Slicing Spam with Occam's Razor[*]

Chris Fleizach
U.C. San Diego
9500 Gilman Dr.
La Jolla, CA 92093-0404
cfleizac@cs.ucsd.edu

Geoffrey M. Voelker
U.C. San Diego
9500 Gilman Dr.
La Jolla, CA 92093-0404
voelker@cs.ucsd.edu

Stefan Savage
U.C. San Diego
9500 Gilman Dr.
La Jolla, CA 92093-0404
savage@cs.ucsd.edu

ABSTRACT

To evade blacklisting, the vast majority of spam email is sent from exploited MTAs (i.e., botnets) and with forged "From" addresses. In response, the anti-spam community has developed a number of domain-based authentication systems – such as SPF and DKIM – to validate the binding between individual domain names and legitimate mail sources for those domains. In this paper, we explore an alternative solution in which the mail recipient requests a real-time affirmation for each e-mail from the declared sender's MX of record. The *Occam* protocol is trivial to implement, offers authenticating power equivalent to SPF and DKIM and, most importantly, forces spammers to deploy and expose blacklistable servers for each domain they use during a campaign. We discuss the details of the protocol, compare its strengths and weaknesses with existing solutions and describe implementation strategies.

1. INTRODUCTION

By almost any metric, spam email has become a pervasive blight on e-mail users and service providers alike. The low marginal costs of spam delivery combined with the effectiveness of early content-based filtering and domain-based blacklisting have led spammers to develop large-scale remailing infrastructures in response. Thus, a modern spam campaign can comprise hundreds of millions of messages, addressed from tens of thousands of domain names, and delivered via thousands of distinct Mail Transfer Agents (MTAs). Indeed, with some reports indicating hundreds of millions of compromised "bot" hosts on the Internet [5], the ability to produce 100 million spam messages a day has become a trivial task.

In response, the anti-spam community has focused considerable attention on limiting domain address spoofing and, through it, the ability to create an effective large-scale spam mailing infrastructure. For example, the Sender Policy Framework (SPF) [3] and DKIM [2] systems allow receivers to validate if an email's "From" domain address is consistent with the source of the message (authenticated via digital signature or the IP address(s) of the sending MTA). While each of these approaches has its benefits, neither is in pervasive

---

use today and at least some of the early adopters have been spammers themselves.

In this paper we present a real-time challenge-based authentication protocol called *Occam* based on an exceedingly simple algorithm: when an email arrives, the receiving MTA sends a validation query back to the server who "should" have sent the message (the MTA responsible for the domain claimed as the source). If this MTA responds that it has indeed sent the message, then all is well; if not, then the domain has been spoofed and the contents are likely a spam or phish. In a real sense, this is mail authentication stripped to its barest essentials.

We believe the Occam protocol offers two contributions over previous approaches. First, Occam is extremely simple to deploy. For all small to medium-sized domains, Occam can simply be enabled – with no site-specific configuration at all – and yet deliver equivalent authenticating power to SPF or DKIM. In Section 4, we discuss how large domains can use Occam. Second, because Occam is a challenge-based authentication system, it shifts the burden of mail authentication to the sender on a per-domain basis. Thus, to participate in the protocol a spammer must provide online server resources and these servers must be capable of answering queries about any e-mail sent from that domain. This requirement increases the infrastructure demand on the spammer and, moreover, the addresses of these servers must be exposed during a spam campaign, thus becoming prime targets for blacklisting.

The remainder of this paper is structured as follows. In Section 2 we describe our proposed Occam protocol. In Section 3 we analyze how spammers might try to sidestep Occam and highlight the strengths and limitations revealed by this evaluation. We describe how large and small domains can implement Occam in Section 4 and discuss performance overheads in Section 5. We conclude with a summary of our findings in Section 6.

