warmy
Email Channel. Reliable.

March 2025

Modern Email Authentication and Inbox Warm-Up: Challenges and Best Practices

This report dives into the underlying causes behind email service provider blocks—from issues with sender reputation and authentication to sudden volume spikes and spammy content—and outlines actionable strategies to prevent and recover from them. It provides an in-depth look at how major providers like Gmail, Outlook, and G Suite manage email delivery, ensuring that your outreach consistently lands in the inbox.

About Warmy and the Research Team

Warmy is the leading email deliverability technology, helping businesses improve their inbox placement, sender reputation, and overall email performance. Powered by AI-driven strategies.

The Warmy Research Team is a dedicated group of email deliverability-certified experts focused on analyzing and optimizing email-sending practices.

Through continuous testing, data-driven insights, and innovative methodologies, they uncover factors that impact deliverability and translate findings into actionable improvements for Warmy's platform. Their expertise helps businesses navigate the complexities of email deliverability with confidence.

Daniel Shnaider
Deliverability Expert

Alexandr Panchenko
Technical Deliverability Expert

Vahagn Shirinyan
Senior Deliverability Expert

Max Popov
Senior Deliverability Expert

Oleksiy Lutskin
Deliverability Expert

Artem Klymenko
Deliverability Expert

Bohdan Tsapenko
Head of Research Team

warmy

The Warmy.io team

Table of contents

Key points or TL:DR

- **Reputation & Authentication Matter**: New or poorly authenticated domains and IPs are prone to blocks. Ensure SPF, DKIM, and DMARC are set up correctly.
- **Content and Volume are Crucial**: Avoid spammy language, excessive links/attachments, and sudden high-volume sends. Gradual warm-ups build trust.
- **Engagement is King**: High bounce rates and low reply rates damage sender reputation. Focus on maintaining clean, engaged mailing lists.
- **Provider-Specific Nuances**: Gmail prioritizes engagement and domain reputation; Outlook/MS365 focus more on IP reputation and blacklist checks. Tailor your strategy accordingly.
- **Proactive Monitoring & Recovery**: Use tools like Gmail Postmaster Tools and Microsoft SNDS to track performance, and address issues quickly to recover from blocks.

Introduction to Email Warm-Up and Deliverability

- Inbox warm-up is the process of gradually sending and interacting with emails from a new or cold email account to build its sender reputation. Much like stretching before a sprint, warm-up slowly ramps up your sending volume and engagement to signal to email providers that you are a legitimate sender. This matters because email providers (like Gmail, Outlook, Yahoo) use sender reputation to decide if your messages land in the inbox or the spam folder. Even a well-crafted email is useless if it never reaches the inbox.

Key factors influencing deliverability include:

- **Sender reputation**: a history of sending wanted emails with low spam complaints. A new domain or IP has no reputation, so warm-up helps establish a positive track record gradually.

- **Authentication setup**: proper DNS records like SPF and DKIM that verify your domain's emails. Without these, providers may mistrust your messages or flag them as spoofed. Warm-up efforts are far more effective when emails are authenticated because providers can attribute the activity to your domain.

- **Engagement**: user interactions such as opens, replies, and low bounce rates. Warm-up services often simulate positive engagements (opening emails, replying, marking as important, etc.) to boost your domain's reputation.

In summary, inbox warm-up is a foundational deliverability practice for new senders. It builds trust with mail systems by sending incremental volumes of email and securing solid engagement, all under the umbrella of proper authentication. A successful warm-up means that as you increase volume, your emails continue to land in inboxes, not spam, thanks to a growing positive reputation and correct email authentication.

POP3 and Modern Security Limitations

Many modern email providers like Gmail, Outlook, and Yahoo still support legacy protocols such as POP3 and IMAP for retrieving email. The reason is largely historical and for compatibility – countless users and devices rely on these standard protocols, and they remain useful for accessing email from desktop clients or consolidating mail from multiple accounts. Google, for instance, allows users to open Gmail messages in other mail clients via POP, and it has extended POP/IMAP to support OAuth 2.0 authentication rather than disabling them entirely. In short, POP3 is "not going away" even as security improves; it's simply being adapted to work with modern login methods.

However, POP3's design comes from a less secure era and has limitations when faced with today's security requirements. Traditional POP3 (and IMAP) authentication uses a simple username/password exchange, which is incompatible with multi-factor authentication (MFA). There's no mechanism in the POP3 protocol to prompt for a secondary code or app confirmation. As a result, if an email account has MFA enabled, you cannot use the normal password over POP3/IMAP – the login will fail due to the extra verification step required.

App passwords serve as a workaround to this problem. An app password is a special, one-time generated password that bypasses MFA and grants access for "less secure apps" or legacy protocols. For example, Google will no longer allow direct username/password login for POP/IMAP if you have two-step verification enabled, but you can generate a 16-character app password to use in your email client. This lets you authenticate via POP3/IMAP even with MFA on, though it's essentially a form of basic authentication (just a different static password). Services like Warmy instruct Gmail users to enable two-factor auth and then create an app password specifically for the warm-up service to use.

