Group: Essential Group	Report Number: Report No14	Report id 14-lec-27,28&29(spf,dkim&dmark)-12-essential

EMAIL SECURITY

Prepared By:
Kazim Ali Obad

Supervisor:
Anmar Mohammed

Date of Task Assignment :

3/29/2026

Due Date:

4/11/2026

Contents

Scenario

Essential Level: Basic Email Authentication Discovery

Scenario Title: "Is This Domain Protected?"

Objective: Perform passive reconnaissance to verify the implementation of SPF, DKIM, and DMARC security protocols.

Environment & Tools

- Operating System: Kali Linux
- Network: Active Internet access
- Primary Tools: dig (CLI), MXToolbox (Web)

Execution Flow

Step	Action	Command
1	Query SPF	dig TXT <domain> | grep spf
2	Verify SPF	Check for v=spf1 … and note the ~all or -all policy
3	Query DMARC	dig TXT _dmarc.<domain>
4	Verify DMARC	Identify p=none / p=quarantine / p=reject
5	Query DKIM	dig TXT <selector>._domainkey.<domain>
6	Verify DKIM	Check for a valid public key in the p= field

1. Summary

This report presents a full email security posture assessment conducted on three major public domains google.com, microsoft.com and Facebook.com .

The assessment was performed on Kali Linux using the dig DNS query tool, following the methodology of a real world CISO security audit.

The three pillars of email authentication SPF (Sender Policy Framework), DKIM (DomainKeys Identified Mail), and DMARC (Domain-based Message Authentication, Reporting & Conformance) were each analysed in sequence. Findings were then correlated to determine overall spoofing risk and to identify the weakest link in each domain's defence posture.

Domain	SPF Policy	DMARC Policy	DKIM Result	Overall Risk
google.com	~all (SoftFail)	p=reject	NXDOMAIN — no public selector	LOW
microsoft.com	−all (HardFail)	p=reject, pct=100, fo=1	CNAME delegation (selector1 → M365)	LOW
facebook.com	redirect=_spf.facebook.com	p=reject, pct=100	default selector: key revoked (p=;)	LOW–MEDIUM
amazon.com	−all (HardFail)	p=quarantine, pct=100	NXDOMAIN (selector1)	MEDIUM
netflix.com	−all (HardFail)	p=reject, fo=1	selector1 returns SPF record (misconfigured)	LOW
x.com	−all (HardFail)	p=reject	NXDOMAIN (selector1)	LOW
paypal.com	~all (SoftFail)	p=reject	NXDOMAIN (selector1)	LOW
shopify.com	~all (SoftFail)	p=reject, pct=100, fo=1	NXDOMAIN (selector1)	LOW
dropbox.com	~all (SoftFail)	p=reject, pct=100	NXDOMAIN (selector1)	LOW
github.com	~all (SoftFail)	p=quarantine, sp=reject	CNAME delegation (selector1 → M365)	LOW

2. Methodology & Tools

All reconnaissance was performed passively via DNS queries no traffic was sent to target mail servers at any point. This approach is fully passive and legal under the targets' public bug bounty programmes and the fact that DNS records are publicly visible to any resolver on the internet.

dig (Domain Information Groper)

The primary tool used throughout this assessment. Installed by default on Kali Linux. Provides verbose, structured output with HEADER, QUESTION, ANSWER, and AUTHORITY sections.

Standard dig query patterns used:

```
dig TXT <domain> | grep spf            # Query SPF record
dig TXT _spf.<domain>                  # Follow SPF include chain
dig TXT _dmarc.<domain>                # Query DMARC policy
dig TXT <selector>._domainkey.<domain>  # Query DKIM public key
```

The dig output contains several important sections that were read during this exercise:

3. Domains Analysis

1. Google (google.com)

SPF Sender Policy Framework

What Is SPF and Why Does It Matter?

SPF is a DNS TXT record that defines which mail servers are authorised to send email on behalf of a domain. When a receiving mail server gets a message, it queries DNS to check whether the sending server's IP address is listed in the SPF record. If the IP is not listed, the receiver can choose to reject, flag, or accept the message depending on the SPF policy.

SPF prevents a class of attacks known as direct-domain spoofing, where an attacker uses a fake "From:" address matching the target organisation's domain. Without SPF, any server on the internet can claim to send mail from @google.com or @microsoft.com.

SPF has a hard-coded limit of 10 DNS lookups per evaluation. This limit exists because each lookup adds latency to mail delivery. If a domain's SPF record triggers more than 10 lookups, the SPF evaluation results in a "PermError" a permanent error that some receivers treat as a failure, silently breaking email authentication.

```
$ dig TXT google.com | grep spf
$ dig TXT _spf.google.com
$ dig TXT _netblocks.google.com
$ dig TXT _netblocks2.google.com
$ dig TXT _netblocks3.google.com
```

Figure 1: Returns v=spf1 include:_spf.google.com ~all. SPF is published with a SoftFail (~all) catch-all policy.