2. OCCAM'S RAZOR

Occam is a new protocol for combating the growth of spam email. Occam is motivated by the observation that most spam email contains forged addresses typically sent by botnets [7]. The strength of the protocol lies in its simplicity. Using Occam, receivers simply ask the sending domains identified in a message whether they actually sent the message. With legitimate email, the domains will acknowledge sending the message. Spam email that forges the domain, however, will not be acknowledged and receivers can classify the email as illegitimate. As a result, Occam requires spam-
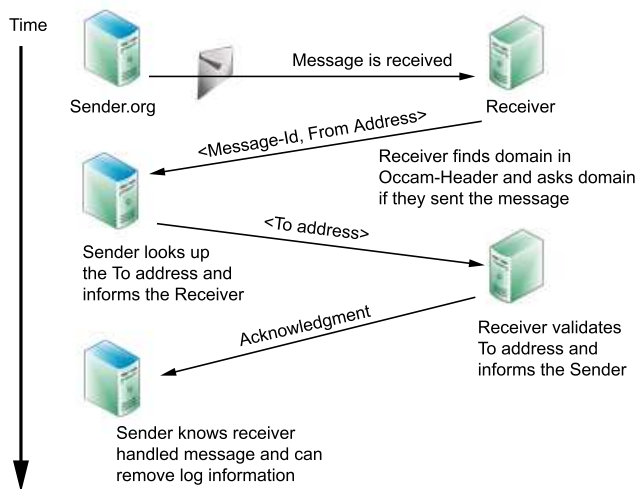
Figure 1: Outline of the Occam protocol for a valid email exchange.

mers to provide available resources for acknowledging the spam they send, and induces spammers to control their own domains and identify those domains in their spam email. These requirements increase the resource burden on spammers and further expose spammers to effective blacklisting. In this section we focus on the operation and implementation of the protocol, and further discuss the implications of Occam in Section 3.

Figure 1 illustrates the operation of the Occam protocol. When a receiving server receives a message, it parses the *Occam-Header* to determine the sending domain. The Occam-Header looks like an email address, with a user and a domain, but the domain specifies the server to contact for confirmation:

        Occam-Header: bob@serverB.org

In most situations, the Occam-Header will be identical to the envelope-sender, also known as the Return-Path. The receiver should only use the DNS MX records when resolving the domain. This requirement helps prevent using botnets as valid domain servers, since many botnets are desktop computers and generally do not have MX records (even though they may have A records). As a concrete example, assume that server A receives a legitimate message for the user *alice@serverA.org* that contains the Occam-Header *bob@serverB.org*. The Message-Id listed in the email message is *Id-1234*.

The receiver then sends a query message to the server for that domain. In this case, server B is the server that resolves to the MX record for serverB.org. The query includes the email "From" address together with the Message-Id:

    From: bob@serverB.org
    Message-Id: Id-1234

The domain server uses these fields to identify the message that was sent in a log of recently sent messages. As a result, Occam requires each domain server to maintain a log of sent messages. Each log entry only needs to include the "To" address, the "From" address and the Message-Id. The "From" address is not always necessary, but provides information for a domain server to identify potentially abusive clients that try to guess Message-Ids and subvert Occam.

The domain server can expire the log entries when the receiver acknowledges the response from the server. These fields can be compressed to about 30 bytes per record; even an outstanding log of 100 million entries would only require 300 MB, small by any standards for contemporary servers.

If the domain server finds the message in the log, it returns the "To" address of the corresponding message back to the receiver as acknowledgement. The acknowledgment allows the sending server to remove the entry from its log. In our example, server B responds with:

        To: alice@serverA.org

Note that since the domain server synchronously sends the response to the receiver using the same socket binding as the query request, it does not need to specify the Message-Id; the receiver knows which request the response matches.

The receiver validates that the "To" address from the domain server matches the "To" address in the original email message; requiring the "To" address in responses prevents malicious domain servers from simply acknowledging all Occam queries. If the "To" addresses match, then the receiver finally delivers the message to the user's mailbox and acknowledges the match to the domain server:

    Status: Received
    Message-Id: Id-1234

The domain server can now remove the corresponding entry from its log of messages sent. Note that, in this exchange, the receiver asynchronously acknowledges the match to the domain server and must explicitly specify the Message-Id.