The need for app passwords highlights a security caveat: by using them, you are granting third-party software access to your account with fewer protections. Warm-up platforms that connect to your mailbox via POP3/IMAP often require storing these credentials (or OAuth tokens) to read and send email on your behalf. This raises **security implications** – users must trust the platform to safeguard credentials, and providers sometimes consider app password access as “less secure” since it bypasses interactive verification. Additionally, some providers periodically tighten policies around basic auth. Notably, Microsoft disabled basic auth for Exchange Online (Office 365) in 2022, requiring modern OAuth for POP/IMAP access, and Google announced that starting in 2025 only OAuth would be allowed for Gmail API/IMAP access on Google Workspace. This trend means warm-up services have had to adapt by implementing OAuth flows or advising on special configurations.

For Warmy users, the takeaway is to use these legacy access methods carefully. If a warm-up tool asks for a direct password or to disable security settings, that’s a red flag – reputable services will use app passwords or OAuth, not your primary password. Ensure you generate unique app passwords and revoke them if you stop using the service. Recognize that while POP3/IMAP access is convenient (for example, to pull in forwarded emails or automate warm-up interactions), it doesn’t natively support modern security, so we rely on stopgap solutions like app passwords. Always follow provider guidelines (like Google’s or Microsoft’s) when enabling such access, to strike a balance between deliverability needs and account security.

Modern Authentication Protocols and Identity Infrastructure

To overcome the weaknesses of basic authentication, email providers have widely adopted **modern authentication protocols** like OAuth 2.0 for granting access to accounts. OAuth 2.0 allows an application (e.g., an email client or a warm-up service) to obtain a token to act on your behalf without ever handling your raw password, and it can be scoped to specific services (like "read and send email"). On top of OAuth, protocols like OpenID Connect (OIDC) provide an identity layer, enabling single sign-on via **Identity Providers (IdPs)** such as Google, Microsoft (Azure AD), Okta, etc. In practice, this means many corporate or cloud email accounts use SSO login flows – for example, signing into Office 365 might redirect you to your company's Okta login page, or signing into G Suite might use Google's common auth portal.

Email client integration with these modern IdPs has proven challenging. Traditional email clients (and automated services) expect to supply a username and password to authenticate. With OAuth/OIDC, the client must launch a web browser or embedded window for the user to complete a login and grant access. This is feasible for a user-driven mail app (like Outlook or Thunderbird prompting you once), but it's much harder to automate in a headless script or third-party service. Warm-up platforms that manage many inbox connections would ideally use OAuth tokens, but implementing and maintaining support for each provider's OAuth flow (Google, Microsoft, Yahoo, etc., each with distinct protocols and consent screens) is complex. Often, users of such platforms have to perform a one-time connection step: e.g. clicking "Connect with Google" which opens a web auth prompt. Some warm-up services have built OAuth integration to let users connect Gmail or O365 in a few clicks. Others rely on legacy IMAP with app passwords because it's simpler to implement, even if not as elegant.

Why do legacy methods persist? In short, inertia and practicality. There are many scenarios where OAuth isn't feasible: consider older devices or printers that send emails, command-line scripts, or small email servers – it's impractical for these to pop up a web login. Moreover, enterprise environments may require admin approval for any third-party app to use OAuth with corporate email, creating barriers. Because of this, big providers have kept legacy authentication available in some form. Google's stance illustrates this: they disabled "less secure app access" (basic auth with real password) but still allow app passwords or OAuth for IMAP/POP. They explicitly state that IMAP/POP aren't being turned off; instead, clients must connect using OAuth2 or similar secure methods. Microsoft similarly moved to OAuth but provided an **OAuth 2.0** mechanism for IMAP/POP in Exchange Online rather than killing those protocols.

For end users and warm-up services, this means a dual approach. Whenever possible, use modern auth – e.g., connect your mailbox by logging in via the provider's official OAuth screen, which yields a token the service can use. This token can often be limited or revoked without changing your main password. But if a provider or scenario forces basic auth (like some smaller email hosts that don't support OAuth), then use measures like unique app passwords and ensure the service handling them is trustworthy.

In summary, today's identity infrastructure is highly secure but sometimes inconvenient for integration. Warmy users might notice some accounts connect via a quick OAuth flow (modern and secure), while others require manual setup with server names and app passwords (legacy method). This is a reflection of the broader email ecosystem: Modern authentication is here to stay, but legacy access lingers on due to compatibility needs. The best practice is to favor OAuth2/OIDC connections when available and reserve basic auth for cases where it's absolutely necessary – and even then, use it with added precautions.