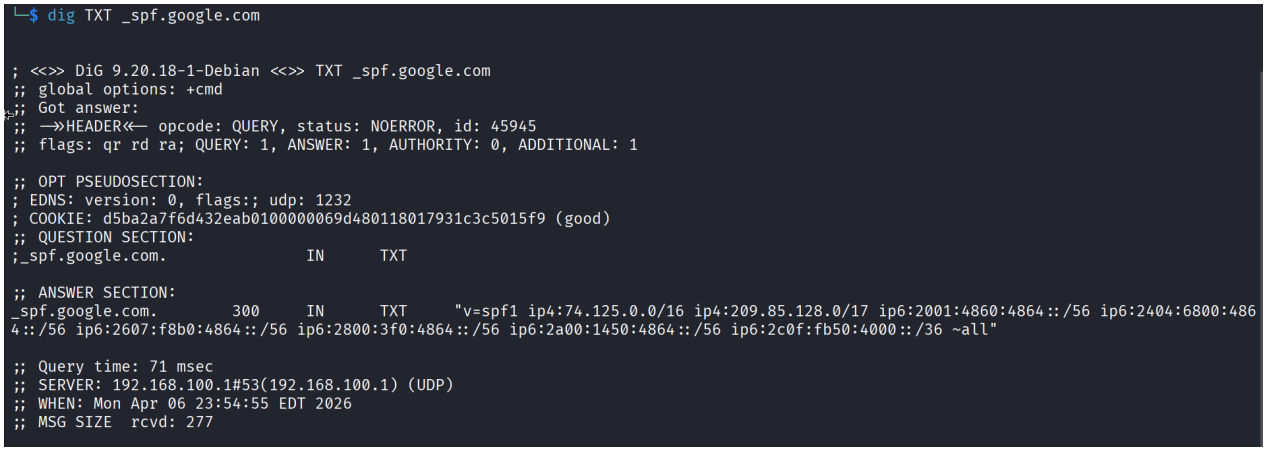

Figure 2: returns a large v=spf1 record with ip4:/ip6: ranges plus include: references. Direct IP ranges do not count toward the 10-lookup limit.

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _netblocks.google.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _netblocks.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44823
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: caa660b0a5ee508b0100000069d480a9792ad8666e7b48a1 (good)
;; QUESTION SECTION:
;_netblocks.google.com.         IN      TXT

;; ANSWER SECTION:
_netblocks.google.com.  300     IN      TXT     "v=spf1 ip4:74.125.0.0/16 ip4:209.85.128.0/17 ~all"

;; Query time: 75 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Mon Apr 06 23:57:26 EDT 2026
;; MSG SIZE  rcvd: 140
```

Figure 3: v=spf1 ip4:74.125.0.0/16 ip4:209.85.128.0/17 ~all. Google's primary IPv4 sending ranges.

```
└─$ dig TXT _netblocks2.google.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _netblocks2.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6344
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 8240ea896a5c598a0100000069d480e45240bf686006c2b6 (good)
;; QUESTION SECTION:
;_netblocks2.google.com.               IN      TXT

;; ANSWER SECTION:
_netblocks2.google.com. 300     IN      TXT     "v=spf1 ip6:2001:4860:4000::/36 ip6:2404:6800:4000::/36 ip6:2607:f8b0:4000::/36 ip6:280
0:3f0:4000::/36 ip6:2a00:1450:4000::/36 ip6:2c0f:fb50:4000::/36 ~all"

;; Query time: 75 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Mon Apr 06 23:58:26 EDT 2026
;; MSG SIZE  rcvd: 246
```

Figure 4: exclusively IPv6 ranges (2001:4860::/36, 2404:6800::/36, 2607:f8b0::/36, etc.). All IPv6 sending infrastructure.

```
└─$ dig TXT _netblocks3.google.com



; <<>> DiG 9.20.18-1-Debian <<>> TXT _netblocks3.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64705
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 4fbe14ee562c64790100000069d483566a371780ce929b0c (good)
;; QUESTION SECTION:
;_netblocks3.google.com.                IN      TXT

;; ANSWER SECTION:
_netblocks3.google.com. 300     IN      TXT     "v=spf1 ~all"

;; Query time: 87 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 00:08:54 EDT 2026
;; MSG SIZE  rcvd: 103
```

Figure 5: returns only "v=spf1 ~all" — the terminal record in Google's SPF chain. No further lookups triggered. Total: 4 out of 10 DNS lookups consumed.

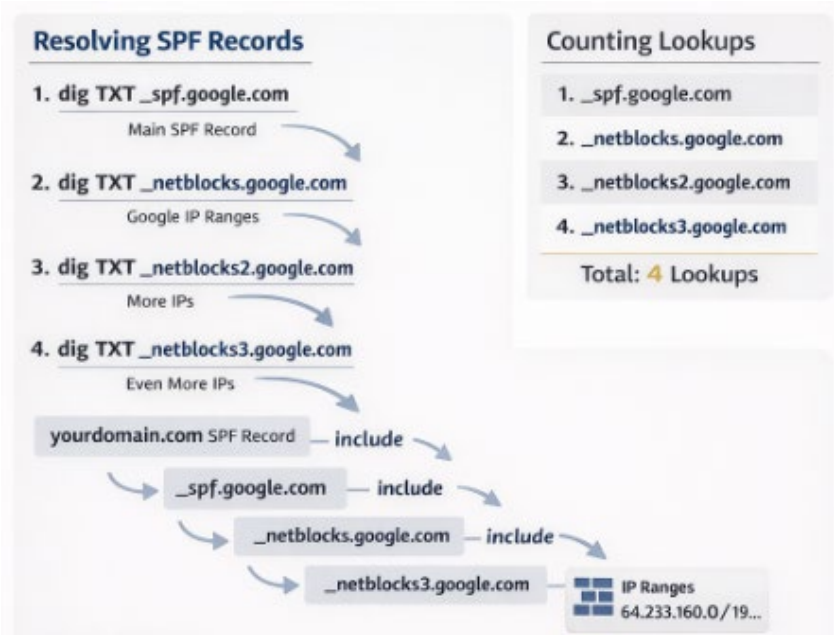

Figure 6: SPF lookup chain diagram — Shows the four-level resolution tree

Analysis:

- **v=spf1** — Version identifier. Tells receivers this is an SPF record.
- **include:_spf.google.com** — Points to Google's authorised IP ranges in a separate record for easier management.
- **~all (SoftFail)** — Mail from unlisted sources is accepted but flagged. Less strict than -all (HardFail). DMARC p=reject compensates for this.
- SPF chain resolves in 4 lookups — well within the 10-lookup limit.

The all mechanism is the final rule in every SPF record. It acts as the default policy for any sending source not explicitly listed. Its prefix character determines how strictly unauthorised senders are handled:

Mechanism	Meaning	Receiver Action	Risk Level
+all	Pass anyone can send	Accept all mail regardless of source	CRITICAL
~all	SoftFail mark as suspicious	Accept but flag with a spam score penalty	MEDIUM
-all	HardFail reject unauthorised	Reject mail from unlisted sources	BEST PRACTICE
?all	Neutral — no opinion	No policy enforced, treat as unverified	HIGH

DMARC (Domain-based Message Authentication)

What Is DMARC and Why Does It Matter?

DMARC is the enforcement and reporting layer that ties SPF and DKIM together. While SPF validates the sending server's IP and DKIM validates the message signature, neither alone tells receivers what to do with a message that fails both checks. DMARC fills that gap.

DMARC also requires that the domain in the "From:" header actually matches the domain that passed SPF or DKIM. Without alignment, attackers can construct messages that pass SPF (via a different domain they control) while still showing a spoofed "From:" address.

$ dig TXT _dmarc.google.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _dmarc.google.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64353
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 7522ad7bdfcf0aa80100000069d483f80192e6bdabaee2dd (good)
;; QUESTION SECTION:
;_dmarc.google.com.             IN      TXT

;; ANSWER SECTION:
_dmarc.google.com.      300     IN      TXT     "v=DMARC1; p=reject; rua=mailto:mailauth-reports@google.com"

;; Query time: 83 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 00:11:36 EDT 2026
```

