Supplemental Authentication via Internet

Fingerprinting

by

Ryan Tate

Department of Computer Science
Duke University

Date: ______________________

Approved:

__________________________
Xiaowei Yang, Supervisor

__________________________
Jeffrey Chase

__________________________
Landon Cox

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science in the Graduate School of Duke University 2011
Supplemental Authentication via Internet Fingerprinting

by

Ryan Tate

Department of Computer Science
Duke University

Date: ________________

Approved:

________________________
Xiaowei Yang, Supervisor

________________________
Jeffrey Chase

________________________
Landon Cox

An abstract of a thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science in the Graduate School of Duke University 2011
Abstract

Internet websites are a regular medium for exchanging sensitive information such as online banking. The security of this information is paramount. Today, one facet of this security - authenticating a website to its users - depends on the trust of a third party (i.e., a certificate authority). However, web browsers currently trust many certificate authorities from around the world. Some of them may be compromised or untrustworthy. This work explores an authentication scheme that does not require trust but instead uses unexploited network characteristics of a website to authenticate the website to users. Our preliminary evaluation shows that this scheme can reject all of over 200,000 verified online phishing website visits while recognizing more than 99% of the 7,000 legitimate websites over the course of a week. Results suggest that network characteristics can provide a supplemental website authentication scheme. It has no noticeable overhead or network footprint and is independent of any third party trust.
## Contents

Abstract ........................................ iv

List of Tables .................................. vii

List of Figures .................................. viii

1 Introduction .................................. 1

2 Overview ..................................... 6

  2.1 Intuition ................................... 6

  2.2 Model ...................................... 7

  2.3 System design ............................... 8

    2.3.1 Limitations ............................. 8

    2.3.2 Constraints and assumptions .......... 9

3 Design ........................................ 11

  3.1 Fingerprinting ............................. 11

    3.1.1 Network layer ........................ 12

    3.1.2 Transport layer ....................... 15

    3.1.3 Application layer ..................... 16

    3.1.4 Environmental factors ................ 17

  3.2 Profiling .................................. 20

    3.2.1 Profile mappings ...................... 20

    3.2.2 Profile windows ....................... 21
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Design Constraints and Assumptions.</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Features and mapping functions.</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Authentication schemes.</td>
<td>23</td>
</tr>
<tr>
<td>3.3</td>
<td>Notation definitions.</td>
<td>23</td>
</tr>
<tr>
<td>3.4</td>
<td>Feature confidence rates.</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Evaluated authentication schemes.</td>
<td>29</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Legitimate website. .................................................. 3
1.2 Phishing website. .................................................... 3
2.1 System design. ....................................................... 9
3.1 Response time depiction. .......................................... 19
3.2 Box plot. .............................................................. 20
3.3 Confidence rating example. ........................................ 24
4.1 Acceptance rates of authentication schemes. ................. 28
4.2 Expected distinction. ................................................ 30
4.3 CDF of confidence ratings. ......................................... 31
4.4 Rejection Rate vs Acceptance Rate. ............................ 32
4.5 Feature expected distinction. ...................................... 33
A.1 IP header. .......................................................... 44
A.2 TCP header. ........................................................ 45
B.1 Comparison of reward and confidence acceptance rates. ... 49
B.2 CDF of reward and confidence ratings. ......................... 49
The Internet is ubiquitous in modern society and websites are not only an expected service of commerce, government, and other organizations, but an essential and sometimes exclusive means of interaction. Many websites have become a medium for sensitive (proprietary, confidential, classified, etc.) information exchange, like online banking. Protecting this information requires reliable security measures and the complexity of the technology has resulted in a multi-faceted approach. One such facet is authentication - in which two entities verify their identities (typically mutually) before exchanging information. Authentication prevents disclosure of sensitive information to untrusted entities. There is a general satisfaction with website authentication (website to user), but current practice makes many critical assumptions. The most critical such assumption is the trust of a third party to protect sensitive information.

Internet website visitors (via web browsers) depend on Internet infrastructures and protocols to ensure the release of sensitive information to only trusted entities. The Internet’s routing infrastructure directs communication to physical machines associated with each Internet Protocol (IP) address returned from the Domain Name
System (DNS). Section A.1 reviews background information on the IP - the primary protocol of the Internet. DNS translates website hostnames (such as google.com) to IP addresses.¹ DNS is an essential service taken for granted as much as IP in authentication. Users do not trust the Internet to meet security needs and so protocols like Secure Sockets Layer (SSL) and a Public Key Infrastructure (PKI) have emerged to provide security within these networks. An essential part of authentication, PKI certificate authorities such as Verisign (Verisign, 2011) provide digital certificates that verify a website is truly the one it claims to be and help ensure the release of information to only that entity.

DNS, the Internet infrastructure, and its PKI are not perfectly secure. Subversion of a certificate authority can allow malicious entities to masquerade as a legitimate website to gain access to sensitive information while fooling web browsers. Further, certificates expire and must be replaced and algorithms and industry standards change. Alternately, IP hijacking and DNS cache poisoning ² can direct Internet traffic to unintended recipients. Wireless communications broadcast data to any machine within range and simple errors or bugs can also result in traffic reaching unintended targets. Internet users must inherently trust third parties - certificate authorities and the Internet’s collection of diverse autonomous systems - to protect sensitive information and currently have no way to verify the authenticity of websites independent of this trust. Finally, authentication is entirely a website (host) choice: users cannot decide on the certificate authority they must trust or require websites to use secure protocols at all.

Requiring trust to protect sensitive information is dangerous because there is a significant threat. There are many malicious websites on the Internet that attempt

¹ DNS Security (DNSSec) is a developing Internet concept to provide authenticated URL to IP address translations (IETF, 2005).

² IP hijacking and DNS cache poisoning are computer network attacks that alter the behavior of Internet routers and the IP address translation of Internet hostnames, respectively.
to circumvent current Internet security measures. It is easy to make a website look like and feel like another - fooling the user who does not heed browser warnings. Figure 1.2 shows an example of an impostor website from (PhishTank, 2011), an anti-phishing service\(^3\), compared with its legitimate counterpart in Figure 1.1. SecurityWeek, an Internet news service, reported that cyber-criminals are creating over 57,000 fake websites each week in attempt to imitate and exploit websites from 375 high-profile brands like Western Union and that over 65% of global Internet users have been victims of a cybercrime (SecurityWeekNews, 2010).

\(^3\) A phishing attack is one in which an adversary attempts to fraudulently steal personal information.
Successfully masquerading as a legitimate website and fooling the browser is more
difficult than simply imitating a website, but not impossible. This effort depends
on obtaining a certificate that a web browser trusts in addition to directing traffic
to the malicious website (such as DNS cache poisoning or IP hijacking). Certificate
subversion is accomplished either via the cooperation of a certificate authority or
the generation of one of their certificates. The Firefox (Mozilla, 2011) web browser
currently trusts over 100 different certificate companies from around the world. Re-
cently, Comodo, a globally trusted certificate authority, acknowledged that a hacker
had compromised nine of its certificates (Hallam-Baker, 2011). There are reports
that some organizations are able to subvert widely used encryption algorithms (Sin-
gel, 2010). Security should extend beyond trust in third parties and fooling a web
browser as to website authenticity should be as difficult as possible.