Forwarding, SPF, SRS, and DMARC

Email forwarding is a common practice – for example, you might forward all your *@work.com* mail to a personal *@gmail.com* address for convenience. However, forwarding can conflict with modern email authentication, especially SPF and DMARC, leading to deliverability problems. Understanding why requires a closer look at how SPF and DMARC work:

- **SPF (Sender Policy Framework)** allows a domain to specify which mail servers are authorized to send on its behalf. It checks the **envelope sender** (the MAIL FROM address used in SMTP, often visible as the Return-Path header) against the sending server's IP. If the IP isn't in the domain's SPF record, SPF fails. Crucially, SPF is tied to the SMTP session's origin. When an email is forwarded, the forwarder (e.g., your work mail server) connects to the final destination (e.g., Gmail) and reuses the original sender's address in the MAIL FROM by default. Gmail sees an email claiming to be from @work.com coming from an IP that likely isn't listed in http://work.com's SPF record (because it's actually coming from the forwarder's server). As a result, SPF checks fail for most forwarded mail. This is expected: the message is coming through an intermediary not in the original sender's SPF policy.

- **DMARC (Domain-Based Message Authentication, Reporting, and Conformance)** builds on SPF and DKIM, requiring that the domain in the visible From address aligns with the domain authenticated by SPF or DKIM. Domain owners can publish a DMARC policy (like p=quarantine or p=reject) telling receivers to discard or flag emails that fail this alignment. A strict DMARC policy (especially p=reject) means that if both SPF and DKIM appear invalid or misaligned, the email should be rejected. In a forwarding scenario, if the original sender's domain has DMARC enforcement, the forwarded message often ends up with **SPF failed** (as explained) and possibly **DKIM pass** (more on DKIM shortly). If DKIM fails or isn't present, then DMARC will fail since SPF failed and there's no aligned DKIM – at which point the receiver might reject the mail based on the sender's DMARC policy. This is exactly what happened historically with some domains: for instance, AOL and Yahoo introduced p=reject policies, which caused forwarded emails from those domains to be dropped by Gmail and others, interpreting them as spoofed messages.

To mitigate SPF breaking during forwarding, the industry developed **SRS (Sender Rewriting Scheme)**. SRS is a technique where a forwarding mail server modifies the envelope sender address to a new address at the forwarder's domain. For example, if an email from *alice@bank.com* is being forwarded by *http://forwarder.com*, the forwarder could change the MAIL FROM to something like *SRS0=alice=bank.com@forwarder.com*. Now Gmail will check SPF on *http://forwarder.com*, and since it's *http://forwarder.com*'s own server sending, that SPF will pass. This preserves the ability to deliver the message (avoiding outright SPF-based rejection). However, SRS introduces a new wrinkle with DMARC: the visible From is still *@bank.com*, but the SPF authentication domain is now *http://forwarder.com* (due to rewriting). DMARC alignment will see that SPF authenticated *http://forwarder.com*, which **does not align** with the From domain *http://bank.com*, so DMARC would still fail. The hope in these cases is that **DKIM** can save the day (or that receivers apply local policy to soften DMARC enforcement for forwarded mail, as many do). It's noted that using SRS will intentionally fail DMARC's SPF alignment, so you must rely on DKIM for DMARC to pass. In practice, many forwarding services implement SRS (e.g., Office 365 now includes SRS for forwarded mails), and receivers often end up using DKIM or ARC (Authenticated Received Chain) to evaluate the mail. ARC is an emerging standard that forwarders can use to attest the authenticity of the original message even after forwarding, but that's beyond our scope here.

What if a forwarder doesn't implement SRS? Then the forwarded mail arrives with the original Return-Path. SPF will fail (as discussed), and DMARC will rely on DKIM. If DKIM passes and is aligned, DMARC can still pass; if not, DMARC fails, and the mail may be rejected. For users and warm-up platforms, this is tricky. If you have a strict DMARC policy on your domain (p=reject) and you send emails to recipients who forward their mail, some of your messages could bounce or be dropped by the final recipient. On the other side, if you are forwarding mail (say, you forward from one of your addresses into the Warmy system or another account), forwarding without SRS can break SPF and risk rejection.

One solution that many have used is to avoid forwarding entirely and use **POP3 fetching instead**. For example, rather than auto-forwarding your work email to Gmail (and breaking SPF), you can have Gmail periodically pull messages from your work account via POP3. In that case, Gmail acts as an email client: it logs in and retrieves the messages, which are delivered to your Gmail inbox as if they were local mail. This approach preserves original authentication results because the message isn't being re-mailed through an intermediate SMTP hop; Gmail is directly retrieving the original email from the source. In the context of warm-up platforms, a similar strategy can be used: instead of forwarding warm-up emails around (which would confuse SPF/DMARC), the platform could retrieve them from each mailbox via POP/IMAP. The security downside, as discussed, is that it requires sharing login credentials with the platform. But deliverability-wise, it's effective. In fact, experts explicitly recommend this method in tricky cases – e.g., to reliably get mail delivered to Gmail without DMARC issues, "use Gmail's POP retrieval method to pull email … rather than forwarding it.". The caveat is that POP3 fetching might introduce slight delays (emails are fetched at intervals) and requires configuration. Nevertheless, it bypasses the forwarding problem entirely by eliminating that extra SMTP hop.