Figure 7: v=DMARC1; p=reject; rua=mailto:mailauth-reports@google.com. Strictest enforcement: mail failing both SPF and DKIM alignment is rejected outright.

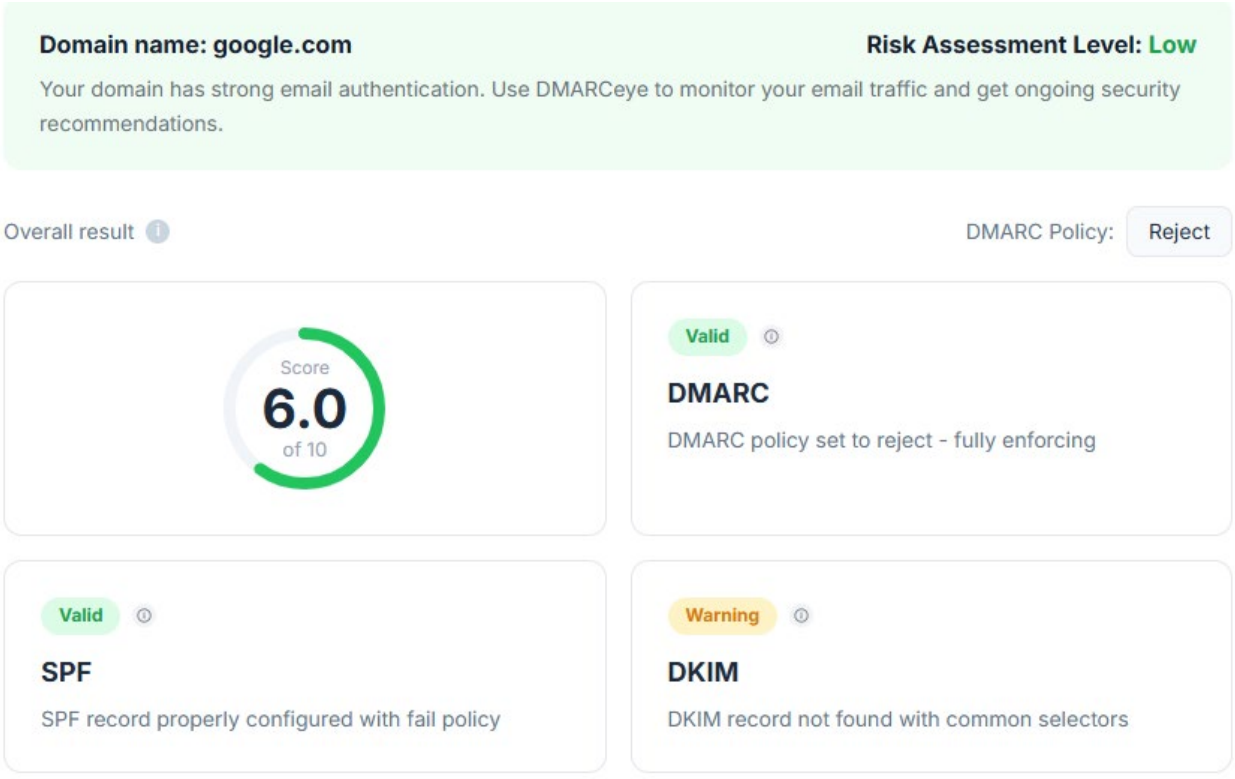

Figure 8: DMARCeye summary for google.com — Score 6.0/10, Risk: Low.

Analysis:

- **p=reject** — Strictest level. Mail failing both SPF and DKIM alignment is rejected outright.
- **rua=mailto:mailauth-reports@google.com** — Aggregate reports enabled. Google monitors its own authentication ecosystem.
- No ruf= tag — Forensic per-message reports not configured. Common for large organisations due to privacy/volume implications.
- No pct= tag — Defaults to pct=100 (policy applies to 100% of mail).

DKIM — DomainKeys Identified Mail

```
$ dig TXT google._domainkey.google.com
$ dig TXT default._domainkey.google.com
$ dig TXT mail._domainkey.google.com
$ dig TXT google._domainkey.gmail.com
```

```
┌──(kali㉿kali)-[~]
└─$ dig TXT google._domainkey.google.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT google._domainkey.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 24305
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: d719bc919d011fa00100000069d48faf3a465c492d73bd5e (good)
;; QUESTION SECTION:
;google._domainkey.google.com.  IN      TXT

;; AUTHORITY SECTION:
google.com.             60      IN      SOA     ns1.google.com. dns-admin.google.com. 895237113 900 900 1800 60

;; Query time: 75 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 01:01:35 EDT 2026
;; MSG SIZE  rcvd: 135
```

