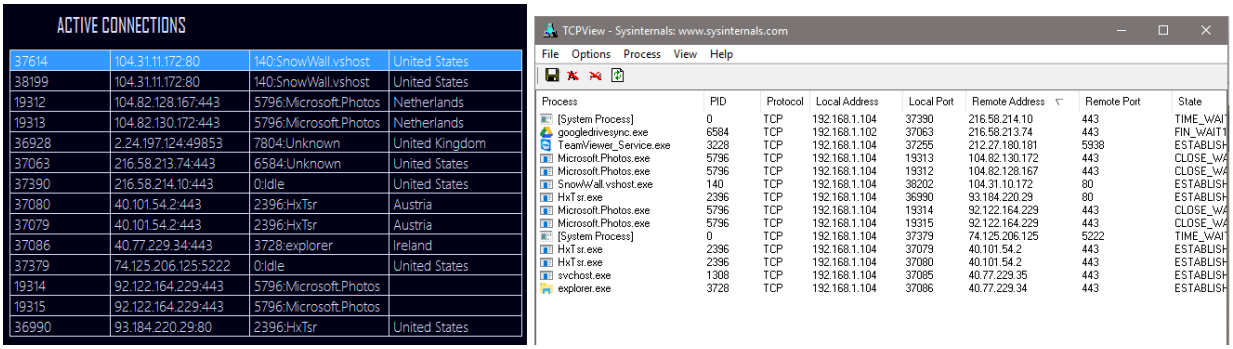


Figure 45: The active connections showed by SnowWall vs. TCP View.

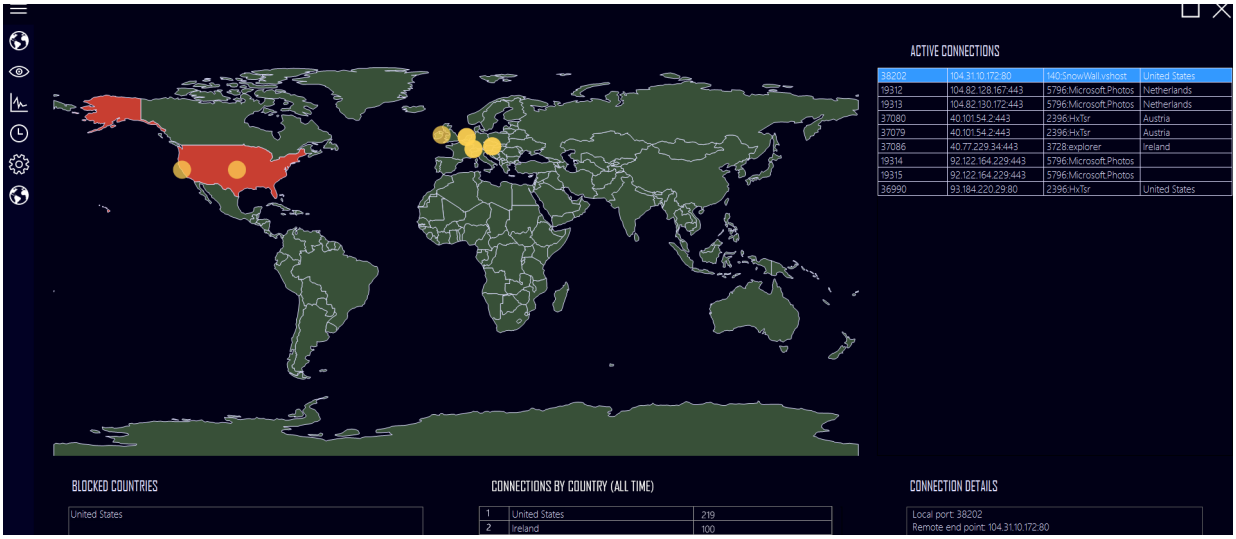


Figure 46: Blocking a country has highlighted it in red and removed some of the connections.

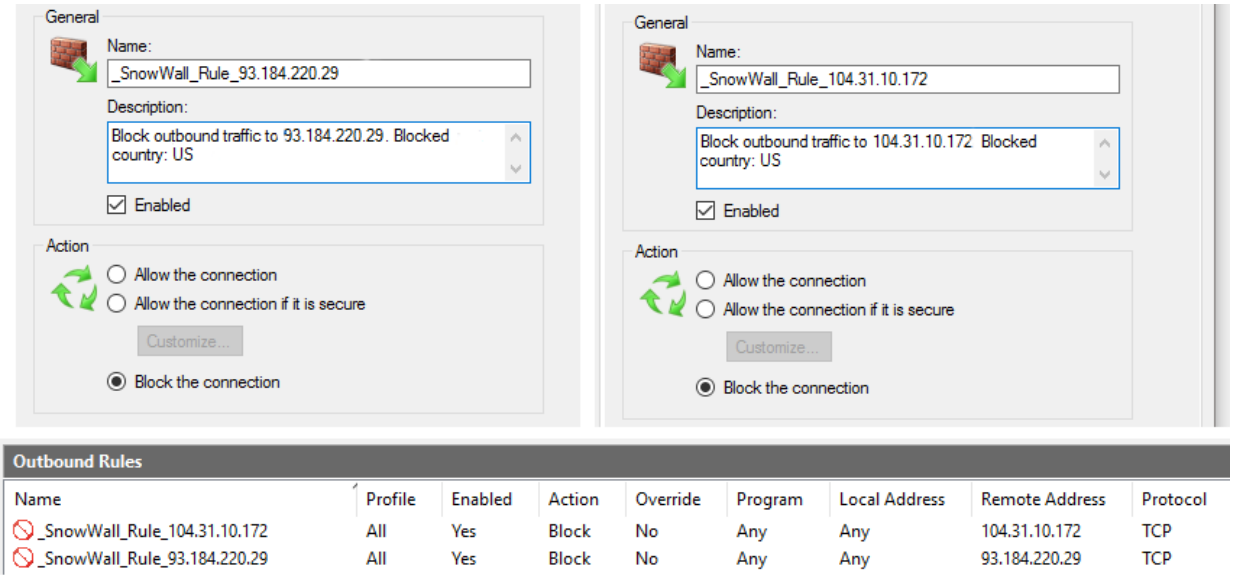


Figure 47: The two connections which have dissapeared from the map and list have been successfully blocked by firewall rules.