In summary, forwarding can disrupt SPF and DMARC, but solutions exist:

- Implement proper **SRS** on any forwarding addresses you control (or choose providers that do so) to fix SPF, and rely on DKIM to handle DMARC alignment.

- Ensure your domain's **DKIM** is set up, so that even if messages are forwarded, there's a good chance they remain authenticated via DKIM.

- Be cautious with strict **DMARC** (p=reject) if a significant portion of your mail might be forwarded. Monitor DMARC reports to see if legitimate forwards are failing. In some cases, a *p=quarantine* might be more practical than *reject* to allow forwarded mails to still reach spam folders rather than being outright rejected.

- As an alternative to forwarding, use **direct retrieval** (POP3/IMAP) for collecting messages across accounts, accepting the trade-off of a more complex setup for better alignment with authentication.

For Warmy users, understanding this means if you set up any forwarding rules or if Warmy instructs you to configure mailbox forwarding, be aware of the SPF/DMARC implications. Warm-up emails that traverse multiple mail systems need careful handling; otherwise, ironically, your warm-up messages could start bouncing due to authentication misalignment. Warmy's platform may instead ask for IMAP access to each inbox precisely to avoid these forwarding problems, ensuring all interactions are direct and preserving authentication.

DKIM in Authentication and Forwarding Contexts

DKIM (DomainKeys Identified Mail) is another pillar of email authentication, and it plays a crucial role, especially when emails are forwarded. DKIM works by having the sending mail server attach a digital signature to the headers and body of the email. The signature is associated with a domain (the d= value in the DKIM-Signature header) and can be verified by receivers using the public key published in that domain's DNS. If the email's content and headers haven't been altered, the DKIM check at the receiver will pass, proving the email was indeed sent by (or on behalf of) that domain and hasn't been tampered with.

One major advantage of DKIM is that it **survives forwarding** in most cases. Because the signature travels with the email as a header, an intermediate forwarder does not need to re-sign the email. As long as the forwarder doesn't modify the signed portions (which typically include the message body and certain headers like Subject, but not the Received headers it adds), the DKIM signature remains valid at the final destination. This means that even if SPF fails due to forwarding, the original sender's DKIM can still pass, and if that DKIM's domain matches the From address domain, DMARC can pass via DKIM alignment. For example, if *test@example.com* sends an email with a DKIM signature from *example.com*, and it's forwarded, the recipient might see SPF fail (because of the forwarder) but DKIM still pass for *example.com*, thus satisfying DMARC in the DKIM-alignment mode.

In practice, mailbox providers and spam filters heavily favor DKIM for this reason. It's more reliable across complex routing. Many email service providers advise: always set up DKIM on your sending domain to ensure mail authentication is robust, especially for forwarded mail.

From a deliverability and spam scoring perspective, a valid DKIM signature (especially one that is **author domain aligned**, meaning the DKIM domain matches the sender's From domain) is a positive signal. Some spam filtering systems (like SpamAssassin used by many mail servers) award small negative scores (which are good, since in spam scoring negative points indicate less spam likelihood) for emails that have valid DKIM. Specifically, SpamAssassin rules like *DKIM_VALID* and *DKIM_VALID_AU* are triggered when an email has a valid DKIM signature, and the latter when the signing domain matches the author's domain. For instance, SpamAssassin's default scores might subtract on the order of 0.1 points for *DKIM_VALID_AU* (meaning the message is slightly less spammy) and even more for a combination of factors like passing DMARC with a strong policy. An example SpamAssassin report might include lines such as: *DKIM_VALID_AU = -0.1* and *DMARC_PASS_REJECT = -1.2*, indicating the presence of a valid aligned DKIM and a passing DMARC (with a domain policy of reject) significantly improved the trust score. These values are relatively small in isolation, but they can make the difference in borderline cases – and they reflect general industry trust: a message that is signed by the purported sending domain is less likely to be impersonation or spam.

However, it’s worth noting DKIM’s nuances:

- If the forwarder modifies the email in a way that breaks the signature (e.g., some mailing lists or spam filters might alter the content, like adding a footer or modifying HTML), then DKIM will fail at the final recipient. This is why sometimes you might see forwarded mails failing DKIM – not because DKIM is unreliable, but because something changed in transit (even something as small as an extra space in a signed header can invalidate it).
- DKIM alone doesn’t guarantee the email isn’t spam (we discuss this more in the next section). It only assures the identity of the sender domain and content integrity. So a spammer can send email with a perfectly valid DKIM signature of their own domain – and the spam filter will see DKIM pass. It might give a slight trust for the identity, but other filters (content, reputation, etc.) still apply. In other words, DKIM tells the receiver “this email really comes from who it claims (domain X) and wasn’t altered,” but it doesn’t say whether domain X is trustworthy or the content is good.