These exchanges correspond to the case when the domain sender has sent legitimate email. There are three cases where the protocol detects illegitimate email. First, if the domain server does not find the message in its log when queried, it responds to the receiver accordingly:

        Status: Unknown

Or, if the domain server does not find the message logged and it wants to know who the message was sent to, it can respond to the receiver asking for the "To" address:

        Status: Unknown To

The receiver can then respond with the "To" field from the message and remove the email as it sees fit. In our example, server A would return:

        To: alice@serverA.org

Second, if the receiver finds that the "To" address does not match what was in the email, then the receiving server concludes that the sending server did not actually send the message. In both cases the receiver can then take appropriate action against the illegitimate email.

Third, if the receiving server does not immediately receive a response from the domain server, it can limit the rate of querying, much like SMTP does when repeatedly trying to deliver messages to bad addresses. After each attempt that does not succeed, the receiving server doubles the amount of time to wait before retrying. After a timeout, the messages can be marked as illegitimate if no response was ever received.

3. SLICING SPAM

Having described the operation of the Occam protocol, we now discuss how spammers might respond to Occam, and what constraints and burdens Occam places on spammers for them to continue to deliver spam successfully. We then discuss the advantages Occam offers compared to current methods, as well as its limitations.

3.1 How might spammers respond?

The goal of Occam is to impose a substantially higher resource burden on spammers, and to further expose spammers to effective blacklisting. Naturally, spammers will respond to Occam and change how they deliver spam.

Put the bots to work. Occam requires senders to validate and acknowledge the email they send. Spammers could try to distribute this load across the bots they already use to send out their spam and use the existing domain name for the bots in the Occam-Header. Occam, however, identifies senders using only the MX records to resolve domain names for servers. As a result, Occam makes it challenging for spammers to use generic bots for this purpose. Since many bots are hosts that will not have MX records that resolve to them, such bots would not be able to validate spam that they send. The implication is that bots cannot be used for the crucial step of validating messages with Occam, although they can, of course, continue to send messages.

In response, spammers could compromise or purchase bots that do have appropriate MX records, but harvesting specialized bots increases their cost and diminishes spammer profits. Alternatively, spammers could establish a DNS domain structure where each bot is assigned a separate subdomain or entirely new domain. Spammers could create MX records to have the domains used in the Occam-Header resolve to the bots spending the spam. Such an elaborate DNS domain structure, though, makes spammers more vulnerable to blacklisting and increases cost. If spammers use many sub-domains, one per bot, the sub-domain structure would create a telling signature that the entire domain is being used as a source of spam. Given this signature, all of the sub-domains could then be easily blacklisted by blacklisting the entire domain. Spammers could use many domains instead of sub-domains, but doing so greatly increases the cost and burden of managing the bots for sending spam. Further, the list of domains directly expose the identity of the bots and expose them to blacklisting. Finally, allowing bots to respond to Occam queries assumes they can accept incoming connections on low numbered ports, a policy which many ISPs do not allow.

Centralization. Rather than distributing the load across many bots, spammers could instead use a centralized server to handle the request load for validating the spam messages they send. In this scenario, spam would have an Occam-Header that resolves to this server. This server would then acknowledge requests from receivers so that spam could successfully be delivered. Spammers could still successfully deliver spam, but Occam forces this server to stay online as spam is sent. It increases the complexity and cost of managing and operating a spam campaign, and the server becomes an obvious target for blacklisting.

With Occam, spammers have to keep track of the email targets that they distribute to each of their bots. Because Occam requires the validating server to respond with the "To" address used in the original email, the validating server cannot blindly acknowledge all requests from receivers. Instead, spammers must precompute and distribute Message-Ids with "To" and "From" addresses to the bots being used for spam relay. The validation server must keep this list so that it can successfully reply to receiver requests. And

spammers must provision the server so that it can handle a validation request load that grows in proportion to the number of spam messages sent. A spammer can use a botnet to send out millions of spam emails, but for them to be successfully delivered the spammer will need a server that has the resources to handle responding to millions of validation requests. As a result, Occam shifts the resource burden from the receiver to the spam sender. In other approaches, such as DKIM, the burden remains on the receiver, which must download keys and cryptographically validate each message. Furthermore, Occam requires spam senders to keep their validating server available and responsive during a spam campaign. Concentrating validation on a centralized server exposes that server to blacklisting. Once the server is associated with spam, adding the server to a blacklist will prevent spam validation and thereby delivery. Since spam campaigns can persist for days [1], early blacklisting can substantially impede spam delivery.