In contrast with website authentication, many reliable authentication methods
count on first-hand knowledge. For example, bank customers authenticate their bank
based on knowing its location and remembering its appearance. Most customers
would not walk into a similar looking bank down the street and deposit money if
someone told them it was their bank. Online, however, it is the norm for users to
accept website authenticity based on third party advice while ignoring readily avail-
able first-hand information. The only first-hand information checked by users is the
correct spelling of a website URL and the visual appearance of the webpage itself.
Website appearance is easily copied and this paper will assume that users type URLs
correctly. Given the sensitivity of the information at stake, a more robust system of
website authentication checks (in addition to trust, website appearance, and URL
spelling) is desirable. Many different checks are the physical norm (company issued
ID cards, personal relationships, knowledge of password or mother’s maiden name,
background checks, etc. often accompany each other). First-hand, readily avail-
able information consisting of many different types provides an avenue for website
authentication independent of third parties.

This paper describes a system that exploits the first-hand information readily available to Internet users that can authenticate a website without third party trust. Section 3.1 describes the readily available information that is inherent in all website visits on different network levels, but is currently unexploited for authentication purposes. Section 3.3 describes a way to exploit this information in order to provide many different checks that together can authenticate a website to a user in a consistent manner. This paper also explores several performance trade-offs in the design of this system and evaluates them in Section 4. The design does not require certificate authorities or computationally complex operations. It requires no supporting infrastructure or the cooperation of any servers (not even the website of interest), can be deployed one user at a time with minimal overhead, results in no loss of web browser function, and introduces no additional network traffic. These aspects are helpful to maximize usability as described in Section 2. Contrast these benefits with certificate based authentication and note how certificate authorities have created an entire industry upon which the majority of Internet users are now dependent.

An implementation of this design was able to reject all known phishing websites while accepting as legitimate their genuine targets over 99% of the time in a week-long evaluation of 3,154 websites. Section 4 describes the results of the system implementation and an analysis of their implications. The intent of this work is to supplement traditional authentication without requiring third party support - it adds an additional layer not dependent on trust. Section 6 briefly discusses future work that would generalize the evaluated design to many more applications with some (likely) straightforward additional work.
2

Overview

2.1 Intuition

First-hand information is currently available for website authentication within the network layers of an Internet connection between a web browser and a website. The intuition is similar to the customer-bank example in which several different aspects (bank name, location, building appearance, etc.) form a complete picture able to uniquely identify a trusted entity. The aspects of interest in this case are systemic details, protocol variations, and measurements which operating systems and web browsers typically hide from users.

Rather than relying on a single component to definitely prove the identity of a website (ie. a trusted, signed certificate), a set of various observable characteristics' values forms a fingerprint to uniquely identify a website (demonstrated in Section 4). Characteristics come from the network, transport, and application layers and reflect environmental factors. Even if each characteristic may be easily emulated, by drawing from many different characteristics, a great deal of effort would be required to identify and emulate all characteristics of a website session consistently with each user’s observations. Section 3.1 elaborates on the difficulty of emulating
(intentionally or incidentally) characteristics between different Internet websites.

The novelty compared with other signature-based works as those in Section 5.2 is that this approach incorporates characteristics from all network layers as well as environmental and user-specific factors that make each set of website characteristics unique to each user (such as network path latency). User-specific characteristics are distinguishing because they are a function of not just the website and its server protocol implementations, but the specific path and other aspects of a website session which vary between users based on location. Compounding the situation is the prevalence of content distribution networks, replicated servers, and web hosting services which provide the same website content from different servers located in different geographical regions. Not all servers providing the same website content will have the same underlying protocol implementations or characteristics for all users. The result is not just a handful of characteristics an adversary must emulate to successfully masquerade as a certain website, but a number of characteristics that are dependent on server and client locations for each intended victim.

2.2 Model

For simplicity, this paper assumes the following threat and client-server model. The system safeguards against an adversary attempting to obtain sensitive information through a phishing (impostor) website designed to fool users and web browsers into believing it is some website of interest to the user (such as facebook.com). Determination is according to the hostname URL that the user types (assumed correctly) into the browser. The Internet traffic is somehow (DNS cache poisoning or IP hijacking) directed to the phishing site rather than the legitimate, intended site. The model assumes that the user (client) accesses the Internet from the same machine using the same web browser via the same access location and method (ie. ISP) consistently. The system will be located on the client machine and will be allowed no cooperation
(it will act passively) with external network entities. Finally, the design assumes that an impostor website is hosted on a different server as that hosting the legitimate site. The intent is to provide authentication supplementing digital certificates such as in the event of the subversion or mistrust of a certificate authority or if a website does not use a secure protocol.

2.3 System design

This system essentially captures a set of Internet characteristics from legitimate website visits (sessions) for each client in the form of a fingerprint, which is a vector of some mapping of observed characteristic(s) described further in Section 3.1 and called features. For simplicity, the design assumes independence between features (and selects them accordingly as described in Section 3.1). Through one or more fingerprints, it will learn profile values for each website such that the resulting profile is able to describe the range of expected (previously observed or acceptable) feature values for a legitimate fingerprint. Authentication of future fingerprints against a website’s profile requires the matching of each feature to the profile values. Figure 2.1 graphically depicts the system design.

2.3.1 Limitations

There is a bootstrapping assumption necessary to begin a profile where no fingerprints exist. It is possible to assume that the first website session is authentic - corroborating it through some other means (certificate-based authentication). A slightly different approach is to take the profile from a trusted individual. In any case, the system can only verify the authenticity of a previously visited website. This is the same technique used in SSH authentication (IETF, 2006) - the first exchange forms the basis upon which future exchanges will be compared. The system does not prove or evaluate all characteristics separately or find an optimal method of profiling
and authenticating. It only demonstrates the utility of the approach and explores some trade-offs.

2.3.2 Constraints and assumptions

A website session is a complicated exchange of data dependent on many dynamic factors which make authentication difficult. Features of one session are not guaranteed to exactly match even subsequent sessions between the same client and website. Session characteristics may change over time because of content or protocol updates or environmental variables. Other sites may happen to exhibit the same characteristics. Also, features are simply subject to noise. Client sessions with a particular website may be rare (small sample size) or a client may never visit a malicious site (only positive examples available) based on each client’s behavior. This is particularly constraining because problems of this type (binary classification, concept or function learning, and anomaly detection as surveyed in Section 5) generally improve with training sample size. Finally, goals for practical use further constrain the design. Table 2.1 lists the design constraints and assumptions and a brief discussion on their effects and rationale.
Table 2.1: Design Constraints and Assumptions.

1. Features must be consistent and diverse. Makes a profile effective for distinguishing websites while reliably authenticating its originator.

2. Features should be independent. For simplicity and to maintain utility of each feature (eliminate redundancy).

3. Some characteristics may behave unpredictably sometimes. Design assumes permanent and unpredictable changes will affect the minority of features simultaneously, but must adapt to complete profile changes and be robust to imperfect information resulting from unrepresentative (atypical) characteristic observations.

4. System will passively observe characteristics. Necessary to reduce footprint, overhead, and external support. Reduces sample size (number of fingerprints) and availability of characteristic observations.

5. There will be a limited, positive set of fingerprints (data samples) for profiling. This is the result of passive sampling and unknown user behavior patterns. Reduces sample size and eliminates some profiling options (lack of counter examples).

3

Design

3.1 Fingerprinting

Fingerprinting is mapping Internet characteristics of a website session into a fingerprint. The intuition is to capture many features from multiple network layers to include those features unique to a particular client location. Feature values are either continuous or discrete. Though continuous features are essentially random variables, making assumptions such as about statistical distributions will be inherently inaccurate because of Constraints 4, 5. Therefore, mappings minimize sensitivity to outliers and anomalous sampling - such as in using the sample median rather than mean. Discrete values may be inherently more reliable because of their stationarity, but less discriminating if relatively common between web servers. Table 3.1 lists the fingerprint features and their associated mapping functions.