One specific term users might encounter in email headers is “***DKIM_VALID_AU***” (or similar). This typically shows up in diagnostic headers or spam filter reports to denote that a DKIM check was valid for the author’s domain (AU = Author). It’s essentially an internal code indicating aligned DKIM success. Seeing this in a message header (for example, some mail systems add an X-Spam-Status header listing tests) is a good sign – it usually correlates with the message having a better chance to land in the inbox.

For Warmy and warm-up scenarios, ensuring DKIM is correctly set up on all sending domains is critical. Since warm-up emails often involve multiple hops (between sending and receiving accounts in the warm-up network), having DKIM in place means those messages will carry authentication through any intermediate hops. Warm-up services typically guide users to add DKIM records to their DNS and verify they're working. As the warm-up emails circulate, you'll want to see that all are DKIM-signed by your domain. This consistency not only helps with immediate delivery (avoiding spam folders) but also trains mailbox providers that your domain signs its mail – a sign of a responsible sender. Keep an eye on any reports or logs: if you see DKIM failures for your domain's emails in the warm-up, something is misconfigured (wrong DNS record, or your sending service not signing properly). When everything is done right, DKIM will be your steadfast ally in maintaining authentication where SPF cannot and in boosting your domain's reputation during the warm-up process.

Limitations of Email Authentication Mechanisms

It’s important to understand what **email authentication mechanisms** like SPF, DKIM, and DMARC do and what they don’t do. These standards were created to tackle specific problems – mainly spoofing of sender identity – but they are not silver bullets for stopping all unwanted email or spam. In other words, think of them as authentication tools, not comprehensive authorization or spam content filters.

Authentication vs. Authorization: In security terms, authentication is about verifying identity (“Is this email truly from the domain it claims?”), whereas authorization is about granting permission or evaluating intent (“Should this email be allowed in my inbox?”). SPF, DKIM, and DMARC are authentication protocols; they make technical assertions about the identity of senders. They do not make value judgments about the content or whether the sender is “good” or “bad” beyond identity. They also don’t directly block an actor from sending email – any domain owner can publish SPF/DKIM and thereby authenticate their emails. There’s no central authority saying “this domain isn’t allowed to send mail”; the protocols are decentralized.

Consequently, these mechanisms do not stop all spam. **Why can spam still get through despite SPF/DKIM/DMARC?** Because spammers can adapt and use them too. For example, a spammer can buy a new domain (say, spammydomain.com), set up valid SPF records and DKIM keys for it, and send out spam from that domain. Technically, those emails will pass SPF and DKIM checks (and even DMARC if set up), because the spammer is the legitimate owner of the domain they’re sending from. As one email admin noted, “over 20% of incoming spam mails pass DKIM” on their server. The spammer isn’t spoofing someone else – they’re sending from their own throwaway domain with proper authentication. The email authentication system will confirm the mail is indeed from spammydomain.com and intact, but it cannot decide that spammydomain.com is malicious; that falls to reputation systems and content filters.

The original **intent behind SPF and DKIM** was to curb phishing and spoofing, not to eliminate spam. SPF was first proposed in the early 2000s to stop **forged sender addresses** – at that time, a common spam tactic was to fake the "From" or return-path to appear as someone else (sometimes to bypass filters or to trick recipients).

SPF lets domain owners declare which IPs are legit for their domain, so receivers can spot fakes.

DKIM (born from Yahoo's DomainKeys and Cisco's Identified Mail, later standardized around 2007) was about ensuring the email isn't altered and genuinely comes from the domain's owners. It was particularly aimed at preventing attackers from impersonating trusted domains or tampering with messages.

DMARC, introduced around 2012, tied these together to give domain owners a way to enforce authentication – basically, "if you get an email from my domain that fails SPF/DKIM alignment, please don't accept it." This was largely motivated by major senders (like banks and e-commerce sites) wanting to block phishers from sending emails appearing to be from them. DMARC provides reporting too, so domain owners can see who's sending mail purporting to be from them and how often authentication passes or fails.

An analogy: SPF/DKIM/DMARC are like an ID check at a club's door – they ensure the ID is real and matches the person (email) presenting it. But having a real ID doesn't guarantee the person isn't trouble – it just means they are truthfully representing who they are. It's up to the club (mail server) to decide if that person is welcome, perhaps based on past behavior or other cues.

Let’s highlight a few **limitations** and corner cases:

- **SPF’s scope**: SPF authenticates the path (IP address) but not the header From address that users see. This means without DMARC, a phisher could send from an IP authorized by their own domain (passing SPF for that domain) while putting a different brand’s email in the From header, potentially tricking users. This is why DMARC’s alignment requirement is important – it forces the visible From to be aligned with the authenticated domain.