DDoS Reflector. Finally, spammers could use the protocol as a reflector DDoS attack. A spammer could send millions of messages claiming to be from a targeted domain identified in the Occam-Header. As a result, receivers will then direct millions of validation requests to that domain. If a site has multiple MX entries in DNS, this configuration could result in a multiplicative increase in the number of queries. Larger sites could likely handle this load, but smaller sites could be overwhelmed (we show in Section 4 how larger sites can avoid the amplification problem). However, when overloaded, a site could just as easily start dropping requests and rely on Occam's backoff and retry mechanism to distribute the load over time. More generally, though, if attackers want to use Occam to launch DDoS attacks, it would be easier for them to launch the attacks directly rather than use Occam. If an attacker can send out millions of messages, they are capable of a straightforward DDoS attack on the domain rather than performing an indirect attack through Occam. Indeed, they could simply send millions of messages directly to that domain, independent of whether Occam is used.

3.2 Advantages

The Occam protocol offers a number of distinct advantages over other methods that are currently in use.

Ease of administration. Occam does not require effort by administrators to make the system work, an important consideration for the thousands of small domains that may not have the technical expertise for more complex approaches. To use DKIM, for instance, domain administrators must create and insert a public key into a special DNS record. They then must configure the outgoing MTA to append signatures to all messages using a private key on each message. Presumably, they would also want to set up the MTA to handle incoming mail using DKIM as well. This process requires a certain degree of proficiency that may not be available for many small domains. The Occam protocol, however, can be implemented directly in MTA software packages, such as Sendmail, qmail or Microsoft Exchange Server. It can then be rolled out into a software upgrade, a process that is more familiar to users. We note that Occam is straightforward to deploy for a small domain. Larger domains will need a more involved process to implement Occam than with SPF, which only requires a DNS entry to be inserted. We argue

that adoption of a protocol depends equally on its acceptance by small and large domains, and Occam makes this process easy for the small ones.

Enhanced culpability. The Occam protocol enhances what approaches like SPF and DKIM can accomplish. Both systems validate that a message came from a specific domain or that a sender is authorized, but the burden of proof rests with the receiver; again, with DKIM, the receiver must perform a cryptographic operation on each message. Moreover, the approach is still open to abuse. For instance, a spammer can just as easily set up a domain that has a perfectly valid SPF rule that specifies any IP address can send mail for the domain. A botnet can then send an unlimited number of messages that all look legitimate from the standpoint of SPF. They could alternatively find "open SPF relays" that allow any sender to send messages. This workaround undermines the values of blacklisting domains based on SPF abuse. Occam, however, shifts more of this burden to the spammer. It forces the actual sender of a message to be involved in its authentication in an online manner. Legitimate hosts stay online and available as a matter of course. Spammers have gone to extreme lengths to avoid being detected and pinned down to a valid online presence. Thus Occam makes spamming more difficult to accomplish without creating an exposed and more expensive centralized infrastructure. Occam, in effect, undermines the value that botnets provide to spammers.

Real-time validation. Occam requires that the "work", in our case responding to a validation query, be performed online by the sender of the message. This requirement contrasts with protocols like Hashcash, where the "work" can be precomputed during idle time across thousands of botnets before any spam campaign begins. With Occam, the spammer must be able to respond successfully to all the queries that arrive in real-time. The effect of responding in a timely fashion is that the spammer must have a valid domain name that resolves to a server in their employ. This server must be available to accept queries on the Occam port and be provisioned well enough to respond to many queries. The Occam protocol forces the spammer to expose this higher value target, presumably more expensive to obtain, and makes the domain and IP used an easy target for blacklisting. If the spammer attempts to switch to a different IP address, the domain still remains blacklisted. Since the spammer must own that domain, blacklisting a domain can no longer affect the credibility of domains that are normally "hijacked," as is done currently.