Constraints in Table 2.1 have restricted feature domains to passively observable and independent characteristics that are consistent between client sessions with the same website but diverse between sessions with different websites. The system passively intercepts all traffic available to the client machine, operating system, or web
browser to include link, network, and application (encrypted) communications data (assuming root access). To make Constraint 1 (consistency/diversity) more concrete, each feature must narrow each fingerprint’s set of possible matching profiles such that the intersection of all feature-wise sets is one website profile. Section 4 demonstrates that design features meet this condition. Sections 3.2.2 and 3.3 discuss methods to handle inconsistencies and unpredictable feature values.

Reference Appendix A for a review of relevant web protocols.

3.1.1 Network layer

Characteristics derived from the header of TCP/IP packets, network path property measurements, and DNS queries reveal differences for websites located on different machines (model assumption). It is likely that many paths between a client and any arbitrary number of servers will overlap given the prevalence of content distribution networks and web hosting services. However, features in this layer are likely the most difficult portion of a client-server connection to fake (the network path’s properties) because they are required for successful delivery of packets and are somewhat protected from tampering by their administrative entities for security and performance reasons.

Characteristics of a network path, such as latency and bandwidth, depend on the physical infrastructure and route topology as well as dynamic conditions such as congestion (due to traffic). Emulating network properties to fool characteristics-based authentication requires knowledge of each intended victim’s network path and conditions. For example, it is possible to inject arbitrary additional delay before transmitting a TCP ACK in order to emulate a larger path latency. However, it is not possible to remove arbitrary delay. Latency can distinguish between websites hosted in different geographic regions with a significant difference in delay. For example, latency characteristics may be effective in restricting potential phishing websites to
nearby locations (such as within the jurisdiction of a single law enforcement agency) and therein reduce the threat significantly. Finally, model constraints may eliminate some characteristics like bandwidth because most clients are likely to experience a similar bottleneck in the 'last mile' of the network for all websites.

**IP address** mapping takes the destination address from the header of the IP packet containing the initial SYN packet the client machine sends. Some websites may have multiple addresses due to dynamic address allocation or hosting on multiple servers. However, (Poole and Pai, 2006) has found that the mapping of IP address to hostnames typically persists for at least 30 days. The system assumes that a website will have a small, consistent pool of addresses available to each client. Section 3.3 addresses dynamic IP address allocation. Each must have a unique address unless virtual hosting or Network Address Translation (design ignores this possibility). An adversary cannot simply forge the IP address of his packets because the traffic will not return without some kind of packet re-routing or infrastructure attack - requiring considerable access and capabilities. The role of IP addresses in the working Internet naturally suite it with the constraints of Sections 2.3.2.

**Latency** is the propagation delay of a packet along a network path (and back). Latency mapping observes the Shortest Observed Round-Trip Time (SORTT) - the smallest observed time difference between when a client sends a TCP data packet and receives its ACK. SORTT is agnostic of path asymmetry and requires no clock synchronization or TCP timestamps, but is subject to other problems like retransmission ambiguity, packet length variation and loss, and congestion or network anomalies. Nearly all measurement complications are additive and therefore the SORTT is the most accurate (and common) measurement which mitigates outliers, anomalies, and limited observations. Design relaxes the typical assumption of negligible server ACK
response time to assuming that such a delay is only consistently observed. Likewise, it is not necessary to assume (via a significant number of measurements) that eventually one sample will have negligible queuing delays as is common. The true path latency is not desired - only the true expected SORTT measurement of a given path. Sections 3.2.2 and 3.3 discuss handling variations in these measurements.

Latency meets design constraints dependent upon Internet path characteristics. Route consistency was a goal of the Internet architecture. Studies of network path stability found that the majority of Internet paths are dominated by a single route and that 90% of routes persist for at least a week (Paxson, 1997) (Zhang and Duffield, 2001). Subsequent studies have shown that these figures have further improved over time. SORTT minimizes dependence on server response time and path congestion. Research has confirmed that Internet path latencies show significant variation (Karn and Partridge, 1991) (Paxson, 1997) (Stemm et al., 2000) (Zander et al., 2005) (Zhang and Duffield, 2001). The only way to tamper with RTT is to inject additional delay (by various means), though many paths may have similar latencies.

**TTL values** refer to the time to live (TTL) field in IP packet headers. TTL is decremented at each node along the network path and is often a convenient measure of a network path (hop count) unaffected by congestion. Design maps the set of observed TTL values (from IP packet headers) listed in their order of appearance throughout a website session. TTL values provide more information than the number of hops to a server because web server packets usually exhibit more than one unique TTL value in a session. For example, sometimes packets alternate between two TTLs. Some OS implementations may use different initial values (than 255), but it is only important to learn the value(s) a client may expect to see from a given path. An example encountered during evaluation is \((64,54,64,64,55,55,64,55,64,55) \rightarrow \{64,54,55\}\)

The mapping does not depend on the number of packets, website content, retransmis-
sions, or TCP implementations. The pattern of TTL values captures characteristics of underlying website or path properties - perhaps load balancing, firewalls, etc. This mapping is difficult to emulate because it requires knowledge of both the initial value and the hop count between each target’s website and machine.

3.1.2 Transport layer

Transport layer features reflect the manner in which a client receives the website content from a server - specifically protocol implementation specifics. Many websites will undoubtedly depend on similar underlying platforms and may exhibit similar characteristics since all websites are offering the same service (HTTP).

TCP options are implemented differently on different operating systems. TCP options mapping builds a string describing several TCP Options, based on Nmap’s TCP fingerprinting encoding (Lyon, 2011), from the initial 'handshake’ SYN packets of a TCP connection initiated by the client. This mapping combines many input fields into a single discrete feature for simplicity. It exploits not only the option values, but their presence and order within the TCP header which vary between operating systems. The mapping captures the NOP, EOL, SACK, MS, Timestamps, and WS fields - arranging a key letter of each option according to its appearance order within the packet header and followed by its supplied value. For example, the presence of an 'L' indicates that an EOL was present. The letter 'N' indicates a NOP for each instance within the options list - some firewalls may reduce select options to a NOP. The number of MSS bytes follows an 'M' and the window scale factor follows a 'W.' {0,1} represent whether an initial value is zero and follow a 'T' for timestamps option. Finally, 'S' represents SACK permitted. A sample TCP Options string is M1430ST11NW6. Both (Lyon, 2011) and (Lee, 2001) attest to the utility of TCP Options in distinguishing websites as does Section 4. Faking this feature requires use...
of the same TCP implementation (that part visible from TCP/IP packets) - which is likely when using the same operating system as a target website.

3.1.3 Application layer

Application layer characteristics describe the website content and HTTP server behavior. Design assumes that an adversary will place the majority of effort into imitating the look of a website and that clients are most likely to detect visible differences. Payload analysis is still fruitful because of the dynamic nature of website content. The use of scripts, Common Gateway Interface (CGI), cookies, and other non-visible content makes the predictability of website content less straightforward.

Content length varies because of dynamic web server content. Web servers may generate data based on client supplied information (such as tailoring content to better display on mobile devices) or by geographic region. Websites embed references to additional resources within the root response - such as an image or script. References may link to a subdirectory of the website or to another website entirely (advertisings, etc.). Advanced browsers, like Firefox (Mozilla, 2011), automatically request linked website resources after parsing the embedded links within the root response. An website session is usually a series of requests/responses. The design mapping measures the size in bytes of the first (root) response to a client’s HTTP GET. If an impostor website contains the same links to additional resources, the only guaranteed fake response is the first (though payload analysis would reveal this phenomena). Emulation may be incidental, but otherwise must match each victim’s local website content.