Figure 9: The "google" selector is not published in public DNS. Google deliberately does not expose DKIM selectors under guessable names.

```
┌──(kali㉿kali)-[~]
└─$ dig TXT default._domainkey.google.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT default._domainkey.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 41951
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 49ca1876bb509f380100000069d490255631731fd8def9dd (good)
;; QUESTION SECTION:
;default._domainkey.google.com. IN      TXT

;; AUTHORITY SECTION:
google.com.             60      IN      SOA     ns1.google.com. dns-admin.google.com. 895237113 900 900 1800 60

;; Query time: 83 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 01:03:33 EDT 2026
;; MSG SIZE  rcvd: 136
```

Figure 10: The "default" selector is also absent. DKIM key management is entirely internal.

```
┌──(kali㉿kali)-[~]
└─$ dig TXT mail._domainkey.google.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT mail._domainkey.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 3504
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: f23b267fd61ac9f00100000069d49045b13f7ca7825613d3 (good)
;; QUESTION SECTION:
;mail._domainkey.google.com.    IN      TXT

;; AUTHORITY SECTION:
google.com.             60      IN      SOA     ns1.google.com. dns-admin.google.com. 895237113 900 900 1800 60

;; Query time: 75 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 01:04:05 EDT 2026
;; MSG SIZE  rcvd: 133
```

Figure 11: This is deliberate Google uses non-guessable, periodically-rotated selector names discoverable only from real email headers.

```
┌──(kali㉿kali)-[~]
└─$ dig TXT google._domainkey.gmail.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT google._domainkey.gmail.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 40678
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google._domainkey.gmail.com.   IN      TXT

;; AUTHORITY SECTION:
gmail.com.              60      IN      SOA     ns1.google.com. dns-admin.google.com. 895796075 900 900 1800 60

;; Query time: 0 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:06:02 EDT 2026
;; MSG SIZE  rcvd: 122
```

Figure 12: SOA from ns1.google.com confirms the query reached Google's authoritative DNS. Same result no public selector exposed for gmail.com.

Analysis:

- All tested selectors (google, default, mail) return NXDOMAIN — this is deliberate, not a failure.
- Google rotates DKIM selectors internally; they are only discoverable from DKIM-Signature headers in real emails.
- DMARC p=reject provides full enforcement even without a publicly visible DKIM selector.

2. Microsoft (microsoft.com)

SPF — Sender Policy Framework

```
$ dig TXT microsoft.com | grep spf
```

```
┌──(kali㉿kali)-[~]
└─$ dig TXT microsoft.com | grep spf1
microsoft.com.          0       IN      TXT     "v=spf1 include:_spf-a.microsoft.com include:_spf-b.microsoft.com inc
lude:_spf-c.microsoft.com include:_spf-ssg-a.msft.net include:_spf1-meo.microsoft.com -all"
```

Figure 13: dig TXT microsoft.com | grep SPF

Analysis:

- -all (HardFail) — Mail from unlisted sources is actively rejected at the SPF layer. Stricter than Google's ~all.
- Five include: sub-records separate different sending systems (internal, cloud, MEO) for independent management.

DMARC — Domain-based Message Authentication

```
$ dig TXT _dmarc.microsoft.com
```

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _dmarc.microsoft.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.microsoft.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40847
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;_dmarc.microsoft.com.          IN      TXT

;; ANSWER SECTION:
_dmarc.microsoft.com.   0       IN      TXT     "v=DMARC1; p=reject; pct=100; rua=mailto:itex-rua@microsoft.com; ruf=
mailto:itex-ruf@microsoft.com; fo=1"

;; Query time: 43 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:09:46 EDT 2026
;; MSG SIZE  rcvd: 174
```

Figure 14: dig TXT _dmarc.microsoft.com

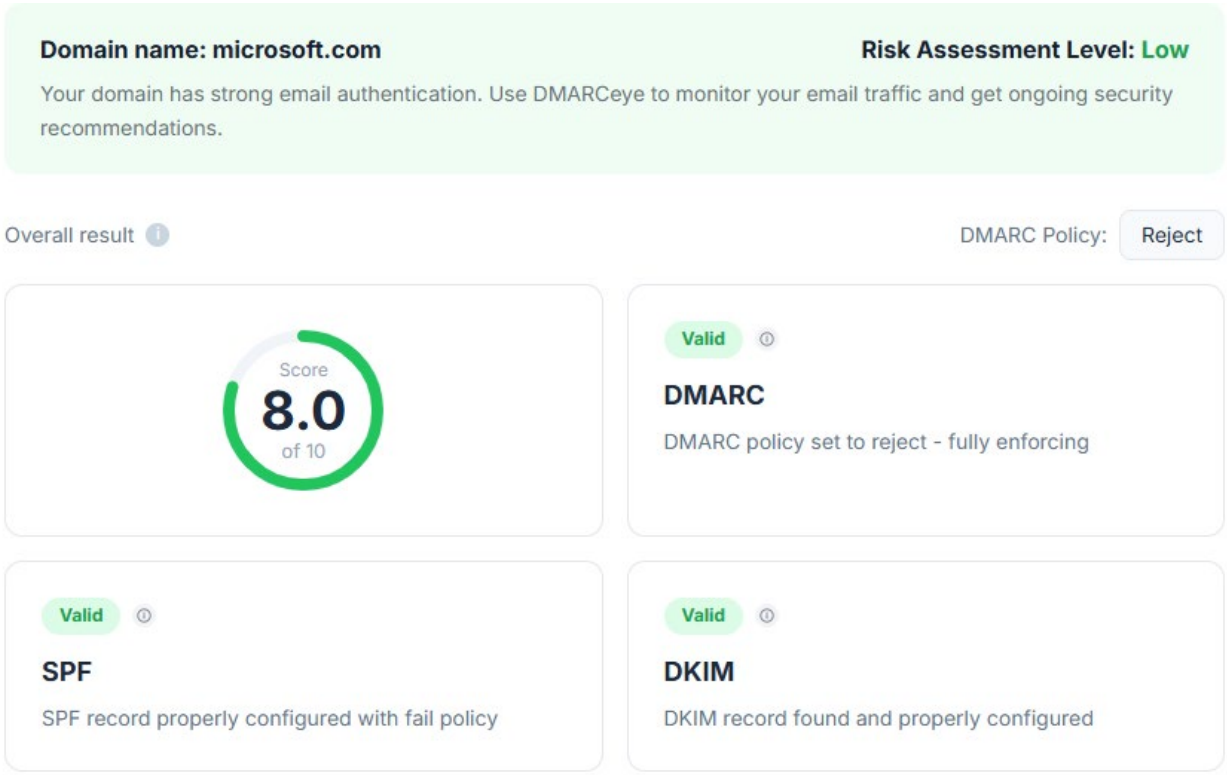

Figure 15: DMARCeye summary for microsoft.com Score 8.0/10

Analysis:

- **p=reject** — Strictest enforcement. Mail failing authentication is rejected.
- **pct=100** — Explicitly set. Policy applies to every message with no phased rollout.
- **ruf=mailto:itex-ruf@microsoft.com** — Forensic reports enabled. Microsoft receives per-message failure reports for incident response.
- **fo=1** — Generate forensic report when any check (SPF or DKIM) fails. Maximises visibility.

DKIM — DomainKeys Identified Mail

$ dig TXT selector1._domainkey.microsoft.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.microsoft.com


; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.microsoft.com
;; global options: +cmd
;; Got answer:
;; →»HEADER«← opcode: QUERY, status: NXDOMAIN, id: 52479
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: f6cc324eb55cfbb60100000069d490044a0d56dff77576c5 (good)
;; QUESTION SECTION:
;selector1._domainkey.microsoft.com. IN TXT

;; ANSWER SECTION:
selector1._domainkey.microsoft.com. 3600 IN CNAME selector1-microsoft-com._domainkey.microsoft.onmicrosoft.com.

;; AUTHORITY SECTION:
onmicrosoft.com.        300     IN      SOA     ns1-208.azure-dns.com. azuredns-hostmaster.microsoft.com. 1 3600 300 2419200 300

;; Query time: 179 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 01:03:00 EDT 2026
;; MSG SIZE  rcvd: 236
```