Input for reputation systems. As mentioned above, a spammer could register many domains and keep changing DNS records so that they point to new servers able to answer queries. However, these rapid DNS changes would create a telling signature in their short TTL and IP churn. According to [9], webmail services are establishing reputations for domains that allow them to filter spam more effectively. These two characteristics, IP churn and short TTL, would be clear indications that the domain was involved in sending spam, evidence that reputation systems could use to reliably identify spamming domains.

Anti-Phishing capability. An unexpected benefit of using Occam is that domains will immediately become aware of when they are being phished (or, more generally, being spoofed). Since receivers will begin querying a spoofed domain for non-existent messages, Occam enables domains to discover immediately when they are being spoofed. Moreover, Occam provides a mechanism for a receiver to determine the "To" address to which a phishing email was sent. Such information would be useful to companies that must often deal with phishing attacks, as it allows them to flag accounts to watch for suspicious activity or to take other measures to contact the users that they know have been exposed. The ability to be notified immediately of phishing and spoofing would consequently be available not only to large and well-funded companies, but any organization, thereby reducing the effectiveness of more elaborate attacks like spear-phishing.

Phishers could try to avoid having the original domain know about the phishing attempt by specifying one of their domains in the Occam-Header. However, aside from the difficulties spammers would have in using their own servers, the discrepancy provides a strong indication that a message is illegitimate if the domain in the Occam-Header lies outside of the top level domain for an organization.

Low overhead. The Occam protocol is simple to implement and straightforward to deploy. It also imposes low overhead to operate. The overhead is proportional to the number of messages received and sent, imposing little additional burden on both small and large sites.

3.3 Disadvantages

As with any approach, Occam has disadvantages as well, which we discuss below. For the large majority of domains, though, we believe the benefits of Occam outweigh these disadvantages.

Mobile mailers. There are some legitimate reasons that a sending server might not be able to respond to an Occam query. One is to retain the ability to send mail from a host intermittently connected to the Internet, while allowing another server to handle incoming mail and SMTP related functions, like error messages. We believe this flexibility in SMTP is abused by spammers and that it is in the best interest of most servers to exert greater control over who is allowed to send mail claiming to be from their domains.

Denying service. The Occam protocol also opens up a potential denial-of-service attack against email receipt. An adversary could potentially try to query for and acknowledge email requests from a sending server in an attempt to make them remove their logs prematurely, thereby preventing delivery of the email by the receiver to the original recipient. As an example, if server A sends a message to server B, a malicious server C could try to guess the Message-Id and the From address and reply more quickly to server A than server B does. Server A would acknowledge sending the message and remove its log information about the message, causing a subsequent validation by the real receiver, server B, to fail.

However, precisely since the Message-Id and the From address are required information for querying a server, there is a reduced chance such an attack would succeed since an attacker has to guess these fields. Correspondingly, it

would benefit all MTA software to add more entropy to the Message-Id fields. Further, most queries by the legitimate receiver would happen in a relatively short amount of time, limiting the window of opportunity of an attacker. Finally, a sending server could keep the necessary information around for some period of time after it has been successfully queried before removing it.

4. IMPLEMENTATION

We have developed a prototype implementation that works with the Sendmail MTA [8]. Given its ease of implementation, it should be straightforward to extend any MTA with an implementation of the Occam protocol. For the sake of brevity, we focus here on the more interesting discussion of implementation in large domains.

For sites that handle much larger volumes of email, scaling could add complexity to the implementation, but need not. Moreover, these sites likely have support to address scaling issues already. The Occam protocol requires only basic logging and querying functionality, operations that can be streamlined with database servers if necessary. Overhead due to the Occam protocol would not then be significant for domains already sending large quantities of email. More significantly, Occam imposes the same requirements on spammers: large-volume spammers would have to similarly scale their logging and querying ability, often under the constraints of using bots. This operation is another requirement that reduces their return on investment, making spamming less profitable.