HTTP header fields are web server settings in the HTTP response header. Feature mapping captures the required and optional HTTP Header fields and stores the field names and their values in an ordered list of string arrays \{field name, value\} exactly
as they appear (to include capitalization) of the first HTTP response. Because many fields’ values vary too much between sessions for this to be a practical method of profile building (lack of predictability), inclusion of each field value occurs by exception. Those exceptions are content-type, server, cache-control, content-encoding, pragma, and the various 'X-' fields. This feature is as consistent as each web server's settings/version. In addition to the field values, (Lee, 2001) has found sufficient variation (and consistency) in the capitalization ordering of HTTP header fields for use in identifying HTTP servers. Emulation may occur using the same web server or by copying the target website, which (again) may differ for different servers hosting the same website.

Digital certificates are practically expected for most websites because they are key to the ubiquitous certificate-based protocols (SSL) securing Internet connections. During the establishment of a secure connection, a website exchanges a digital certificate with the client browser. Certificates identify the certificate authority verifying (signing) a website’s claim of authenticity (and browsers may independently verify). This is useful information also for characteristics based authentication. Impostor websites may likely not use any certificate authority, but it is also impossible for an adversary to both use the same (legitimate) certificate and successfully decrypt information of value from the client (provided there is no subversion of the certificate authority).

3.1.4 Environmental factors

This section describes the observable characteristics dependent on the load or traffic intensity of both the network path and the server and on the performance characteristics of the server (hardware and software efficiency) and network path (physical attributes). For simplicity, the design ignores time (of day) patterns. Many works strive to eliminate the effects of environmental factors (ie. network congestion) from
measurements, but characteristics based authentication should embrace them because many factors influence their behavior.

Latency variation is mostly a function of congestion resulting in additional queuing delays. It is necessary to assume that traffic patterns are predictable, which (Wouhaybi et al., 2008) has shown generally holds over time. The actual mapping captures the ratio of maximum to minimum observed RTT based on work in (Paxson, 1997), which describes a great range in such a metric over various network paths. Network paths may overlap significantly between sites or share similar loads, but otherwise emulation is likely challenging to not only remotely determine the variation for all victim - website paths but induce apparent conditions.

Server response time is a well-studied metric of web server performance. Response time is a function primarily of web server hardware capabilities, HTTP service implementation, and traffic intensity. It may includes the server’s retrieval of a webpage prior to its response. When a browser submits an HTTP request, it typically does so immediately following TCP connection establishment - or nearly coincident with the last packet it sends in the three-way TCP handshake - and both packets will have the same TCP timestamp. This is not true for the time between a server’s ACK of the client’s request and its response packet. That time gap can provide a remote measure of server response time. It is possible to measure the response time for each successive HTTP request in a similar manner, though many servers reduce them beyond detection. (Claffy et al., 1993) and others have shown that response time depends upon the server’s load when a request arrives and (Stemm et al., 2000) found response time was limited by server-side performance and by server computation speed for HTTPS connections. Emulation is difficult if an impostor server does not have the same hardware and service characteristics or traffic load. It may
be impossible to reduce the response time of a less powerful server (vs a legitimate server under light load).

Figure 3.1 depicts the three mapping functions the design uses (in order of attempt). The first maps the TCP timestamp value of the server’s first response packet minus the timestamp of the client’s request for it. Note that both of these packets will have the same TCP timestamp echo reply and ACK numbers. Exploiting TCP timestamps in this manner ignores latency variation between the two packets, but is only effective if the TCP timestamp granularity is fine enough (ms) and used. (Veal et al., 2005) reported over 73% of webservers use TCP timestamps and that the majority have a granularity of 10ms. In the absence of sufficient TCP timestamps, the alternate mapping is the difference between the same two packets’ arrival times at the client assuming identical one-way latencies (not usually valid). When the HTTP response and request ACK arrive in the same packet, response time maps the time increase from the response packet’s RTT over the previous RTT measurement (a SYN packet).
3.2 Profiling

Profiling websites involves learning a website’s known (feature-wise) behavior from a set of (presumed legitimate) fingerprints.

3.2.1 Profile mappings

Mapping features into profiles essentially draws a minimum boundary around the observed values. For continuous features, this is a box plot and provides a degree of generalization for previously unobserved continuous values without making assumptions on the statistical distribution of the data. Latency measurements falling between two previously recorded measurements should be expected, but a similar conclusion for TTL values is unwarranted. A profile of discrete features is the simple union of all observed values. The underlying assumption is that discrete feature values are not random, but relatively stationary and indicative of specific protocol implementations. For example, if mywebsite.com has always responded with an HTTP response header containing "Server = apachewebserver," then any legitimate fingerprints should also have the same server name. Both mappings do not generalize well to predict unusual or changed values, but Sections 3.2.2 and 3.3 address this problem. Note that both mapping functions do not require negative samples or a minimum number of samples (satisfying Constraint 5 in Section 2.3.2).
Table 3.1: Features and mapping functions.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Feature</th>
<th>Char → Feature</th>
<th>Feature → Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>IP addr</td>
<td>1st request dest addr</td>
<td>set of all addr</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
<td>Shortest obs RTT</td>
<td>[min, max] set of all addr</td>
</tr>
<tr>
<td></td>
<td>TTL Values</td>
<td>Ordered set of TTL</td>
<td>set of all ordr sets</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP Options</td>
<td>Coded string</td>
<td>set of all strings</td>
</tr>
<tr>
<td>Application</td>
<td>Content Length</td>
<td>1st resp size</td>
<td>[min,max] set of all strings</td>
</tr>
<tr>
<td></td>
<td>HTTP Header Fields</td>
<td>Field:value array</td>
<td>set of all arrays</td>
</tr>
<tr>
<td></td>
<td>Digital Certificate</td>
<td>Signing Cert. Authority</td>
<td>set of all CAs</td>
</tr>
<tr>
<td>Environment</td>
<td>Latency Variation</td>
<td>RTT: min/max</td>
<td>[min,max]</td>
</tr>
<tr>
<td></td>
<td>Response Time</td>
<td>Resp - req ACK TSV</td>
<td>[min,max]</td>
</tr>
</tbody>
</table>

3.2.2 Profile windows

Constraints 3 and 5 (Section 2.3.2) require a profile robust to unexpected changes in feature values. Naively expanding a profile to include every value observed may steadily increase the false accept rate. An optimal solution is to detect and remove only those feature changes which are permanent while preserving those likely to reoccur or predicting future value changes (perhaps via cluster analysis), but this is a difficult problem. A simpler solution is to place a limit on the number of fingerprints composing a profile to mitigate the effects of permanent changes in feature values.

Restricting the number of fingerprints in a profile to a time (or count) window will guard against the duration that a permanent change causes a profile to remain inaccurate. For example, permanent changes in TTL values from the same location are likely indications of a path change. Too short a window may preclude information necessary to accept legitimate fingerprints if features behave periodically such as in websites hosted on multiple servers. Section 4 evaluates an empirical derivation of the optimal window size. Finally, limiting profile windows attenuates the impact of significant changes affecting a majority of features simultaneously.
3.3 Authenticating

Authentication is determining whether a website is consistent with its presumed profile and likewise whether the fingerprint will be incorporated into that profile. To function within Constraints 3 and 5 (Section 2.3.2), authentication must accommodate changing and unpredictable feature values as well as a lack of counter examples. Strictly matching a profile occurs when a feature value falls within the continuous range or intersects the union of all discrete values. Strict feature-wise profile matching is prone to false rejections - especially given a small window. The design must assume that profiles will not accurately describe all legitimate feature ranges. Its solution is a confidence rate facilitating the inclusion of new samples in proportion to the degree of profile violation. Section 4 analyzes the empirically derived threshold for this confidence rating essential for a zero false accept rate - necessary to maintain profile integrity and more importantly for successful authentication.