Figure 16: dig TXT selector1._domainkey.microsoft.

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.microsoft.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.microsoft.com
;; global options: +cmd
;; Got answer:
;; →»HEADER«← opcode: QUERY, status: NXDOMAIN, id: 7041
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;selector1._domainkey.microsoft.com. IN TXT

;; AUTHORITY SECTION:
onmicrosoft.com.        266     IN      SOA     ns1-208.azure-dns.com. azuredns-hostmaster.microsoft.com. 1 3600 300
2419200 300

;; Query time: 87 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:10:27 EDT 2026
;; MSG SIZE  rcvd: 162
```

Figure 17: dig TXT selector1._domainkey.microsoft.com

Analysis:

- **CNAME delegation** — selector1._domainkey.microsoft.com → selector1-microsoft-com._domainkey.microsoft.onmicrosoft.com. Microsoft controls the target, enabling silent key rotation.
- Two selectors provisioned (selector1, selector2) for seamless rotation — one active, one in reserve.
- Only domain among the primary three where a valid DKIM key is directly reachable via a common selector name.

3. Facebook (facebook.com)

SPF — Sender Policy Framework

$ dig TXT facebook.com | grep spf

```
┌──(kali㉿kali)-[~]
└─$ dig TXT facebook.com | grep spf

facebook.com.           0       IN      TXT     "v=spf1 redirect=_spf.facebook.com"
```

Figure 18: dig TXT facebook.com | grep spf

Analysis:

- **redirect=_spf.facebook.com** — Delegates the entire SPF policy to a sub-record. Does not consume an include: lookup count.
- DMARCeye flags SPF as "Invalid" because no direct SPF record exists on the domain itself — this is a tool limitation, not a real security gap.

DMARC — Domain-based Message Authentication

$ dig +short TXT _dmarc.facebook.com