When running the first test, we have noticed two times that that one connection is missing from SnowWall and we suspect this is due to the geolocation which delays showing the connection by a few fraction of a second. Additionally it seems that SnowWall could not correctly identify the process name and path of one program, namely the Google Drive application.

Through the second test we have learned that the map delays displaying connections compared to the list as we have noticed they disappeared faster from the list.

The rest of the tests have been completed without issues. Table 8 below summarises the completion task of each of the tests above. We have rated each task with a rating out of 5

in order to describe how much we consider the software fulfills the task.

Participant	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
1	4	4.5	5	5	5	5
2	5	4.5	5	5	5	5
Completion rate	90%	90%	100%	100%	100%	100%

Table 8: Task completion rate for the functionality test

5.2 User Testing

5.2.1 Experimental setup and participants

After verifying that the software achieves its overall functionality, we have proceeded to alpha testing. In this case, the participants were invited to use SnowWall freely and answer a set of questions about the features and presentation for example to rate the usability. A short introduction has been offered to each user such that they are aware what SnowWall is used for.

For this experiment, we have invited 4 Windows non-advanced users to test the application by sending them over the program and asking them to use it for their daily tasks.

5.2.2 Results

The results have been mostly positive. All users were very excited with the design and have found it very easy to use. The UX has been evaluated positively, but users have complained the program is not extremely responsive. One of four testers has encountered software crashes. The user satisfaction with the product has been high and we have been congratulated for the design and features, but at the same time warned on the issue of lack of responsiveness.

Most users have added that they would gladly used the product if it were freely available, and have shown concern when seeing the connections on the map. With the results of this initial user evaluation, we believe we have reached the last step in our human-centered design plan.

This test, while by no means exhaustive, has showed us that while this product has great potential and seems very attractive to users, it still needs to be improved. We hope that in the future we can address the issues pointed out by this testing phase as well as the user feedback and proceed to experiment with a large sample set.

Question/Participant	1	2	3	4	Overall
Design (logo, UI)	5/5	5/5	5/5	5/5	100%
Ease of use	5/5	4.5/5	5/5	5/5	97.5%
Responsiveness	3/5	5/5	3/5	4/5	75%
Crashes or errors? (yes/no)	yes	no	no	no	75%
Would use daily	4/5	4/5	5/5	5/5	90%
Would recommend	5/5	5/5	5/5	5/5	100%
Satisfaction with the product	4/5	5/5	4.5/5	4.5/5	90%
Total					89.64%

Table 9: User rating results of alpha testing, with the lowest expressing dislike/disagreement and the highest expressing like/agreement, aggregated as percentages.

5.3 Professional use

One of the most successful results in the evaluation phase has been the opportunity to test SnowWall in a professional environment. Specifically, we have used SnowWall to aid with security investigations due to data breaches, by allowing the suspect machines, (which run Windows 7 and belong to a stock-trading company in Central London), to be logged for 24 hours and providing insights into any breaches. This has highly accelerated the previous operating procedure and provided steps which would have been unfeasible to perform by a single security analyst in a few hours. We have received both positive and negative feedback, but we can rest assured we have seen SnowWall in action in a demanding and urgent task.

5.4 Geolocation

As we have previously mentioned, SnowWall is agnostic to the geolocation method used by the backend. Given that we were unable to implement our own solution, we had to evaluate existing geolocation providers in order to choose the best one for our application.

5.4.1 Evaluation Criteria

The five main criteria for evaluation have been the following:

- speed, i.e. the time required to make the request
- limitations, i.e. do they limit the amount of queries in a given time-frame
- accuracy, i.e. how fine-grained are the results
- consistent, i.e. are the results of geolocation consistent over all providers?

- organization information, i.e. do they provide information about the owner or the ISP leasing an IP address?

5.4.2 Geolocation providers

The geolocation providers compared in this test offer free public RESTful APIs for querying an IP address. Below is a short survey of these providers and the kind of geolocation techniques and data they use.

1. **FreeGeolocationAPI**[27] uses the GeoLite database provided for free by MaxMind
2. **ip-api.com**[28] uses public DNS information supplied by GoogleDNS and OpenDNS. They do not make public the source of their geolocation information.
3. **ipapi.co**[29] Is a commercial provider offering a free service. They do not make public what their source is.
4. **Nekudo**[30] also uses the GeoLite database

5.4.3 Reference Data

The accuracy of the information retrieved by the services below has been tested against a dataset of geolocation information provided by **Netcraft**. The NetCraft IP address information is derived from public number registry databases. Between them, the five databases contain approximately 10 million netblocks of information which form the WHOIS database. The majority of netblocks assigned to physical locations are registered in these databases by large ISPs. The ISPs delegate the blocks of IP addresses down to their customers. NetCraft aggregates these services daily into a database which has been used to verify the consistency data geolocated by other services.

The information available in this database is:

- source database name (e.g. RIPE, APNIC, AFRINIC, LACNIC, ARIN)
- country code
- description (usually the owner name, then an address if available)

5.4.4 Experimental setup and methodologies

In order to perform this evaluation we have chosen 25 IP addresses at random from SnowWall's database, detected over the course of four days. We have set up a set of

[27] `https://freegeoip.net`
[28] `https://ip-api.com/json`
[29] `https://ipapi.co`
[30] `http://geoip.nekudo.com/`

scripts which query geolocation information from each provider for all the addresses, whilst at the same time timing the process, catching for errors such as reaching a maximum quota and logging the results. We have queried NetCraft's database for the same IP addresses and logged the results, then proceeded to compare geolocation information for each IP.

5.4.5 Results

The results for each of the evaluation criteria are discussed below.

Speed The table below shows the average speed of completing the request calculated by running and timing each query three times, once a minute.

Framework	Test 1	Test 2	Test 3	Average time
FreeGeolocationAPI	3.626s	3.360s	3.313s	3.436s
ip-api.com	15.877s	15.893s	12.533	14.767s
ipapi.co	8.001s	9.143s	7.298s	8.147s
Nekudo	2.018s	1.985s	1.844s	1.949s

Table 10: Elapsed time for each provider to geolocate a sequence of 25 IP addresses.