Feature confidence ratings and empirically derived thresholds make the system robust to changing characteristics, unrepresentative sampling, and a limited number of samples (even just one). Basing authentication on many different features exploits the difference between a point and collective anomaly. An example is when unpredictable changes adversely affect only a minority of features - a web server operating system change that does not coincide with a change of IP address or website content. A fingerprint with latency 1ms slower than the profile max should not be rejected outright. Section 4 analyzes three different authentication schemes described in Table 3.2 based on finding a collective anomaly in new fingerprints.

3.3.1 Confidence ratings

Confidence ratings hold several properties. The basis of a confidence rating is some distance or similarity measure such as Euclidean or Levenshtein distance (Leven-
Table 3.2: New fingerprint authentication schemes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>MatchSum</td>
<td>Number of features strictly matching the profile.</td>
</tr>
<tr>
<td>ProfileRate</td>
<td>Average of all feature confidence ratings vs the profile.</td>
</tr>
<tr>
<td>BestFprint</td>
<td>Highest average feature confidence rate vs a single fingerprint. Ignores the profile and compares new fprint against all historical fprints (in profile window).</td>
</tr>
</tbody>
</table>

Table 3.3: Notation definitions.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>Fingerprint</td>
</tr>
<tr>
<td>$f_i$</td>
<td>Feature</td>
</tr>
<tr>
<td>$w$</td>
<td>Website</td>
</tr>
<tr>
<td>$\mathcal{P}$</td>
<td>Profile</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Confidence rating</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Profile median</td>
</tr>
<tr>
<td>$f_i \in \mathcal{P}$</td>
<td>$f_i$ matches $\mathcal{P}$ ($f_i$ falls within the continuous range or intersects the union of all discrete values in $\mathcal{P}$)</td>
</tr>
</tbody>
</table>

shtein, 1966) able to provide a means of comparing values. Confidence ratings are normalized in $[0,1]$ such that the differing scales of each feature do not disproportionately weight them. As a feature confidence rating approaches 1, its value approaches the expected profile range. For simplicity, the design attributes values equidistant from the profile range with the same confidence (whether less than the min or greater than the max).

Notation defined in Table 3.3 will assist in the following sections.

Continuous feature confidence rates are a ratio of Euclidean distances from the profile median to feature and profile min/max values. The ratio is the distance from the
Figure 3.3: Confidence rating example: $\delta_1 = \delta_5$.

median to the feature’s closest profile value divided by the median’s distance to the feature value. A continuous feature’s confidence rating is therefore defined as

$$
\delta_{f_i} \triangleq \begin{cases} 
1, & f_i \in \mathcal{P} \\
\arg \min_{j \in \mathcal{P}} \frac{|f_i^j - f_i^\eta|}{|f_i - f_i^\eta|}, & \text{otherwise}.
\end{cases}
$$

(3.1)

The denominator normalizes the mapping since it is strictly not less than the numerator for $f_i \notin \mathcal{P}$. The median estimates the profile expected value - it is less sensitive to outlying values than the mean (recall Constraints 3,5 in Section 2.3.2).

Figure 3.3 shows an example in which $\eta = 3$ and the dotted lines represent the profile min/max values. Confidence ratings of lines 1 and 5 are $\frac{|2 - 3|}{|1 - 3|} = \frac{|4 - 3|}{|5 - 3|} = 1/2$, which is appropriate since lines 1 and 5 are both twice as far from line 3 as lines 2 and 4, respectively. Note that the $\delta_{f_i}$ evaluates to 1 when $f_i = f_i^j$ in Equation 3.1.

Discrete feature confidence rates are not always warranted. They can weaken authentication if feature values are stationary (ie. TTL, TCP Options). Unlike continuous values, minor changes may be just as significant as major changes. For example, value disparities deriving from legitimate website changes will likely result in the renaming or reassignment of a value, while disparities from an impostor website are likely more subtle - typos and capitalization changes. A straightforward confidence
rating will, in such cases, have the opposite effect so it is best to count whether each value is present exactly as in the profile. Levenshtein distance \(^1\) can accommodate this guideline. A simple hashing of each field value into a single character reduces string array feature types into simple strings for Levenshtein analysis. The mapping function then becomes a ratio of the the highest Levenshtein distance of all feature values in the profile set. Normalization occurs by placing the larger of the feature and profile value in the denominator. This is given by

\[
\delta_{f_i} \triangleq \begin{cases} 
1, & f_i \in P \\
\arg \max_{j \in P} \frac{\text{Leven}(f_i, f_j)}{\max(|f_i|, |f_j|)}, & \text{otherwise}
\end{cases}.
\]  (3.2)

Select feature confidence

There will always be special cases. Bit-wise distance or any above method is a poor choice for IP address considering the hierarchical nature of the IP address space. When domain name IP addresses vary, they typically vary consistently within a subnet or cycle within a fixed set of addresses. A previously observed prefix match can provide context for new addresses more likely to occur than a completely different network. A mapping should place greater confidence in nearby servers or those likely within the same administrative domain. The intuition is one that routers make - that addresses belonging to the same region will have a common subnet. One method is to determine a common prefix match from profile addresses. However, some websites may have multiple subnets with completely different prefixes. The best approach is to determine each profile’s normal subnet variations (cluster by prefix) and assign a confidence rating greater for prefix matches within a normal subnet range.

\(^1\) Levenshtein distance is a dynamic programming algorithm that determines the minimum number of edits to convert one string to another. It reflects total ordering as well as the magnitude of changes.
Table 3.4: Feature confidence rates.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Confidence rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP addr</td>
<td>Subnet Match</td>
</tr>
<tr>
<td>Latency</td>
<td>Normalized Euclidean distance</td>
</tr>
<tr>
<td>TTL Values</td>
<td>Profile match only</td>
</tr>
<tr>
<td>TCP Options</td>
<td>Profile match only</td>
</tr>
<tr>
<td>Content Length</td>
<td>Normalized Euclidean distance</td>
</tr>
<tr>
<td>HTTP Header Fields</td>
<td>Normalized Levenshtein distance</td>
</tr>
<tr>
<td>Digital Certificate</td>
<td>Profile match only</td>
</tr>
<tr>
<td>Latency Variation</td>
<td>Normalized Euclidean distance</td>
</tr>
<tr>
<td>Response Time</td>
<td>Normalized Euclidean distance</td>
</tr>
</tbody>
</table>

**Fingerprint confidence**

The confidence rating of a fingerprint is the average of its feature confidence ratings against a certain website profile. This is given by

\[
\delta_w = \frac{1}{n} \sum_{i=1}^{n} \delta_f^w.
\]  

(3.3)

where \( \delta_f^w \) are given by Equations 3.1 and 3.2 and listed in Table 3.4.
Figure 4.1 shows the acceptance rate - which is the percent of legitimate fingerprints the system accepts while rejecting 100% of impostor fingerprints. This is the primary metric showing the efficacy of the system as an authentication method and its shows that Internet characteristics based authentication is viable. Further discussion is below. Table 4.1 lists the evaluated authentication schemes, which are combinations of authentication schemes from Section 3.3 (Table 3.2) and profile windows as described in Section 3.2.2.