- **DKIM’s weakness**: DKIM signatures can be replayed. If a spammer gets hold of a legitimately signed email, they could resend that same message to many recipients and the DKIM would still appear valid (since it’s the original signature). This is a known limitation – DKIM doesn’t bind a signature to a particular recipient or time. Hence, spam filters may track how often identical DKIM signatures appear to catch replay spam. Also, as mentioned, a spammer can have their own domain with DKIM, so DKIM by itself doesn’t equate to “not spam”

- **New domains and reputation**: DMARC can backfire for new domains if misconfigured. If you set p=reject on day one for a brand new domain that no one’s seen before, your emails might get rejected or dropped during warm-up if something goes wrong (and you’ll get reports). It’s generally advised to start with p=none (just monitoring) and move to quarantine/reject once you’re sure all your mail streams are authenticating properly.

- **Third-party senders**: If you use a marketing platform or a service to send emails on your behalf, SPF/DKIM/DMARC need special care. SPF would require including the third-party’s servers in your record, and DKIM would require them to sign with your domain’s key (or you can use a subdomain for their mail). Without that coordination, your DMARC policy could cause their emails (which are legitimate, but coming from their servers) to be rejected. This is why many email platforms have you add DNS records for DKIM and sometimes include an include in SPF.

The bottom line is that these standards **significantly help** in reducing certain kinds of abuse (especially phishing and direct domain spoofing), but they do not directly address spam content or sender reputation. Spammers have adapted: instead of spoofing well-known domains that now have DMARC, they more commonly register look-alike domains or use domains with no reputation. Spam and phishing have not disappeared; the battle simply moved. As one commentator quipped, the landscape sometimes feels like “patch over patch over patch (SPF, DKIM now ARC)” – implying that we keep adding layers to address new weaknesses. Indeed, ARC was introduced to handle the forwarding problem DMARC created, and future tweaks will arise as needed.

For Warmy users and anyone managing email campaigns, this means you **must not assume** that just because you set up SPF/DKIM/DMARC, your mail will be inboxed. Those are table stakes – they prevent you from being outright rejected for failing authentication and protect your brand, but you still need to mind your sending behavior, list quality, content, and engagement. Authentication is the first step in a much larger deliverability picture. It ensures you’re identifiable and accountable. After that, it’s your **reputation** (which is built by following best practices over time) that determines inbox placement. Conversely, if you neglect authentication, you’re almost guaranteed to have poor results – many receivers might reject or flag your mail as untrusted, and you leave yourself open to spoofing.

In summary, SPF, DKIM, and DMARC are indispensable tools – use them to authenticate your emails and assert your identity. But remember their purpose: they tell the world who sent the email and whether it’s legitimate, not whether it’s desirable. The final judgment on delivering to inbox vs. spam involves additional filters and rules outside the scope of these protocols.

Additional Techniques for Reducing Spam and Improving Reputation

Beyond SPF, DKIM, and DMARC, mail administrators use a variety of techniques to reduce spam and ensure only legitimate senders get through. While these methods are more on the receiving side, knowing about them can help senders (including those warming up inboxes) understand how to behave to avoid pitfalls. Here are a couple of notable techniques and their relevance to deliverability and warm-up:

- **DNS resolvability enforcement**: Many mail servers check the DNS configuration of the sending domain and IP as a rudimentary legitimacy test. This often includes verifying that the sending server's IP address has a **reverse DNS (PTR record)** that maps to a domain, and that this domain in turn resolves back to the same IP (forward-confirmed reverse DNS). Lack of a proper reverse DNS is a common hallmark of spam sources (e.g., a compromised machine on a consumer ISP might have a generic or missing rDNS). It's common practice for receivers to penalize or even reject mail from IPs without rDNS or with mismatched rDNS. Additionally, some receivers check that the **HELO/EHLO domain** given by the sending server is a real domain that resolves, and that the **envelope sender domain** (MAIL FROM domain) has an MX or A record. If the MAIL FROM domain doesn't exist in DNS, some servers will flat-out reject the email as it's likely forged – after all, if a bounce needs to be sent, there's nowhere to send it if the domain is bogus. All these checks essentially ensure the sender's DNS is in order. For senders, the implication is: configure your DNS properly. Ensure your sending IP has a PTR record pointing to a plausible hostname (ideally one that points back to that IP). Make sure any domain you send from exists in DNS and ideally has an MX record (or at least an A record). These are basic "hygiene" factors for a reputable mail setup. If you're using an ESP or a warm-up service's network, they typically handle PTR records on their infrastructure, but if you manage your own mail server or dedicated IP, don't overlook this. It's also wise to avoid sending from new domains that have no DNS footprint besides just being registered; receivers might do a quick DNS lookup and if the domain was registered yesterday and has minimal DNS records, it could raise suspicion.