We can conclude that by far Nekudo is the fastest of the three, followed by FreeGeolocationAPI.

Limitations By analysing the table below it is easy to see that ip-api.com and ipapi.co are unfeasible for the scale of SnowWall, which may issue hundreds of queries per minute for a user who e.g. does not use anti-tracking methods in their browser.

Framework	Maximum Queries
FreeGeolocationAPI	15,000/day
ip-api.com	150/minute
ipapi.co	1000/day
Nekudo	no limits

Table 11: Limitations of each framework as per the number of queries in a fixed amount of time.

Accuracy The evaluation for accuracy has been performed by analysing the results for the 25-IP Address query. The table below shows best-case results for each framework.

Whilst Nekudo achieves city-level granulation, their API does not provide region information. Thus we were unable to perform a conclusive evaluation.

We have also calculated the percentage of results which have achieved the named accuracy, and the table below shows a performance mark of each framework on our test dataset.

Framework	Country-Level	Region-Level	City-Level	Postcode-Level
FreeGeolocationAPI	✓	✓	✓	✓
ip-api.com	✓	✓	✓	✓
ipapi.co	✓	✓	✓	✓
Nekudo	✓		✓	

Table 12: Best-case geolocation accuracy.

Framework	Country-Level	Region-Level	City-Level	Postcode-Level
FreeGeolocationAPI	92%	76%	76%	52%
ip-api.com	100%	100%	100%	52%
ipapi.co	100%	100%	100%	52%
Nekudo	92%	N/A	76%	0%

Table 13: Performance of geolocation accuracy.

The best results are achieved by ip-api.com, ipapi.co and RIPE. It is worth mentioning that the result of geolocation for the first two have been entirely identical for each IP address on our dataset.

Consistency The table below shows a pair-wise comparison of the results returned by each provider, namely the percentage of queries which have yielded different results in Provider A from Provider B. We suspect that ip-api.com and ipapi.co use the same provider as their results have been identical, even if incorrect with respect to WHOIS data.

	FreeGeolocationAPI	ip-api.com	ipapi.co	nekudo
FreeGeolocationAPI		32%	32%	0%
ip-api.com	32%		0%	32%
ipapi.co	32%	0%		32%
Nekudo	0%	32%	32%	

Table 14: Pairwise comparison of inconsistencies in geolocation data on our small dataset for each provider. The values represent the percentage of queries with the same result.

Additionally, the following table shows the percentage of conflicts in geolocation between each provider and the results on the reference dataset. Whilst we cannot guarantee that a dataset solely based on data-driven methods is entirely accurate, we can guarantee country information with much better precision since all the information are taken from regional authorities. This has allowed us to mediate between two geolocation providers which have placed the same IP in two very different countries - for example, Nekudo and ipapi.co.

Organization Information The table below shows which providers return organization information on top of the geolocation data.

	Inconsistency vs. reference data
FreeGeolocationAPI	32%
ip-api.com	12%
ipapi.co	12%
Nekudo	36%

Table 15: Comparison of inconsistencies in geolocation data for each provider versus the reference dataset. The values represent the percentage of queries with the same result.

Framework	Organization information
FreeGeolocationAPI	
IP-API.com	✓
ipapi.co	✓
Nekudo	

Table 16: Providers which support organization information.

5.4.6 Conclusions

Following the results above we have decided that for the real-time usage required by the Map View the best case scenario is to use one of the two fastest frameworks. Considering also the support for a more detailed address we have chosen FreeGeolocationAPI to run in the back-end when we query connections. It is worth noting that while in the case of this evaluation the queries have been executed sequentially, SnowWall issues the geolocation queries in parallel, thus the overhead of creating a connection to the server is minimised.

An issue caused by these results is that no service which provides organization information is fast enough to show results in real time. We have resorted to hide the owner information from the Map View, and to run the RIPE geolocation in the background on the Organization View, since the Organization View is not required to run in real time.

6 Case Studies

After the first stages of implementation have been completed, we have started to use SnowWall in order to evaluate the privacy leaks currently occurring on Windows systems with popular software. This section presents the experimental setup, test methodology, and the main results we have found in these experiments.

6.1 Experimental Setup

Our experimental setup includes two virtual machines hosted with Oracle VirtualBox[31], one with Windows 7 and one with Windows 8.1 installed, including the latest updates. No privacy settings have been initially setup, other than the system defaults. We have also run a Windows 10 Creator's Update test, on an out-of-the box Microsoft Surface with the latest updates installed. The table below lists the software products we have either installed or focused on for each system in this experiment.

	Windows 7	Windows 8.1	Windows 10
Microsoft Office Word 2016	✓	✓	✓
Microsoft Office Excel 2016	✓	✓	✓
Microsoft Outlook 2016	✓	✓	✓
Office Click-To-Run		✓	✓
Google Chrome	✓	✓	✓

Table 17: The list of software products evaluated with SnowWall on each system

Furthermore, we have used SnowWall to inspect the process and network activity of the system when performing simple tasks such as searching for a file or creating and deleting files and navigating through the file structure. With this experiment we intend to see whether these tasks are also an issue, as it has been claimed even search queries or the content of text files is sent to Microsoft.

6.2 Methodology

Each test has been run with only SnowWall and the relevant software running, each after a system restart, in order to avoid processes which may still run from previous tasks. The experiment required that the user starts up SnowWall and then begins to use the software for five minutes, which includes exiting the program and starting it back again, then turns off SnowWall and reboots. In the browser tests, only three websites have been navigated to, Google, Facebook, and Messenger. We have filtered out the direct connections to these

[31] `https://www.virtualbox.org`

websites from the set of connections logged throughout the experiment. In the email test, only drafts have been composed and no e-mail has been sent.

We have also performed a simple "system" test in which the user was just required to browse the file system and use search.

6.3 Results

We have aggregated the collected data from the relevant process and group counted the number of connections pointing to various organizations and countries. Furthermore, we have investigated any other programs or processes which have started up or opened connections at the same time, including system processes.

Overall results Tables 18-20 show the statistics we have calculated for each test program over the three experiments. We have intentionally chosen to test a browser in order to have an idea of the network activity of a program which connects to the internet and is exposed to online advertising - thus our data confirms that on all operating systems Chrome manifested a large number of connections. We were also interested to see if the new Microsoft Edge is in any way more private with the user. The results only show it similarly intrusive as Chrome.