```
┌──(kali㉿kali)-[~]
└─$ dig +short TXT _dmarc.facebook.com

"v=DMARC1; p=reject; rua=mailto:a@dmarc.facebookmail.com; ruf=mailto:fb-dmarc@datafeeds.phishlabs.com; pct=100"
```

Figure 19: dig TXT _dmarc.facebook.com

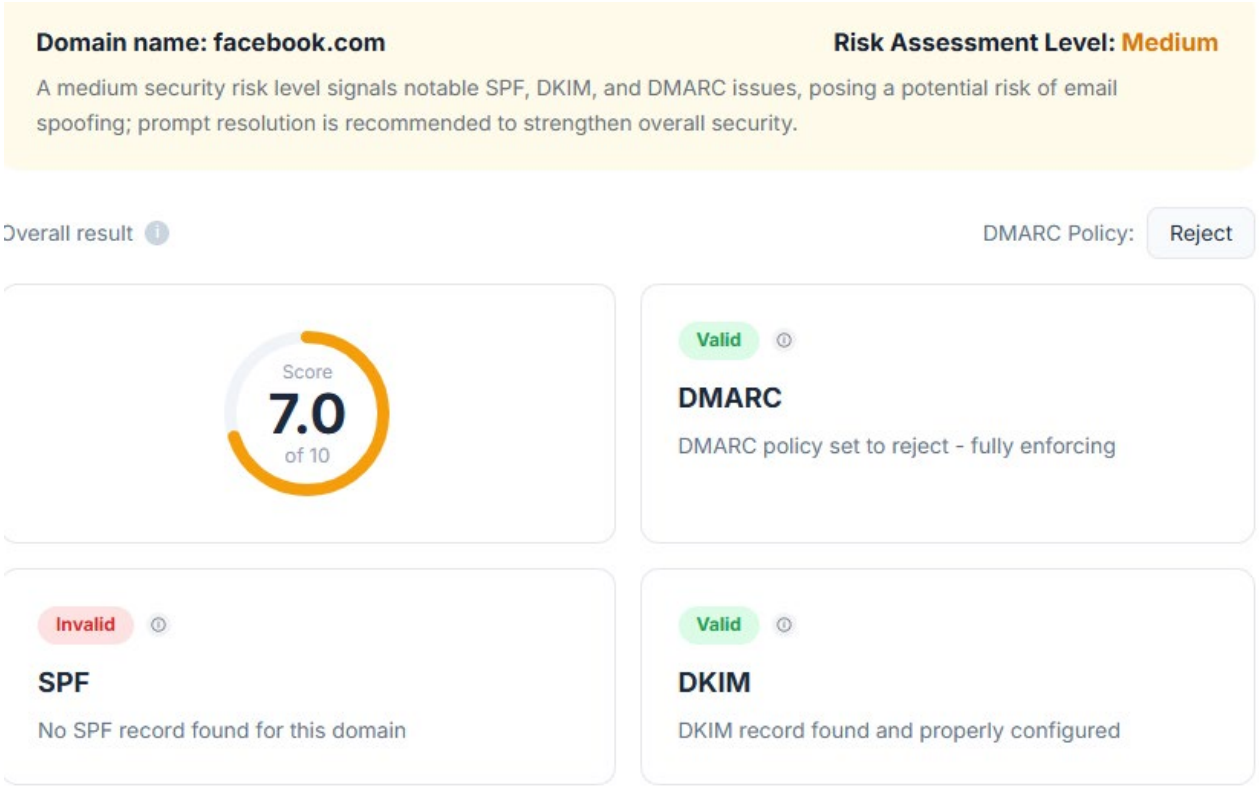

Figure 20: DMARCeye summary for facebook.com — Score 7.0/10

Analysis:

- **p=reject** — Full enforcement. Failing mail is rejected.
- **pct=100** — Policy applies to all messages.
- **ruf=mailto:fb-dmarc@datafeeds.phishlabs.com** — PhishLabs monitors Facebook's DMARC failure data in real time for phishing campaigns targeting the brand.

DKIM — DomainKeys Identified Mail

```
$ dig TXT s1._domainkey.facebook.com

$ dig TXT s2._domainkey.facebook.com

$ dig TXT default._domainkey.facebook.com
```

```
┌──(kali㉿kali)-[~]
└─$ dig TXT s1._domainkey.facebook.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT s1._domainkey.facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 49219
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 512bc41ea3055fda0100000069d48fcfa5bcbf382f45232b (good)
;; QUESTION SECTION:
;s1._domainkey.facebook.com.    IN      TXT

;; AUTHORITY SECTION:
facebook.com.           1800    IN      SOA     a.ns.facebook.com. dns.facebook.com. 4207849484 14400 1800 604800 300

;; Query time: 67 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 01:02:07 EDT 2026
;; MSG SIZE  rcvd: 128
```

Figure 21: dig TXT s1._domainkey.facebook.com.

```
┌──(kali㉿kali)-[~]
└─$ dig TXT s2._domainkey.facebook.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT s2._domainkey.facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 32893
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: b285f8688ddbf0510100000069d48fe63d860fc650e0281b (good)
;; QUESTION SECTION:
;s2._domainkey.facebook.com.    IN      TXT

;; AUTHORITY SECTION:
facebook.com.           1800    IN      SOA     a.ns.facebook.com. dns.facebook.com. 4207849484 14400 1800 604800 300

;; Query time: 71 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 01:02:30 EDT 2026
;; MSG SIZE  rcvd: 128
```

Figure 22: dig TXT s2._domainkey.facebook.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT default._domainkey.facebook.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT default._domainkey.facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40997
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;default._domainkey.facebook.com. IN    TXT

;; ANSWER SECTION:
default._domainkey.facebook.com. 0 IN   TXT     "t=y; k=rsa; p=;"

;; Query time: 44 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:12:20 EDT 2026
;; MSG SIZE  rcvd: 108
```

Figure 23: dig TXT default._domainkey.facebook.

Analysis:

- s1 and s2 selectors: NXDOMAIN — not publicly exposed, consistent with Google.
- **default selector: p=; (empty key)** — Key intentionally revoked. Signing with this selector produces DKIM failure. Facebook relies on DMARC p=reject for enforcement.
- t=y flag on the default selector indicates testing mode — receivers should not reject based on this record.

4. Amazon (amazon.com)

SPF — Sender Policy Framework

$ dig TXT amazon.com | grep spf

```
┌──(kali㉿kali)-[~]
└─$ dig TXT amazon.com | grep spf
amazon.com.             0       IN      TXT     "v=spf1 include:spf1.amazon.com include:spf2.amazon.com include:amazo
nses.com -all"
amazon.com.             0       IN      TXT     "spf2.0/pra include:spf1.amazon.com include:spf2.amazon.com include:a
mazonses.com -all"
```

Figure 24: dig TXT amazon.com | grep spf

Analysis:

- **v=spf2.0/pra** — Legacy Sender ID record (Microsoft-era). Redundant today but harmless.
- **-all (HardFail)** — Both SPF records use HardFail — stricter than Google, Facebook, PayPal, and Shopify.
- amazonses.com include: authorises Amazon Simple Email Service as a sending provider.

DMARC — Domain-based Message Authentication

$ dig TXT _dmarc.amazon.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _dmarc.amazon.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.amazon.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49553
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;_dmarc.amazon.com.             IN      TXT

;; ANSWER SECTION:
_dmarc.amazon.com.      0       IN      TXT     "v=DMARC1;" "p=quarantine;" "pct=100;" "rua=mailto:report@dmarc.amazo
n.com;" "ruf=mailto:report@dmarc.amazon.com"

;; Query time: 52 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:13:52 EDT 2026
;; MSG SIZE  rcvd: 168
```

Figure 25: dig TXT _dmarc.amazon.com

Analysis:

- **p=quarantine** — Weaker than p=reject. Failing messages go to spam folder rather than being rejected. Represents a relative weakness versus Google, Microsoft, and Facebook.
- Both rua and ruf route to Amazon's own DMARC address — in-house monitoring.

DKIM — DomainKeys Identified Mail

$ dig TXT selector1._domainkey.amazon.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.amazon.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.amazon.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 50729
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;selector1._domainkey.amazon.com. IN    TXT

;; AUTHORITY SECTION:
amazon.com.             900     IN      SOA     dns-external-master.amazon.com. hostmaster.amazon.com. 2010198605 180
 60 604800 900

;; Query time: 88 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:14:15 EDT 2026
;; MSG SIZE  rcvd: 137
```

Figure 26: dig TXT selector1._domainkey.amazon.com.