Our prototype implementation is unsuited for domains that have dozens or hundreds of mail server. These domains can approach Occam in a different manner, however. One option would be to centralize their logging and query operations, although doing so may be awkward for large sites. Instead, large domains can load balance logging and querying in a straightforward fashion. A mail server sending mail can add an Occam-Header that points back to that specific server, instead of the entire domain. Doing so distributes load naturally and associates, on the same server, the process of sending mail and answering queries, requiring no coordination among large, distributed mail server farms. This solution also eliminates the amplification DDoS attack. Each of these mail servers do not need to have multiple MX values. On the other hand, receiving mail servers who do not respond to Occam can refuse incoming Occam queries, and these packets can be dropped at border gateways.

5. ESTIMATED IMPACT

An important consideration for any new protocol is the impact it would have on the current Internet infrastructure and the servers responsible for deploying it. The Occam protocol does have the potential to raise bandwidth costs and server utilization. We argue that these costs, however, have minimal impact.

We examined the time overhead for our Occam prototype implementation compared to similar approaches, DKIM [2] and SPF [4]. We sent 1,000 messages from one server to the other five times. Without any additions, Sendmail took 100.4s to transfer the messages. With SPF enabled, it took 100.03s. With DKIM, the total time was 251.2s, a significant additional overhead. Finally, with Occam, it took 102.4s, showing that the overhead is not burdensome.

The communication overhead of Occam is proportional to the amount of email, particularly spam email, sent on the Internet. Given the amount of spam sent on a daily basis, this overhead might be substantial. To estimate the overall communication overhead on the Internet, we can perform a back-of-the-envelope calculation to indicate the added data costs that the protocol would impose. If we take an upper estimate on the number of email messages sent [6], there were 171 billion messages delivered daily in the first quarter of 2006, of which 71% were spam. If every one of those email messages required three UDP packets to determine their status, plus one MX record lookup, where we assume the packet size is a generous 200 bytes, then 800 bytes would be needed by Occam per email. This overhead would add 136 TB of total data into the Internet per day. Spread out over an entire day, the load averages 1.58 GB/s, a very small rate at Internet scales.

6. CONCLUSION

The Occam protocol provides a simple light-weight mechanism for authenticating e-mail messages. Its simplicity makes it easy to understand and, as well, easy to administer. Moreover, spammers who would choose to adhere to the protocol are forced to support and expose dedicated infrastructure for the duration of their campaign. Finally, as a side-effect, the Occam protocol notifies domain owners when their addresses are being spoofed, a useful feature for combating phishing attacks. Occam is not a silver bullet for solving the spam problem and, like most anti-spam technology, is most effective in tandem with existing approaches including spam filtering and blacklisting services. However, we believe Occam's advantages make it a valuable addition to the repertoire of weapons in the fight against spam.

7. REFERENCES

[1] D. S. Anderson, C. Fleizach, S. Savage, and G. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *16th USENIX Security Symposium (Security'07)*, Aug. 2007.

[2] domainkeys-milter, 2007. http://sourceforge.net/projects/dk-milter/.

[3] E. Kurmanin. smf-spf Sendmail SPF milter, 2007. http://smfs.sourceforge.net/smf-spf.html.

[4] libspf2 - SPF libary, 2007. http://www.libspf2.org/.

[5] J. Markoff. Attack of the zombie computers is a growing threat, experts say. *New York Times*, Jan. 2007. http://www.nytimes.com/2007/01/07/technology/07net.html.

[6] Radicati Group. Worldwide daily email traffic climbs to 171 billion messages, spam rises to 71 percent, says radicati group, May 2006. http://www.tekrati.com/research/News.asp?id=6933.

[7] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proceedings of the ACM SIGCOMM Conference*, Pisa, Italy, Sept. 2006.

[8] Sendmail Consortium. Sendmail, 2007. http://www.sendmail.org.

[9] B. Taylor. Sender reputation in a large webmail service. In *In Proc. of the Conference on Email and Anti-Spam (CEAS'06)*, 2006.