We have noticed an exponential increase in the number of TCP connections opening up for a given program proportional with how recent is the version of Windows. We have observed that Word and Excel tend to open a similar number of connections, while Outlook slightly outnumbers them. But the most interesting results in our opinion come from the insights we have gained about processes running in the background, especially on Windows 10.

At the same time as collecting statistics, we have also noted down the countries and organizations targeted by each process, as well as other processes which have opened connections at the same time with the running test program. We do not include the results for Chrome. As one can see in the tables, 600 or more connections in 5 minutes of browsing are overwhelming, and there is plenty of research work dedicated to browser tracking [4].

Windows 10 Due to the very large number of connections detected on a short period of time, we felt it was required to produce a per-country and per-organization survey of our results on Windows 10. They can be previewed in Table 21. Our research confirms that a large majority of data is indeed sent to Microsoft, but we have also noticed that many connections to Amazon and IPs leased by Akamai are created even with the browser not running. We have also observed that most data is routed through Ireland - whenever a program such as an Office program is started, the initial connections open towards Microsoft Dublin, then a in a few seconds more to the US, the Netherlands, Hong-Kong, Finland, etc.

Unlike in Windows 7 and 8.1, where the"system" test which only involved file browsing and

Program	Word	Excel	Outlook	Chrome
CONNECTION STATISTICS				
Total connections	4	6	68	958
Total hosts	3	3	9	250
Total connections (process)	2	4	20	760
Total connections (others)	2	2	48	198
Total hosts (process)	1	1	4	
Total hosts (other)	2	2	5	92
Closed connections on exit (process)	2	2	16	
COUNTRY STATISTICS				
Total	2	2	5	
Total (process)	2	2	4	
Total (other)	1	1	1	
ORGANIZATION STATISTICS				
Total	1	1	1	
Total (process)	1	1	1	
Total (other)	1	1	1	

Table 18: Windows 7: Quantitative results

search did not trigger anything relevant, a surprising amount of connections open up from Windows 10 when performing the same tasks. Thus we have also focused on the system tests when producing country and organization statistics.

Another observation worth making is that unlike in Windows 8.1 and 7, connections take much longer to close. We have waited for approximately 2 to 3 minutes after we have stopped using the program to test how long it takes connections to close. Around at least 10 connections have been left open by each of Word, Excel and Outlook after program exit.

Running in the background We have also provided a log of the software which runs in the background as some of the main tasks are run. These of course have largely become an issue with Windows 10. We have recognized some as simply settings synchronisers, while others should have no reason to open connections in that situation. For example, Microsoft OndeDrive has been detected to run at the same time with any Office software on Windows 10, actively opening TCP connections at the program's start and closing them at exit.

Windows 7 only opened connections via the system process `svchost.exe`, mainly one or two connections per test. In Windows 8.1 we notice that also the same service opens more processes, in addition to `BackgroundTaskHost.exe`, `SettingsSyncHost.exe` and the System Idle Process.

The background processes opened by Windows 10 are way more diverse and have been documented in Tables 22 and 23.

Office Click-To-Run has been an interesting case study. In one instance Office Click-To-Run has been caught red-handed sending data not only to Microsoft, but also to a suspect

Program	Word	Excel	Outlook	Click-to-run	Chrome
CONNECTION STATISTICS					
Total connections	59	41	76	10	837
Total hosts	14	14	17	7	243
Total connections (process)	13	16	20	4	615
Total connections (others)	46	25	56	5	222
Total hosts (process)	5	7	6	3	226
Total hosts (other)	14	9	13	4	80
Closed connections on exit (process)	8	16	15	4	615
COUNTRY STATISTICS					
Total	6	5	5	4	
Total (process)	4	5	4	3	
Total (other)	5	5	5	3	
ORGANIZATION STATISTICS					
Total	2	2	2	2	
Total (process)	1	2	1	1	
Total (other)	2	2	2	2	

Table 19: Windows 8.1: Quantitative results

IP range some geolocation providers have identified as belonging to the US Department of Defence. We are aware that this may be a false positive as we acknowledge our geolocation framework is not extremely accurate. At the same time, it is a suspect result and we were unable to reproduce it since.

6.3.1 Limitations

The main limitations of this experiment are brought by the geolocation framework. A thorough investigation should survey in depth every IP address identified, which unfortunately in our current state we were unable to do. For example, many IP adresses owned by Akamai Technologies provide owner information in WHOIS databases but not even continent information in geolocation. Other IP addresses refuse ping probes, thus cannot be detected by the simplest latency-based geolocation methods.

Because of these reasons, we are aware this research might include a large number of false positives or true negatives, but we hope this provides a good start on the field of researching privacy leaks on Windows and will inspire future work on the matter.

A possible extension for this project would be repeating these experiments with an improved geolocation framework.

Program	Word	Excel	Outlook	Click-to-run	Chrome
CONNECTION STATISTICS					
Total connections	213	90	117	96	1129
Total hosts	52	37	35	34	264
Total connections (process)	79	43	45	16	799
Total connections (others)	134	47	72	80	330
Total hosts (process)	12	10	11	12	251
Total hosts (other)	47	27	33	26	189
Closed connections on exit (process)	63	10	36	11	799
COUNTRY STATISTICS					
Total	7	3	6	6	
Total (process)	4	3	4	4	
Total (other)	6	3	6	6	
ORGANIZATION STATISTICS					
Total	6	5	6	4	
Total (process)	2	2	1	2	
Total (other)	5	5	6	4	

Table 20: Windows 10: Quantitative results.

6.3.2 Discussion

From our evaluation we can confirm that the legacy Windows 7 is the most private operating system. When evaluating each offline software product only a very small number of connections have started. This is yet a concerning proof that Windows 7 is no longer entirely private, as some users were hoping.

Windows 8.1 has placed itself second, opening a few dozen connections for each program we have tested. With 8.1 we have started to notice that background tasks and the SystemIdleProcess have become extremely active in their networking activity, targeting Microsoft's or various advertising end-points.