SMTP transaction delays (tarpitting/greeting delay): A clever trick in the anti-spam arsenal is to exploit the impatience of spam bots. Legitimate mail servers follow the SMTP protocol diligently, waiting as needed for responses. Some spam-sending software ("ratware") tries to send high volumes and doesn't wait around – it might start firing commands without proper handshakes. Techniques like **greet delay** intentionally make the receiving server pause before sending the initial "220 Welcome" banner or before acknowledging certain commands. For example, an SMTP server might wait 5-20 seconds after a client connects before saying anything. A well-behaved sender will simply wait (the SMTP RFCs say clients should wait up to several minutes for a reply). But a spam bot might start spewing commands early; if it does, the server can drop the connection because the client isn't protocol-compliant. Another tactic is to accept the connection but deliberately slow down. This slows the throughput of spam senders dramatically – they might give up and disconnect, or send fewer messages because each one takes longer, whereas a legitimate sender will patiently retry or continue as needed. Some systems even do a hybrid approach: only impose these delays if the connecting IP or domain looks suspicious or hasn't been seen before (to avoid slowing down all mail). There's also **greylisting,** a form of delaying at the message level: the server temporarily rejects the first attempt from an unknown sender with a 4xx error, under the logic that a real mail server will attempt delivery again after a few minutes, whereas many spam bots won't. Greylisting can be very effective against simplistic spam software, though it introduces delays in receiving mail and can be less useful against sophisticated spammers who now retry.

How do these affect **inbox warming and deliverability**? As a sender warming up a domain/IP, you should be prepared for some of your messages (especially early on) to face such delays or tests. For example, if your warm-up emails from a new IP hit a mail server that greylists unknown senders, your first emails might be deferred. A warm-up service or your own sending system should be configured to handle retries properly – ensure that if a message is deferred ("temporary failure"), it will try again later. Warm-up schedules often send low volume, which is good, but make sure they also account for potential delays (e.g., not giving up too soon if an email isn't accepted on first try).

Also, ensure your sending infrastructure doesn't itself misbehave. If a receiving server is using a greet delay and your sending software doesn't wait for the "220" and instead times out too quickly or starts transmitting, you'll get disconnected. Most reputable sending systems handle this fine, but it's a consideration if you built a custom mail script. Essentially, follow the SMTP protocol strictly – doing so means you'll pass invisible tests that weed out bots.

Another technique related to DNS is **enforcing resolvable domains in headers**. Some spam filters check URLs in the message or domains in the From header to see if they resolve. If you're mentioning a domain or using one that doesn't exist, that's a red flag. Warm-up emails usually contain plain, non-commercial content, so this is less of a concern there, but when you move on to actual mailing, ensure links and domains are valid.

One more point: **feedback loops and spam complaints**. ISPs like Google and Microsoft use engagement (or lack thereof) as a huge factor. While not a "technique" like the above, it's worth noting that if users consistently ignore or mark your emails as spam, your reputation suffers. Warm-up tries to counteract this by generating positive engagement signals (opens, replies from friendly addresses, etc.), effectively training the algorithms that your mails are wanted. This isn't foolproof, but it helps establish a baseline reputation. Always complement warm-up with good sending practices: mail people who expect you, avoid sudden spikes in volume, and send content that doesn't trigger content-based spam filters (excessive images, spammy phrases, etc.).

In summary, receiving servers deploy a **layered defense**: first checking technical authenticity (DNS, SPF, DKIM), then using connection-level tricks (rDNS checks, delays, greylisting), then content and reputation analysis. As a sender focused on deliverability:

- Make sure your DNS and server setup won't fail basic checks (PTR records, matching HELO, etc.).
- Be patient and consistent – if some messages are delayed, continue with the warm-up; these delays often ease once your sending pattern is recognized as legitimate.
- Understand that these defenses mean a cold sender must earn trust gradually. Early emails might be lightly throttled or tested; as you build a clean track record, the path clears.

For Warmy users, luckily much of this low-level detail is handled behind the scenes by Warmy's network and guidelines. Your job is mainly to follow the setup steps (SPF, DKIM, etc.), and let the warm-up process proceed steadily. If you do everything right, over time your emails will no longer trigger the speed bumps in mail delivery, and you'll see faster delivery and inbox placement.

Conclusion and Practical Recommendations for Warmy Users

In conclusion, mastering email deliverability is a journey that combines **technical configuration** with **reputation management**. Modern email authentication standards – SPF, DKIM, and DMARC – form the bedrock of trust, and they must be correctly in place for any warm-up or campaign to succeed. Equally important is understanding their limits and the surrounding ecosystem of email security.

For users of Warmy, here are the key takeaways and best practices derived from the topics we've covered:

- **Always authenticate your sending domain**: Before sending any volume, set up SPF and DKIM for your domain and ensure they are valid. Warm-up emails should consistently pass SPF and DKIM checks. This not only prevents outright failures and spam flags but also contributes positively to spam filter scoring. Monitor your authentication results (Warmy may provide dashboards, or use external tools) to catch issues early. Implement DMARC in monitoring mode (p=none) at first, and once you're confident, consider enforcing it (quarantine/reject) to protect your brand – but be mindful of forwarding scenarios as discussed.