There are two aspects of greatest interest: performance and effectiveness. Performance includes the ability to provide a low rating for impostor fingerprints and a high rating for legitimate fingerprints. The main system performance metric is the expected distinction - the expected difference between confidence ratings of impostor and legitimate fingerprints. Further discussion of system performance is in Section 4.2. Expected distinction shows that the fingerprints are effective in separating impostor from legitimate websites in the expected case, but effective authentication requires profile integrity or zero tolerance for even the outlying cases.
4.1 Authentication effectiveness

The high acceptance rates paired with zero false accepts in Figure 4.1 shows that Internet characteristics can authenticate websites and the design is valid, though not perfect. A satisfactory - over 99% via the ’7’ profile window and over 90% for even a single fingerprint profile - legitimate fingerprint acceptance rate indicates that practical use of this system is feasible. However, Figure 4.1 suggests - from the degraded acceptance rate of the ’70’, ’10o’, and ’7w’ windows - that profiles constructed from fingerprints as old as a week may not be effective for authentication. Fortunately, this negative point holds for only the unexpected case as evident in Figure 4.3 where the vast majority of all impostor confidence ratings are significantly lower than all
Table 4.1: Evaluated authentication schemes consist of an authentication scheme (top level) combined with a profile window restriction (bottom level).

<table>
<thead>
<tr>
<th>Authentication Scheme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProfileRate</td>
<td>Average of all feature confidence ratings vs the profile.</td>
</tr>
<tr>
<td>BestFprint</td>
<td>Highest average feature confidence rate vs a single fingerprint. Ignores the profile and compares new fingerprint against all historical fingerprints (in profile window).</td>
</tr>
<tr>
<td>MatchSum</td>
<td>Number of features strictly matching the profile.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>profile built from 1 fingerprint taken the same day as the new fingerprint</td>
</tr>
<tr>
<td>2</td>
<td>2 from the same day</td>
</tr>
<tr>
<td>5</td>
<td>5 from the same day</td>
</tr>
<tr>
<td>7</td>
<td>7 from the same day</td>
</tr>
<tr>
<td>10</td>
<td>10 from the same day</td>
</tr>
<tr>
<td>10o</td>
<td>10 all taken 7 days before</td>
</tr>
<tr>
<td>7w</td>
<td>7 taken 1-per day for 7 days before</td>
</tr>
<tr>
<td>70</td>
<td>70 taken 10-per day for 7 days before</td>
</tr>
</tbody>
</table>

of their legitimate counterparts. More importantly for security considerations, acceptance based on strict profile matchings virtually guarantees an impostor website is rejected. Actually, there were no strict profile matches by any impostor fingerprints in any experiment (no impostor fingerprints received a confidence rating =1).

On the other hand, there was a legitimate fingerprint profile match rate of 60% for the ProfileRate authentication scheme using a '10' profile window which generally increased with window size up to 75% for the '70' profile window.

As discussed in Section 3.3, the system must achieve a zero false accept rate. It can do this by setting a confidence rating threshold such that 100% of empirical impostor fingerprint authentications fall below this threshold. This may not be the optimal solution that generalizes to all possible impostor websites, but it is likely close to optimal (proving this is future work) for this authentication design. The
acceptance rate discussed above for Figure 4.1 is the ratio of legitimate fingerprints falling on the high (right) side of a vertical line drawn on Figure 4.3 at (the x value of) impostor CDF=1. In fact, the confidence rating yielding the impostor CDF=1 is this threshold. Ideally, there will be a complete separation between impostor and legitimate fingerprints, but the system is not perfect and this did not occur with any authentication scheme or profile window combination. The specific values of each authentication scheme’s threshold are not important.

4.2 System performance and design analysis

Figure 4.2 depicts the expected distinction for each evaluated authentication scheme. This metric is better visualized in Figure 4.3, which shows the cumulative distribution function (CDF) of confidence ratings for both impostor and legitimate fingerprints authenticated using the ProfileRate authentication scheme on different profile win-
Figure 4.3: CDF of ProfileRate scheme confidence ratings. Depicts the separation between impostor and legitimate fingerprints.

dows. Expected distinction is equivalent to the length of a horizontal line drawn between impostor and legitimate fingerprints in Figure 4.3. The other authentication schemes had similar CDFs (MatchSum resulted in was a stepwise version with otherwise the same pattern).

There is a clear separation between impostor and legitimate fingerprints for the majority of websites. This suggests that nearly all combinations of authentication scheme and profile window will work well most of the time. This is beneficial in the sense that the design has little control over the number of fingerprints available for profiling. However, evaluation results offer useful conclusions as to design decisions regarding the confidence rating threshold and profile window size.

Figure 4.4 shows the effects of relaxing the zero false accept rate constraint - required for security as discussed in Section 3.3. A low acceptance rate can be annoying to users (and website hosts). Figure 4.4 shows that permitting a confidence rating threshold slightly lower than that required for the acceptance rate metric (zero tolerance of impostors) can greatly increase the authentication rate of legitimate
Figure 4.4: Tradeoff between impostor rejection rate and (legitimate) acceptance rate for the ProfileRate authentication scheme. A minimal increase in the risk of authenticating impostor websites greatly raises the acceptance rate for all profile windows sizes.

fingerprints with a relatively minimal risk of accepting impostor fingerprints. For most window sizes to achieve 100% legitimate acceptance, less than 0.5% of impostors will be accepted. Since in most cases impostor websites do not even come close (Figure 4.3), this may be an acceptable risk for most users.

The effectiveness of all profiles with a limited number of fingerprints upholds Constraint 5 of Section 2.3.2. Figures 4.2 and 4.1 both show that a smaller window is actually beneficial. As a profile adds fingerprints, legitimate fingerprint confidence rates increase as well as impostor confidence rates. There is a cross-over point around seven fingerprints where the benefit of raising the legitimate confidence rate is maximized and beyond which any additional confidence for legitimate fingerprints is offset by an increasing impostor acceptance rate and diminishing acceptance rate.
Features. Figure 4.5 shows that features from many different network layers are useful for authentication, though not all equally. In general, the most distinguishing feature types were discrete. Latency variation was the least helpful - it is likely that many network paths overlapped at the same dominant link driving a similar pattern for many websites.

Authentication scheme tradeoff. It is evident from Figure 4.2 that the MatchSum scheme gives the greatest expected distinction of legitimate fingerprints. This is not surprising since the confidence rate for this profile type is more binary than the others - close but unmatching features do not count at all. This is also the reason that it has the lowest legitimate acceptance rates (Figure 4.1). ProfileRate has a higher profile matching rate (a greater proportion of legitimate fingerprints with confidence rating $= 1$), which is a safer property. It also generalizes better for dynamic conditions or anomalies in sampling and provides a greater distinction in the expected case. The BestFprint scheme is more consistent, has a slightly higher acceptance rate, and requires a lesser window of fingerprints to be effective. BestFprint attempts to account for where multiple servers host the same website and a client’s profile,
actually consisting of several profiles, would generalize too greatly in the event that differencing servers have greatly differing feature values.

Timeliness and other constraints. The system provides authentication information in a timely manner since the system is not computationally or data intensive. The design captures all fingerprint values upon receipt of the first HTTP response. There are no demanding operations - only simple database querying and management. The system has no noticeable overhead and creates zero network traffic. The data requirement is also negligible - less than 15MB for all evaluation data which was an order of magnitude beyond the requirements of normal operation.

4.3 Methodology

The system observed traffic at the network interface card level via jNetPcap (SlyTechnologies, 2010). There were no issues with encrypted traffic based on selected features and because all necessary characteristic observations were taken with the first successful HTTP response. There were no problems though many sites transitioned to secure logins.