Windows 10 has confirmed our worries as well as many user's complaints. The number of open connections is very large, and the overhead on the network is nearly damaging the user's experience of Windows. A big number of Windows Store applications, and applications such as Photos, Search, Cortana and Skype seem to open a even larger number of connections than the foreground programs themselves. This has provided further ideas about features to be added to SnowWall, e.g. a start-up program manager and a means to remove unwanted Windows Store apps.

Application	Microsoft	Verizon	Akamai	CloudFlare	Skype	Amazon
Click-to-run	42	0	49	0	2	0
Outlook	91	2	22	2	1	3
Word	62	4	120	4	2	7
Excel	33	2	53	2	1	0
System	15	2	24	2	1	0

Application	US	IE	NL	SG	HK	IS	FI	AT	LU
Click-to-run	61	9	10	3	4	0	0	0	1
Outlook	61	12	19	0	0	0	27	1	1
Word	63	14	35	5	1	0	3	0	2
Excel	31	6	16	5	1	0	3	0	1
System	24	5	10	0	0	0	0	0	1

Table 21: Windows 10: a survey of the target organization and location of connections

Excel	
SkypeHost (WindowsApps)	3
OneNote(WindowsApps)	6
WinStore (WindowsApps)	12
Cortana Search (WindowsApps)	5
System Idle Process	8
backgroundTaskHost	1
svchost	9
OneDrive	1
HxRsr(WindowsApps)	3

Word	
SkypeHost (WindowsApps)	3
WinStore (WindowsApps)	12
Cortana Search (WindowsApps)	6
System Idle Process	54
svchost	13
OneDrive	1
MicrosoftPhotos(WindowsApps)	7

Table 22: The number of connections opened by background processes when running the Excel test (top) and Word test (bottom)

Outlook	
SkypeHost (WindowsApps)	3
OneNote(WindowsApps)	6
WinStore (WindowsApps)	12
Cortana Search (WindowsApps)	4
System Idle Process	30
backgroundTaskHost	1
svchost	10
OneDrive	1
searchProtocolHost	6

Chrome	
SkypeHost (WindowsApps)	3
OneNote(WindowsApps)	6
WinStore (WindowsApps)	12
Cortana Search (WindowsApps)	3
System Idle Process	347
backgroundTaskHost	1
svchost	11
OneDrive	1

Table 23: The number of connections opened by background processes when running the Outlook test (top) and Chrome test (bottom)

7 Conclusions and future work

7.1 Conclusions

This project has set out to produce a tool which would allow a friendly and efficient interface for monitoring network connections and allowing the users to a more fine-grained control over their privacy. We have created SnowWall, an application compatible on Windows 7 and above capable of performing this task. This application has a modular back-end which queries the system TCP/IP driver API for active connections, interfaces with the Windows Firewall, adds logging handlers to process start and stop events, and provides RESTful access and minor data processing to existing geolocation services. SnowWall also provides monitoring in the SQL format, which is scalable, flexible, and allows proficient data analysis of the logs. The front end has been designed with the help of user-centered design research. We believe this choice has allowed us to create a more versatile and simple front-end which can motivate users to active use.

We have evaluated this program with the help of user testing, as well as qualitative analysis of the geolocation frameworks we use. We have also tested the program in a corporate environment for performing a security analysis and we have observed that SnowWall has been very useful in inspecting minor data breaches. While we are aware that SnowWall is neither an intrusion detection system or an anti-virus tool, the fact that it contains all its features in one place allows it to greatly improve the operating procedure of network analysis. Thus we believe the product we have created is disruptive and useful.

Finally, we have used SnowWall in a range of case studies, which has allowed us to inspect the current state of privacy on Windows systems. Given the lack of peer-reviewed literature in this area, we believe that even our small studies are a great contribution to privacy research on Windows. Our results have only underlined what we were suspecting: that a lot of data is being leaked, through Windows, straight to Microsoft, and that this has also become an issue with Windows 7, even if at a very small scale. We have noticed an exponential growth in the number and bandwith of network connections open while the user is not using the Internet, from few dozens of connections in Windows 8.1 to almost a hundred in Windows 10. We have learned that on Windows 8.1 and 10 many of the leaks are performed through processes independent of the one being used, which allows the user the possibility to take simple action by blocking these services or removing unwanted programs. Last, but not least, we have learned that Microsoft collects data in all parts of the world, but mainly Ireland, the US, Hong-Kong, and the Netherlands. And we were surprised to see that connections to advertisers such as Amazon, Facebook, and Google, have still been detected on the system even when the user didn't even start the browser since reboot.

To conclude, we believe we have achieved the main objectives of this project, and we are looking forward to future work which may improve its documented flaws.

7.2 Improvements

We have considered our weak points and below we suggest a list of improvements that could be made to the platform, in order to build the best and most resilient version of SnowWall.

Geolocation Because of the ongoing problem of IP geolocation, SnowWall has been built with a modular design, allowing interchangeable geolocation frameworks to be used, depending on the user's desired granularity, and has been tested with various geolocation frameworks and techniques. We were unable to provide a better solution than the current state-of-the-art, and thus we were forced by time constraints and project priorities to choose available methods and do our best to aggregate them in the best way; we hope that improving the geolocation accuracy will prove crucial in detecting malicious hosts and improving the performance and protection offered by SnowWall.

Data processing We are aware that the processing work required by geolocation could be done asynchronously on a server e.g. with cron jobs. This would allow us to better compare the geolocation data we receive from various services and protocols and also to query other services which do not provide a RESTful API.

Program manager Currently SnowWall provides no features allowing it to uninstall programs. We believe this could be a good addition to the Program View allowing the user to quickly remove software which they do not use but has made itself remarked because of opening network connections.

Front-end Choosing Windows Forms as a front-end development framework was to allow a proof-of-concept of the application. We are aware that with more time invested into this project, a better front-end could be built with the Windows Presentation Foundation, which would have allowed more versatile data binding, animations, font scaling, and multiple other controls, or even a Universal Windows Platform application which would seamlessly integrate with touch on tablets.