Analysis:

- NXDOMAIN for selector1 — Amazon rotates selectors internally, consistent with Google and Facebook.
- SOA from Amazon's own authoritative nameservers confirms the query reached the correct DNS.

5. Netflix (netflix.com)

SPF — Sender Policy Framework

$ dig TXT netflix.com | grep spf

```
┌──(kali㉿kali)-[~]
└─$ dig TXT netflix.com | grep spf
netflix.com.            0       IN      TXT     "v=spf1 include:_spf_ipv4.netflix.com include:_spf.google.com include
:amazonses.com include:servers.mcsv.net include:_spf.salesforce.com include:_spf.createsend.com -all"
```

Figure 27: dig TXT netflix.com | grep spf

Analysis:

- **-all (HardFail)** — Unauthorised senders are rejected at SPF layer.
- Six providers included: own IP ranges, Google Workspace, Amazon SES, Mailchimp, Salesforce, Campaign Monitor — reflects Netflix's transactional, marketing, and system email needs.

DMARC — Domain-based Message Authentication

$ dig TXT _dmarc.netflix.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _dmarc.netflix.com


; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.netflix.com
;; global options: +cmd
;; Got answer:
;; →»HEADER«← opcode: QUERY, status: NOERROR, id: 18977
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;_dmarc.netflix.com.            IN      TXT

;; ANSWER SECTION:
_dmarc.netflix.com.     0       IN      TXT     "v=DMARC1; p=reject; fo=1; rua=mailto:netflix@rua.netcraft.com,mailto
:dmarcreports@netflix.com,mailto:dmarc_agg@dmarc.250ok.net;ruf=mailto:netflix@ruf.netcraft.com,mailto:dmarcreports@ne
tflix.com,mailto:dmarc_fr@dmarc.250ok.net"

;; Query time: 4 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:16:10 EDT 2026
;; MSG SIZE  rcvd: 293
```

Figure 28: dig TXT _dmarc.netflix.com.

Analysis:

- **p=reject** — Full enforcement.
- **fo=1** — Forensic report generated on any authentication failure.
- Two independent vendors (Netcraft + 250ok/Validity) provide redundant DMARC monitoring.

DKIM — DomainKeys Identified Mail

$ dig TXT selector1._domainkey.netflix.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.netflix.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.netflix.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11326
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;selector1._domainkey.netflix.com. IN   TXT

;; ANSWER SECTION:
selector1._domainkey.netflix.com. 0 IN  TXT     "v=spf1 include:_spf_ipv4.netflix.com include:_spf.google.com include
:amazonses.com include:_spf.salesforce.com -all"

;; Query time: 84 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:16:32 EDT 2026
;; MSG SIZE  rcvd: 210
```

Figure 29: dig TXT selector1._domainkey.netflix.com

Analysis:

- **DNS misconfiguration detected** — selector1._domainkey.netflix.com returns an SPF record instead of a DKIM key. This means DKIM signing via this selector will produce verification failures.
- Despite the DKIM misconfiguration, DMARC p=reject still provides enforcement via SPF alignment.
- This anomaly should be remediated: the DKIM selector DNS entry needs to be corrected to point to the actual public key.

6. X (Twitter) (x.com)

SPF — Sender Policy Framework

$ dig TXT x.com | grep spf

```
┌──(kali㉿kali)-[~]
└─$ dig TXT x.com | grep spf
x.com.                  0       IN      TXT     "v=spf1 ip4:199.16.156.0/22 ip4:199.59.148.0/22 include:_spf.google.c
om include:_spf.salesforce.com include:_oerp.x.com include:phx1.rp.oracleemaildelivery.com include:iad1.rp.oracleemai
ldelivery.com -all"
```

Figure 30: dig TXT x.com | grep spf

Analysis:

- **-all (HardFail)** — Unauthorised senders are rejected.
- Direct IPv4 ranges (199.16.156.0/22, 199.59.148.0/22) are X's own data centre IP blocks inherited from Twitter infrastructure.
- Oracle Email Delivery used alongside Salesforce and Google — notable for a social media platform.

DMARC — Domain-based Message Authentication

$ dig TXT _dmarc.x.com

Figure 31: dig TXT _dmarc.x.com

Analysis:

- **p=reject** — Full enforcement. No pct= specified — defaults to 100%.
- Reporting routed to Cloudflare DMARC reporting infrastructure — outsourced monitoring.
- Hashed reporting address prevents easy enumeration of the reporting endpoint.

DKIM — DomainKeys Identified Mail

$ dig TXT selector1._domainkey.x.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.x.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.x.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 25014
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;selector1._domainkey.x.com.    IN      TXT

;; AUTHORITY SECTION:
x.com.                  300     IN      SOA     a.u10.twtrdns.net. noc.twitter.com. 2023121201 3600 600 604800 300

;; Query time: 87 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:18:02 EDT 2026
;; MSG SIZE  rcvd: 125
```

Figure 32: dig TXT selector1._domainkey.x.

Analysis:

- NXDOMAIN — selector1 not exposed. X uses non-guessable internal selectors.
- SOA from twtrdns.net reveals X.com still operates on legacy Twitter DNS infrastructure.

7. PayPal (paypal.com)

SPF — Sender Policy Framework

$ dig TXT paypal.com | grep spf