3,389 verified online phish sites from (PhishTank, 2011) on 19 January were authenticated against the 46 unique target brands labeled for each phish site. Also authenticated were the targeted sites against themselves. A Google search for the brand names (they were not URLs), resulted in 158 unique URLs most popularly associated with those brand names and subsequent filtering of the URLs was necessary to remove those URLs which were clearly the same brand website (based on a shared IP address) in order to prevent redundant counting. For example, q.ebaystatic.com and p.ebaystatic.com used the same IP address and only one was included. The result was 3,154 profile to fingerprint authentications against 111 website profiles.

Fingerprints were taken while loading each site with default Firefox (only modi-
fied to disable warnings) for the Kubuntu OS to simulate a typical user session. All fingerprints were taken within 1730-2100 hrs on the same machine using a commercial ISP. The data was drawn from the same time each day to isolate the effects of change based on user behavior (not evaluated). Each new (to be evaluated) fingerprint was authenticated against profiles created from known legitimate (via traditional authentication) fingerprints obtained prior to the new fingerprint based on the profile windows in Table 4.1. Relying on the assumption that no certificate authority subversion occurred during the evaluation, the digital certificate feature was omitted.

DNS cache poisoning and IP hijcaking were not attempted. The actual hostnames/URLs of reported impostor phishing sites permitted simulation of the redirection of packets to phishing sites. The evaluation tied each impostor site to a legitimate target via the PhishTank label. IP address was excluded as a feature because the PhishTank data was known to not use the same addresses as their targets and the results would be biased accordingly. There were no IP address matches between impostor websites and their targets while over 75% of legitimate fingerprints did match their profile addresses. Note that actual system deployment should certainly include this feature. For simplicity, indeterminate feature values were excluded from confidence ratings - which happened exclusively for the response time feature in less than 1% of authentications.
5

Related work

Supplemental Authentication via Internet Fingerprinting (SAIF) is related to the broad fields of anomaly detection and profiling.

5.1 Anomaly detection

IDES, an intrusion detection system, builds profiles of observable computer system behavior from audit records for real-time anomaly detection (Denning, 1987). SAIF builds profiles of observable network behavior and uses them for real-time authentication (anomaly) determination. SAIF anomalies are not singleton unusual behaviors, but a combination of multiple unusual values. IDES activity profiles are more complicated - they characterize behavior based on statistical models and multivariate correlations. Multivariate profiling is likely to produce stronger profiles by adding more features and statistical modeling can be more accurate than simple box plots, but SAIF assumptions limit data samples available for analysis.

(Buneci, 2008) detected temporal patterns and anomalies in distributed applications using a feature vector of quantitative and statistical data on time series data. SAIF fingerprints are essentially quantitative feature vectors as well, but ignore time
patterns and statistical data. Similarly identifying patterns in feature values viewed as time series data could better predict future feature values. It would establish data-dependent profile windows rather than imposing a window size arbitrarily on all profiles.

Much like SAIF’s careful selection of features and confidence rating system, the Ntop network anomaly detection system in (Deri et al., 2003) determines anomalies by applying thresholds to recent network traffic characteristics. Ntop analyzes observable network characteristics from all types (protocols) of network flows and finds global rules for all flows based on a counter tracking several characteristics like TTL. Conversely, SAIF develops rules as profile ranges tailored to each source-destination pair and also looks at application data and environmental factors. Ntop pairs alerts with a risk factor indicating how likely an event is to be anomalous when the counter exceeds the threshold. This is similar to the SAIF confidence rating, but Ntop does not combine risk factors and bases them on global behavioral patterns.

Perhaps the most related work is (Mahoney and Chan, 2001), which looked at 33 packet header fields to identify hostile traffic based on a minimal amount of protocol-specific knowledge. The system, PHAD, learns normal ranges of packet header fields at the link, IP, and transport levels from a training data set. This is very similar to SAIF’s fingerprinting and profiling methods, but ignores application and environmental factors and evaluates each packet individually and based on global metrics. Also, PHAD ranks anomalies using the time since an event’s last occurrence rather than in proportion to the data itself.

5.2 Profiling

Nmap is a popular free security scanner that fingerprints machines based on their responses to network probes (Lyon, 2011). Nmap identifies Operating Systems (OS) and other information by comparing responses to a repository (profile) of information
on common OS and service reactions based on variations in protocol implementations. The SAIF TCP Options feature borrows Nmap’s methods. Nmap’s active probes allow for many more available features and a much more reliable profiling of systems. However, it creates a signature and draws features from only the TCP/IP protocols.

Based somewhat on Nmap, Hmap compares web server responses against a compiled list of signatures to determine vendor and version numbers through implementation differences at the application level (Lee, 2001). Hmap profiling is based on particular lexical, syntactic, and semantic differences in HTTP server responses. Hmap’s work for HTTP response headers inspired SAIF fingerprinting of this feature. Hmap restricts features to the application level and to directly observable characteristics (no measurements).

(Kohno et al., 2005) developed a technique to uniquely identify servers using clock skew extracted from timestamp information. (Murdoch, 2006) adapted it to reveal services even within anonymous networks. The technique requires clock synchronization, which SAIF cannot consider because their approaches require more sample data than model constraints permit. These works have already demonstrated the efficacy of clock skew as a reliable remote fingerprinting method for physical devices.

A network-level behavioral clustering system uses similarities among malicious HTTP traffic to identify malware (Perdisci et al., 2010). The clustering algorithm uses a pattern vector of statistical features, such as average response length, from HTTP traffic traces to produce clusters that serve as signatures. This work helped determine the suitable features for SAIF application level fingerprinting and also inspired the use of a normalized Levenshtein distance (they used it on URLs and several parameters). Clustering traces is an alternative to profile window restrictions, but can be unreliable without adequate samples and given outlying feature values.

(Zander et al., 2005) uses network flow characteristics of select features, such as packet inter-arrival time and flow size, to classify flows with an unsupervised
Bayesian classifier (Autoclass). Authors selected features using a sequential forward selection algorithm that found the best attribute set (giving best classification via Autoclass). This is a more effective method than SAIF which (somewhat arbitrarily) selects features based on their known properties, though it is more computationally demanding. Optimizing SAIF’s design will likely require a similar approach to find the best set of features rather than (as currently implemented) a set that works.
Applications. A natural extension to the threat model (Section 2.2) is to incorporate automatic authentication of websites based on similar URLs or content. Currently, the design identifies a user’s intended website from the hostname (via URL assumed typed correctly) in a browser’s first HTTP request. By applying some string matching algorithm to a historical user (or general population) set of frequently visited URLs and likewise some similarity measure of webpages, it is likely possible to reveal whether a user intended to visit a more popular or frequently visited website. The next step will be to apply characteristics based authentication schemes and alert users in the event that visited sites are likely impostors or, alternately, provide a list of closely matching (more popular) known legitimate sites.

Design improvements. There are many ways to improve the design’s simple implementation to make it robust to changes and unpredictable values. It uses empirically derived thresholds for profile windows and confidence ratings. (Zhang and Duffield, 2001) detect change-points and set a historical time window restricting profile values to only the most recent, change-free regions using two algorithms called
CP/Bootstrap and CP/RankOrder. Another method is to accept features within some threshold of the profile - as in 1.5 times the inter-quartile range of the lower and upper quartiles, which captures 99.3% of normally distributed data (Chandola et al., 2009). (Buneci, 2008) detected patterns in time series data.

There are likely many unexploited features that meet design constraints and adding them to the fingerprint would strengthen the design. One option is in tracking external links from embedded content based on the work in (Almishari and Yang, 2010) showing that websites such as ads-portal domains are likely to have different linked content characteristics. Machine learning algorithms can help identify the optimal set (from a list) of features such as in (Zander et al., 2005) - discussed in Section 5.2. A variation is to more heavily weight the more useful features.