7.3 Future work

Building a geolocation framework As we have previously stated in both the Background section, in Evaluation and in the Case Studies, we believe that the quality of SnowWall as a research tool would be much improved by providing it with an adaptive and on-target geolocation framework, as the current available free solutions, as well as the most popular paid solution, MaxMind, are notoriously full of errors. A possible continuation of this project would be applying the research and current state-of-the-art and produce a new such framework. We have envisioned a latency-based framework which makes use

of mined data, bordering between the two methods but concentrating on scale. Such a method would require large datasets of landmarks as well as great networking resoruces.

Packet Filtering A way to greater improve the analysis and study of leaked data, a means of packet filtering could be implemented into SnowWall. We have not concentrated on this feature throughout this project as we have noticed the data which is send over to Microsoft is generally encrypted, but at the same time given the existing toolset provided by SnowWall we believe that minimal packet filtering features would provide more insight into the connections.

Separating between inbound and outbound connections Currently SnowWall cannot differentiate between inbound and outbound connections as we were unable to find an API or any other means to retrieve this information. Currently we perform a simple filtering based on connection state, but we are aware that displaying both inbound and outbound connections on the Map View may be misleading to the user who is only allowed to block outbound connections in that view. We propose as a future extension evaluating the bandwidth of each connection in real-time, and by the number of packets sent and received inferring whether it is inbound or outbound.

Context menu for blocking programs It has been suggested in our user design research that we add a context menu in the Windows Explorer which would allow right-clicking a program's executable and automatically blocking it to the Windows Firewall.

History visualization tool It would provide great insights and a even more convincing visualisation if users were able to preview a history of connections, for example by dragging a slider under the map and being offered a visual log of results.

7.4 Legal concerns

7.4.1 Geolocation

It is still an open question how do legal frameworks position themselves with respect to IP geolocation, since IP geolocation can be used both as a tool to protect privacy or to invade it.

In the United Kingdom, IP geolocation is limited by the Data Protection Act to only yield the physical address of the ISP. Any further tracking (e.g. for criminal tracing) has to be carried out by getting the ISP to check their logs.

On the other hand, many online enterprises are now subject to strict laws imposed in the US and Europe, such as the USA PATRIOT Act, the Bank Secrecy Act, the US Treasury Department's Office of Foreign Assets Control etc., which require identification of the online

visitors. By identifying where online visitors really are, geolocation can be used to protect institutions from fraud, and individuals from cyber-stalking, identity theft etc, as well as help in criminal investigation and research in computer security.

7.4.2 Privacy

A data survey performed by the Data Sovereignity group[73] has found that 92% of IP professionals agree with the idea that user's data should not leave the country. We have noticed throughout our research that this is not at all the case with Microsoft Windows. Storing data outside the country it it used involves many risks, not only privacy, but also consistency and integrity.

In 2016, the European Union has introduced a new Data Protection Regulation[74] which is mostly concerned with the security of IT systems and states that standards should be implemented in all businesses which collect or use data. Currently a business can be fined in the UK at most 400,000 pounds in case of negligence in a data breach. Given the Vault7 leaks, and the colossal number of records lost or leaked throughout the years, the new regulation is much tougher on businesses and the fines they need to pay if they are found to have carelessly handled user's data. We hope that these regulations will inspire businesses to apply security standards more thoroughly, as well as incite both academic and corporate research into more proficient anonymisation techniques.

Acknowledgements

I would like to thank my supervisor, Dr. William Knottenbelt, for his evergreen enthusiasm, his support, the brilliant time we had together working on this project, and all the adjacent opportunities. Prof. Susan Eisenbach, for the patient and masterful care she took of my well-being throughout these years. Dr. Naranker Dulay, for his help with my research, and for teaching me to wear my tin foil hat with style. Dr. Evangelos Georgiou, my industrial placement manager, for everything I've learned about development in the .NET framework. The wonderful team at NetCraft, who have provided some of the geolocation datasets used in this project. Prof. Mihaela Visinoiu and Prof. Dorin Arventiev, my high-school computing and maths teachers, for nurturing the scientist in me. Stefan Zaharia, for his help with the proof-reading of this thesis. Andrei Pacuraru, for his boundless support. Last, but not least, my family and friends, for putting up with my madness.

8 References

[1] Martin, S., Rainie, L., Madden, M. *Americans' Privacy Strategies Post-Snowden*, Pew Research Center, March 2015.

[2] *Vault 7.* WikiLeaks research community, 24 Apr 2017. Avaiable at: `https://our.wikileaks.org/Vault_7`

[3] Rahulamathavan, Y., Moonsamy, V., Batten, L., Shunliang, S., and Rajarajan, M. *An Analysis of Tracking Settings in Blackberry 10 and Windows Phone 8 Smartphones.* ACISP 2014, LNCS 8544, pp. 430–437, 2014

[4] Englehardt, S., Narayanan, A. *Online Tracking: A 1-million-site Measurement and Analysis*, ACM CCS 2016

[5] Arp, D., Quiring, E., Wressnegger, C., Rieck, K. *Privacy Threats through Ultrasonic Side Channels on Mobile Devices.* Available at: `https://www.sec.cs.tu-bs.de/pubs/2017a-eurosp.pdf`

[6] Järvinen, H. *Microsoft's new small print – how your personal data is (ab)used*, European Digital Rights organization (EDRi), 17 Jun 2015. Available at: `https://edri.org/microsofts-new-small-print-how-your-personal-data-abused/`

[7] Keizer, G. *Windows 10 makes diagnostic data collection compulsory*, Computer World, 10 Aug 2015. Available at: `http://www.computerworld.com/article/2968288/microsoft-windows/windows-10-makes-diagnostic-data-collection-compulsory.html`

[8] *General privacy settings in Windows 10*, Microsoft Website. Available at: `https://privacy.microsoft.com/en-US/general-privacy-settings-in-windows-10`

[9] *October 2016 Preview of Monthly Quality Rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1*, Microsoft Support Website, 8 Nov 2016. Available at: `https://support.microsoft.com/en-us/help/3192403/october-2016-preview-of-monthly-quality-rollup-for-windows-7-sp1-and-windows-server-2008-r2-sp1`

[10] *October 2016 Preview of Monthly Quality Rollup for Windows 8.1 and Windows Server 2012 R2*, Microsoft Support Website, 8 Nov 2016. Available at: `https://support.microsoft.com/en-us/help/3192404/october-2016-preview-of-monthly-quality-rollup-for-windows-8.1-and-windows-server-2012-r2`