- **Use warm-up to build a positive sender reputation gradually**: Warmy will send emails on your behalf in increasing numbers and even interact with them (open, reply) via its network. This process is vital for a new domain or IP. Resist the temptation to rush. During this warm-up phase, focus on consistency and hygiene: do not send other large campaigns from the domain until warm-up is done, and keep the content of warm-up emails benign (Warmy handles this with generic content). The goal is to appear to mailbox providers as a real, small-scale sender at first, then gradually a larger one – never an overnight sudden bulk sender. This acclimatization avoids triggering volume-based throttling or suspicion.

- **Be cautious with legacy protocols and access**: If Warmy or your use case requires connecting your inbox via IMAP/POP, prefer secure methods. Enable 2FA on your accounts and use app passwords or OAuth flows to grant access. This keeps your main credentials safe. Remember that Google and Microsoft are phasing out basic auth – if an OAuth option is available for connecting (e.g., "Sign in with Google" in Warmy), use it. If you must supply an app password, treat it like a secret key: you can revoke it anytime from your account security settings. And once warm-up is complete or if you stop using the service, revoke unnecessary app passwords to close that access path.

- **Understand the impact of email forwarding**: If you employ any forwarding (say, you forward emails between addresses as part of your warm-up circle, or forward your Warmy emails to a central monitor), be aware of SPF and DMARC implications. Wherever possible, use direct retrieval (POP3/IMAP) or ensure the forwarder uses SRS. This can prevent issues where warm-up emails might be accidentally rejected by strict filters. The simple rule: an email that is forwarded should either be DKIM-signed (so it survives) or have SRS applied if SPF needs to pass. If neither is true, don't be surprised by sporadic bounces. For Warmy's internal operations, trust their guidance – they likely have mechanisms to deal with this (many warm-up platforms instruct on how to avoid forwarding problems, perhaps by having you connect all accounts directly).

- **Keep an eye on metrics and reports**: Utilize DMARC aggregate reports (via a service or email address you set in the DMARC record) to see if any sources are failing authentication. This can reveal if, for instance, some forwarder is causing issues or if someone is spoofing your domain. Also, watch Warmy's analytics or email provider's feedback: if any warm-up message did land in spam or got a spam complaint, adjust accordingly (though warm-up networks usually auto-handle remediation by increasing positive engagement to counteract).

- **Gradually transition to real sending, and follow best practices**: After a warm-up period, your domain/IP will have an initial good reputation. From there, as you start sending actual emails (newsletters, outreach, etc.), do it gradually. Don't go from sending 50 warm emails a day to 50,000 marketing emails the next – ramp up to your desired volumes over days or weeks, so ISPs see continuity. Make sure your email lists are opt-in and cleaned (high bounce or complaint rates can undo all the warm-up goodwill quickly). Continue to follow content best practices (avoid all-caps, excessive punctuation, misleading subjects, etc.), even though content was not the focus of this report, it's the next piece of the puzzle for staying in the inbox.

- **Stay informed on evolving standards**: Email security is continuously evolving. For example, **ARC** is being adopted to help with forwarded mail and preserve authentication through hops. While you as a sender don't need to implement ARC (it's mostly on receivers/forwarders), being aware of it is good. Likewise, new authentication-related policies (like BIMI for brand logos) are emerging – they don't directly affect deliverability like SPF/DKIM, but they are part of the trust landscape. Keep an eye on announcements from major providers: if Google says "we now require OAuth for all connections", ensure your processes adapt. Warm-up services will also evolve to accommodate these changes (for instance, by updating their connection methods).

- **Security and reputation go hand in hand**: It might be tempting to find shortcuts (like disabling security features to make automation easier), but in email, **cheating security usually backfires on deliverability**. For example, turning off MFA and allowing less-secure access might simplify connecting a device, but it exposes your account and might even lower trust (Google flags accounts that don't meet security standards). It's better to work within the security frameworks – use the proper channels (OAuth, app passwords) – so that your sending behavior is aligned with what providers expect from reputable senders.

By following these recommendations, Warmy users can navigate the modern email landscape confidently. You will be leveraging warm-up to its fullest effect: not only improving inbox placement through engagement but also demonstrating to ISPs that you are a responsible, authenticated, and legitimate sender. Warm-up is as much about educating the sender as conditioning the receiver – and now that you're equipped with knowledge about POP3 vs. OAuth, forwarding challenges, and how filters work, you can make more informed decisions in your email strategy.

In the end, successful email deliverability comes down to earning trust. Every protocol you implement (SPF/DKIM), every warm-up email sent, every reply or open, and every clean list you maintain – all these are signals that build trust. With a solid foundation of authentication and a smart warm-up regimen, you're well on your way to seeing your emails land where they should: in the inbox, ready to be read. Good luck, and happy sending!

warmy

Auto All-In-One Tool For Email Deliverability To Make Your Email Channel Reliable

We are passionate about solving email deliverability challenges and making email a reliable channel for every business

325+	9 countries	95+
Years Of Combined Email Deliverability Expertise	Home To Our Talented Team	Countries Have Daily Active Users In Warmy