Figure 4.5 clearly shows that some features are better than others. The implied assumption that all features are equally indicative of website authenticity is unrealistic. Placing a larger reward on the better authenticating features is logical. An example will be to scale each feature’s confidence rating by its empirically established expected distinction (the values in Figure 4.5). A cursory evaluation of this idea increased the expected distance between an impostor and legitimate website, but did not improve the acceptance rate (primary effectiveness measure) significantly. Results are in Appendix B. A similar idea is to set each threshold by feature for each profile rather than establishing them on data accumulated for all websites.

An evaluation of different profiling and authentication methods was not performed. There are likely better classification algorithms suitable for this task and also more effective confidence assignments. The implementation above was very simple - allowing manipulation of several key aspects of any learning algorithm (profiling, reward, confidence) and was sufficient to demonstrate the utility and some design trade-offs of characteristics based authentication.
An evaluation of the design in this research has shown that Internet characteristics can be effective in authenticating a website to a user. The fingerprinting and profile authentication approach can supplement certificate-based website authentication without requiring the trust of a third party or any external support. Many characteristics of an Internet connection between a web browser and website are difficult or impossible to imitate and vary significantly between websites because of the complexity and imprecise nature of the Internet protocols. This information is currently unexploited and the authentication system evaluated can reject all impostor websites with a legitimate rejection rate conservatively under 1% with no noticeable overhead and zero network footprint.
Appendix A

Background

A.1 Internet Protocol

The Internet Protocol (IP) is the Internet’s primary method for machines to communicate across a computer internetwork using datagram packets. It delivers datagrams (packets) from source hosts to destination hosts based on their IP addresses, which are 32-bit fields (IP version 4) in the IP packet header. This delivery protocol is essential since the Internet is composed of several autonomous systems (very basically) consisting of a collection of host machines connected via physical links to routers which forward packets according to their IP addresses. IP provides a ‘best effort’ attempt to deliver packets and so other protocols have emerged to ensure reliable delivery because packets might be lost, delivered out of order, etc. One such protocol is discussed in Section A.2 below.

To ensure that packets do not circulate forever in the Internet, each IP packet has a Time To Live (TTL) header field. The idea is that each router will decrement the TTL field value in a packet for each second it holds the packet, but in practice this has become a single decrement for each router. Commonly, the TTL field is used
as a hop-count representing the number of network nodes or routers (hops) a packet has taken from the source to destination. In Figure A.1, a diagram of an IP packet header, each `-` represents one bit out of the 32 bits per row. There is much more to IP than this brief overview.

Not a part of IP, DNS (referenced in the introduction) provides a hostname to IP address translation service. Where IP addresses are how computers send and receive packets according to IP, humans prefer more meaningful or memorable addresses (google.com) which must be translated by DNS into IP addresses. It is possible for a hostname to have more than one IP address and for one IP address to translate to more than one hostname (virtual hosting). The former is true for non-static IP addresses. Dynamic address allocation (DHCP being the most notable) solves many problems and is a common practice. Hostnames may have multiple IP addresses also because multiple machines provide their web content based on load balancing, content distribution, or server replication.

A.2 Transport Control Protocol

Where IP does not provide for reliable delivery, Transport Control Protocol (TCP) does exactly that. One of the core protocols of the Internet, TCP creates a state-
based connection between Internet hosts. There are several variations of the protocol, but all provide additional information through fields within the TCP header of a TCP/IP packet, which immediately follows the IP header and precedes the datagram payload. These fields allow hosts to maintain the order of the data, track and resend missing packets, and do so over multiple connections. TCP provides many other benefits and options. Below is an explanation of selected fields and functionality relevant to this thesis.

TCP tracks the state of each connection. Each TCP header has a SYN flag which alerts hosts to the establishment of a connection and likewise a FIN flag to indicate the conclusion of the connection. Establishing a TCP connection is known as a three-way handshake in which the initiator first contacts the other with a SYN packet which the other acknowledges by sending an ACK (acknowledgment) packet and then finally the initiator responds with its own ACK of the first ACK. These three SYN-flagged packets also help synchronize information such as the TCP options the hosts will use and the initial sequence numbers. Data may flow in both directions until each direction indicates that no more data follows by setting the FIN flag.

In order to provide reliable delivery, TCP sequence numbers identify the ordering
of the packets. A sequence number corresponds to the total message byte ordering of
the first payload byte in the packet. When a host sends an ACK upon receipt of each
data packet, it includes the next expected sequence number. Therefore, the sender
knows that the receiver has received at least much data as the last ACK number
and may retransmit packets it thereby infers are lost. There are many variations
to this basic protocol description that have emerged to achieve certain trade-offs
in performance and which impact the measurement of Internet characteristics, but
further discussion of which is in Section 3.1.

A.3 Hypertext Transfer Protocol

Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web
(www) and is a standard for transferring media content - like websites. An HTTP
session is a series of requests and responses - from clients (user agents - web browsers
typically) to Internet web servers (which provide the website content). An HTTP
request follows the TCP header and is usually similar to the following:

GET /index.html HTTP/1.1
Host: www.examplewebsite.com

Requests must provide the host information in the case of virtual hosting where a
web server with a single IP address may provide services for multiple websites with
different URLs.

The first requested resource is typically an HTML document (basic webpage),
but website content is often more complex. There may be links to images, scripts,
or even other websites (ie. advertisements) embedded within the first response.
Typically, popular web browsers will request such resources for display or script
execution or check on the status of external links in order provide a more seamless
user experience. The content may be dynamic based on location or on the services
provided. A response (header) from a server resembles:
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3 (Unix)
Content-Length: 438
Content-Type: text/html; charset=UTF-8

consisting of several fields providing both standard and optional information and followed with the content of the requested resource. Further discussion of HTTP response field values is in Section 3.1.
Appendix B

Reward rate

Figure 4.5 clearly shows that some features are better than others. Assigning reward rates to each feature relaxes the implicit assumption that all features are equally effective in discriminating websites. Scaling the confidence rating of each feature by its empirically established expected distinction (the values in Figure 4.5) rewards features by their ability to distinguish legitimate websites. In order to keep the resulting ratings normalized in [0,1], each reward rate must be divided by the sum of all feature rewards. Letting $r_{fi}$ denote feature reward (empirical expected distinction) and recalling the notation in Table 3.3, a fingerprint’s reward rating is therefore given by

$$\frac{\sum_i [r_{fi} \times \delta_{fi}]}{\sum_i r_{fi}}.$$  

Figures B.1 and B.2 suggest that reward ratings increase the expected distance between an impostor and legitimate website, but only improve the acceptance rate (primary effectiveness measure) significantly for larger profile windows. The primary effect is to reduce impostor fingerprint ratings. This is evident in Figure ref:rewCDF with the left shift of impostor fingerprint CDFs (heavy to light dashes).
Figure B.1: Comparison of reward and confidence acceptance rates. Shows that reward rates do not significantly improve authentication effectiveness.

Figure B.2: CDF of reward and confidence ratings. Shows that reward ratings increase the expected distinction between impostor and legitimate fingerprints.
Bibliography


Buneci, E. (2008), “Qualitative performance analysis for large-scale scientific work-


Claffy, K., Polyzos, G., and Braun, H. (1993), “Measurement considerations for as-
sessing unidirectional latencies,” Internetworking: Research and Experience, pp.

Engineering, 13, DOI=http://dx.doi.org/10.1109/TSE.1987.232894.

an anomaly detection system: An empirical approach,” Terena Networking


transport protocols,” ACM Transactions on Computer Systems, 9, DOI=http://
doi.acm.org/10.1145/118544.118549.