[11] Mughees, M. H., Zhiyun Qian, Z., Shafiq, Z., Dash, K., Hui, P. *A First Look at Ad-block Detection – A New Arms Race on the Web.* arXiv:1605.05841

[12] Brinkmann, M. *Microsoft intensifies data collection on Windows 7 and 8 systems.* Ghacks, 28 Aug 2015. Available at: https://www.ghacks.net/2015/08/28/microsoft-intensifies-data-collection-on-windows-7-and-8-systems/

[13] *Microsoft Customer Experience Improvement Program*, Microsoft.com. Available at: https://www.microsoft.com/products/ceip

[14] *List of Windows 10 tracking / telemetry / ads hosts?*, GitHub.com. https://github.com/StevenBlack/hosts/issues/155

[15] Brinkmann, M. *Comparison of Windows 10 Privacy tools.* Ghacks.net, 14 Aug 2015. Available at: https://www.ghacks.net/2015/08/14/comparison-of-windows-10-privacy-tools/

[16] *Microsoft Privacy Statement.* Microsoft.com, Mar 2017. Available at: https://privacy.microsoft.com/en-us/privacystatement

[17] *ISO/IEC standard 7498-1:1994.* Available at: http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip

[18] Clark, D. *The Design Philosophy of the DARPA Internet Protocols.* SIGCOMM '88 Symposium proceedings on Communications architectures and protocols. ACM: 106–114. doi:10.1145/52324.52336. ISBN 0897912799.

[19] *RFC 1122, Requirements for Internet Hosts – Communication Layers*, 1.1.3 Internet Protocol Suite, 1989. Available at: https://tools.ietf.org/html/rfc1122#page-9

[20] *Root DNS Servers Website.* Available at: http://root-servers.org/

[21] *WHOIS Protocol Specification.* https://tools.ietf.org/html/rfc3912

[22] Wright, G. R., Stevens, W. R. *TCP/IP Illustrated, Volume 2: The Implementation.* Addison-Wesley Longman Publishing Co., Inc. Boston, USA, 1995. ISBN:0-201-63354-X

[23] Ingham, K., Forrest, S. *A History and Survey of Network Firewalls*

[24] *United States vs. Morris, No. 774, Docket 90-1336.* United States Court of Appeals, Second Circuit. Argued Dec. 4, 1990. Decided March 7, 1991.

[25] Wack, J., Cutler, K., Pole, J. *Guidelines on Firewalls and Firewall Policy.* Recommendations of the National Institute of Standards and Technology (NIST), NIST Special Publication 800-41, January 2002. Available at: http://www.usmd.edu/usm/adminfinance/itcc/firewallpolicynis.pdf

[26] Page, B. *A report on the Internet Worm.* Available at: http://www.ee.ryerson.ca/~elf/hack/iworm.html

[27] Mogul, J. *Using screend to implement IP/TCP security policies*. Tech. Rep. NSL Technical Note TN-2, Digital Network Systems Laboratory, Palo Alto, CA. 1991.

[28] Ranum, M. J. *A network firewall. In Proceedings of the First World Conference on System Administration and Security*, 1992. SANS Institute.

[29] Chapman, D. B. *Network (in)security through IP packet filtering*. In UNIX Security Symposium III Proceedings, 14-16 Sept 1992, Baltimore, MD, USA. USENIX Association, Berkeley, CA, 63–76. Available at: `http://www.greatcircle.com/pkt_filtering.html`

[30] Julkunen, H., Chow, C. *Enhance network security with dynamic packet filter*. 1998. Tech. Rep. EAS-CS-97-2, University of Colorado at Colorado Springs Department of Computer Science

[31] *VPNs and Firewalls*. Microsoft TechNet website. Available at: `https://technet.microsoft.com/en-us/library/cc958037.aspx`

[32] Avolio, F. M., Ranum, M. J. *A network perimeter with secure external access*. In Internet Society Symposium on Network and Distributed Systems Security, 3-4 Feb. 1994, San Diego, CA, USA. Internet Society, Reston, VA, USA, 109–119.

[33] Pescatore, J., Young, G. *Defining the Next-Generation Firewall*, Gartner RAS Core Research Note G00171540, 12 October 2009, R3210 04102010.

[34] Chapman, D. B., Zwicky, E. D. *Building internet firewalls*. ISBN 1-56592-124-0. First Edition, November 1995

[35] Bailey, M., Cooke, E., Jahanian, F., Watson, D. *The Blaster Worm: Then and Now*, IEEE Security Privacy, 1540-7993/05, 2005. Available at: `nsrg.ece.illinois.edu/publications/IEEE_Security_Privacy_Blaster_Final.pdf`

[36] Syed, F. *Understanding Worms, Their Behaviour and Containing Them*. Available at: `http://www.cse.wustl.edu/~jain/cse571-09/ftp/worms/index.html#sasser`

[37] *SANS Survival Time*. Available at: `https://isc.sans.edu/diary/Survival+Time+on+the+Internet/4721`

[38] Horowitz, M. *The Windows firewall is the overlooked defense against WannaCry and Adylkuzz*, Computerworld, 12 May 2017. Available at: `http://www.computerworld.com/article/3197421/networking/the-windows-firewall-is-the-overlooked-defense-against-wannacry-and-adylkuzz.html`

[39] *PLATINUM: Targeted attacks in South and Southeast Asia*, Windows Defender Advanced Threat Hunting Team, April 2016. Available at: `https://blogs.technet.microsoft.com/mmpc/2017/06/07/platinum-continues-to-evolve-find-ways-to-maintain-invisibility/`

[40] *How Windows Firewall Works.* Microsoft TechNet website. Available at: `https://technet.microsoft.com/en-us/library/cc755604(v=ws.10).aspx`

[41] *Windows Firewall with Advanced Security*, Windows IT Center, Available at: `https://docs.microsoft.com/en-us/windows/access-protection/windows-firewall/windows-firewall-with-advanced-security`

[42] Hill, K. *How an internet mapping glitch turned a random Kansas farm into a digital hell.* Fusion. 2016-04-10. Available at: `http://fusion.net/story/287592/internet-mapping-glitch-kansas-farm/`

[43] *Automatic identification and data capture (AIDC) techniques.* ISO/IEC 19762:2016 Available at: `http://www.iso.org/iso/catalogue_detail.htm?csnumber=61301`

[44] Padmanabhan, V. N., Subramanian, L. *An investigation of geographic mapping techniques for internet hosts*, in Proceedings of ACM SIGCOMM Conference, San Diego, CA, August 2001.

[45] Freedman, M., Vutukuru, M., Feamster, N., Balakrishnan, H. *Geographic Locality of IP Prefixes*, Internet Measurement Conference (IMC), Berkeley, CA, 2005. Available at: `http://nms.csail.mit.edu/papers/ipgeo-imc05.pdf` [Accessed 19 Jan 2017]

[46] Zhang, M., Ruan, Y., Pai, V., Rexford, J. *How DNS misnaming distorts Internet topology mapping*, USENIX Annual Technical Conference, Boston, MA, June 2006. Available at: `https://www.usenix.org/legacy/event/usenix06/tech/full_papers/zhang/zhang_html` [Accessed 22 Jan 2017]

[47] Davis, C., Vixie, P., Goodwin, T., Dickinson, I. *A means for expressing location information in the domain name system.* RFC 1876 (1996).

[48] Moore, D., Periakaruppan, R., Donohoe, J., Claffy, K. *Where in the world is netgeo.caida.org?*, INET, 2000.

[49] Krishnamurthy, B., Wang, J., *On Network Aware Clustering of Web Clients.* ACM SIGCOMM, August 2000

[50] Guo, C., Liu, Y., Shan, W., Wang, H. J., Yu, Q., Zhang, Y. *Mining the web and the internet for accurate ip address geolocations.* In Infocom mini conference, '09.

[51] Gueye, B., Ziviani, A., Crovella, M., Fdida, S., *Constraint-based geolocation of internet hosts*, in IEEE/ACM Transactions on Networking, December 2006.

[52] Enge, P., Misra, P., *Special issue on global positioning system.* Proceedings of the IEEE, 87(1):3–15, January 1999.

[53] Katzbassett, E., John, J. P., Krishnamurthy, A., Wetherall, D., Anderson, T., Yatin. *Towards IP geolocation using delay and topology measurements*, in IMC '06 Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, Pages 71-84

[54] Percacci, R., Vespignani, A. *Scale-free behavior of the internet global performance.* The European Physical Journal B - Condensed Matter (2003).

[55] Vincenty, T.*Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations*, Survey Review (1975).

[56] Eriksson, B., Barford, P., Maggs, B., and Nowak, R., *Posit: An Adaptive Framework for Lightweight IP Geolocation*, in Sigmetrics Performance Evaluation Review, Volume 40, Issue 2, Pages 2-11, September 2012.

[57] Wong, B., Stoyanov, I., Sirer, E., *Octant: A comprehensive framework for the geolocation of internet hosts*, in USENIX NSDI Conference, Cambridge, MA, April 2007.

[58] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, *Towards Street-Level Client-Independent IP Geolocation*, in Proceedings of USENIX NSDI 2011, vol. 5, no. 5, Boston, MA, March 2011.

[59] Laki, S., Matray, P., Haga, P., Csabai, I., Vattay, G., *A detailed path-latency model for router geolocation*, 2009 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks Communities and Workshops, Washington, DC, 2009, pp. 1-6. doi:10.1109/TRIDENTCOM.2009.4976258

[60] Eriksson, B., Barford, P., Sommers, J., and Nowak, R., *A Learning-based Approach for IP Geolocation*, in Proceedings of Passive and Active Measurements Conference, Zurich, Switzerland, April 2010.

[61] Arif, M., Karunasekera, S., Kulkarni, S., Gunatilaka, A., Ristic, B., *Internet Host Geolocation Using Maximum Likelihood Estimation Techniques*, in Proceedings of IEEE International Conference on Advanced Information Networking and Applications (AINA), Perth, Australia, April 2010.

[62] Youn, I., Mark, B., Richards, D., *Statistical Geolocation of Internet Hosts*, in Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN), San Francisco, CA, August 2009.

[63] B. Eriksson, P. Barford, and R. Nowak, *Network Discovery from Passive Measurements*, in Proceedings of ACM SIGCOMM Conference, Seattle, Washington, August 2008.

[64] C. Jin, H. Wang, and K. Shin, *Hop-Count Filtering: An Effective Defense Against Spoofed Traffic.* in Proceedings of IEEE INFOCOM Conference, San Francisco, CA, April 2003.

[65] Hu, Z., Heidemann, J., Pratkin, Y., *Towards Geolocation of Millions of IP addresses*, USC/ISI Technical Report ISI-TR-680, May 2012.

[66] *GEANT network.* http://www.geant2.net/

[67] D. Morato et al., *ETOMIC: A testbed for universal active and passive measurements*, Proceedings of IEEE TRIDENTCOM 2005, Best Testbed Award, p283-289, 23-25 February 2005, Trento, Italy (2005)

[68] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, *PlanetLab: An Overlay Testbed for Broad-Coverage Services*, SIGCOMM CCR, vol. 33, no. 3, pp. 3–12, 2003.

[69] *Standard ECMA-334 C Language Specification*, ECMA, 2006. Available at: `https://www.ecma-international.org/publications/standards/Ecma-334.htm`

[70] *Information technology - Programming languages - C*. ISO/IEC 23270:2006. Last reviewed 2012. Available at: `https://www.iso.org/standard/42926.html`

[71] *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*, ISO 9241-210:2010.

[72] *Desktop Operating System Market Share* Available at: `https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0`

[73] Woodward, J. *Data Sovereignty Survey - 92% of IT Security Professionals say data should remain in UK*, CNS, 29 Iun 2016. Available at: `https://www.cnsgroup.co.uk/media-hub/news/news-article/2016/06/29/cns-group-data-sovereignty-survey-reveals-92-percent-of-it-security-professionals-believe-data-should-remain-in-the-uk`

[74] *Overview of the General Data Protection Regulation (GDPR)*, Information Commissioner's Office, May 2017. Available at: `https://ico.org.uk/for-organisations/data-protection-reform/overview-of-the-gdpr/`