```
┌──(kali㉿kali)-[~]
└─$ dig TXT paypal.com | grep spf
paypal.com.             0       IN      TXT     "v=spf1 include:pp._spf.paypal.com include:3ph1._spf.paypal.com inclu
de:3ph2._spf.paypal.com include:3ph3._spf.paypal.com include:3ph4._spf.paypal.com include:sendgrid.net include:aspmx.
pardot.com ~all"
```

Figure 33: dig TXT paypal.com | grep spf

Analysis:

- **~all (SoftFail)** — Less strict than -all. DMARC p=reject compensates.
- Five own sub-records (pp._spf, 3ph1–3ph4._spf) segment different PayPal sending systems for independent auditing.
- SendGrid and Pardot (Salesforce) authorised for transactional and marketing email respectively.

DMARC — Domain-based Message Authentication

$ dig TXT _dmarc.paypal.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _dmarc.paypal.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.paypal.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59276
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;_dmarc.paypal.com.             IN      TXT

;; ANSWER SECTION:
_dmarc.paypal.com.      0       IN      TXT     "v=DMARC1; p=reject; rua=mailto:d@rua.agari.com; ruf=mailto:d@ruf.ag
ri.com"

;; Query time: 47 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:18:52 EDT 2026
;; MSG SIZE  rcvd: 139
```

Figure 34: dig TXT _dmarc.paypal.

Analysis:

- **p=reject** — Full enforcement.
- **Agari reporting** — PayPal uses a specialist vendor given its extremely high value as a phishing target. Agari provides real-time analysis and brand protection monitoring.

DKIM — DomainKeys Identified Mail

$ dig TXT selector1._domainkey.paypal.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.paypal.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.paypal.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 11457
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;selector1._domainkey.paypal.com. IN    TXT

;; AUTHORITY SECTION:
paypal.com.             300     IN      SOA     ppdns.paypal.com. hostmaster.paypal.com. 2012397148 7200 600 1209600
300

;; Query time: 80 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:19:18 EDT 2026
;; MSG SIZE  rcvd: 123
```

Figure 35: dig TXT selector1._domainkey.paypal.com.

Analysis:

- NXDOMAIN — PayPal rotates DKIM selectors internally.
- ppdns.paypal.com as authoritative nameserver confirms PayPal operates own DNS infrastructure.

8. Shopify (shopify.com)

SPF — Sender Policy Framework

$ dig TXT shopify.com | grep spf

```
┌──(kali㉿kali)-[~]
└─$ dig TXT shopify.com | grep spf
shopify.com.            0       IN      TXT     "v=spf1 include:_spf.google.com include:mail.zendesk.com include:send
grid.net ~all"
```

Figure 36: dig TXT shopify.com | grep spf

Analysis:

- **~all (SoftFail)** — DMARC p=reject provides hard enforcement.
- Lean three-provider SPF: Google (internal), Zendesk (support), SendGrid (transactional).

DMARC — Domain-based Message Authentication

$ dig TXT _dmarc.shopify.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _dmarc.shopify.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.shopify.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58120
;; flags: qr rd ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;_dmarc.shopify.com.            IN      TXT

;; ANSWER SECTION:
_dmarc.shopify.com.     0       IN      CNAME   dmarcreject.shopify.com.
dmarcreject.shopify.com. 0      IN      TXT     "v=DMARC1; p=reject; pct=100; fo=1; rua=mailto:dmarc-aggregate@shopif
y.com;ruf=mailto:dmarc-reports@shopify.com"

;; Query time: 47 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:20:12 EDT 2026
;; MSG SIZE  rcvd: 237
```

Figure 37: dig TXT _dmarc.shopify.com

Analysis:

- **CNAME indirection** — _dmarc.shopify.com → dmarcreject.shopify.com → actual TXT. Allows policy updates without changing the DNS entry directly.
- **p=reject, pct=100, fo=1** — Full enforcement, 100% coverage, maximum forensic visibility.

DKIM — DomainKeys Identified Mail

$ dig TXT selector1._domainkey.shopify.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.shopify.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.shopify.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 24033
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;selector1._domainkey.shopify.com. IN   TXT

;; AUTHORITY SECTION:
shopify.com.            1200    IN      SOA     gold.foundationdns.com. dns.cloudflare.com. 2401233091 10000 2400 604
800 1200

;; Query time: 47 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:20:32 EDT 2026
;; MSG SIZE  rcvd: 142
```

Figure 38: dig TXT selector1._domainkey.shopify.com

Analysis:

- NXDOMAIN — selector1 not exposed.
- Shopify uses FoundationDNS (via Cloudflare) as authoritative DNS provider.

9. Dropbox (dropbox.com)

SPF — Sender Policy Framework

$ dig TXT dropbox.com | grep spf

```
┌──(kali㉿kali)-[~]
└─$ dig TXT dropbox.com | grep spf
dropbox.com.            0       IN      TXT     "v=spf1 ip4:45.58.64.0/20 ip4:185.45.8.0/22 ip4:162.125.0.0/16 ip4:19
9.47.216.0/22 ip4:108.160.160.0/20 ip4:205.189.0.0/24 ip4:160.34.15.16/28 ip4:52.5.134.202/32 ip4:205.220.162.87/32 i
p4:205.220.174.83/32 ip4:167.89.98.146/32 ip4:167.89.89.46/32" " ip4:167.89.96.134/32 ip4:159.183.109.97/32 ip4:149.7
2.220.85/32 ip4:159.183.15.144/32 ip4:159.183.2.51/32 ip4:159.183.2.58/32 ip6:2620:c6:8000::/48 include:amazonses.com
 include:_spf.google.com include:mail.zendesk.com include:mktomail.com" " include:rp.oracleemaildelivery.com exists:%
{i}._spf.mta.salesforce.com ~all"
```

Figure 39: dig TXT dropbox.com | grep spf

Analysis:

- Largest SPF record tested — combines direct IPs (own servers) with multiple cloud provider includes.
- **exists:%{i}._spf.mta.salesforce.com** — Dynamic exists: mechanism for Salesforce. Unusual and can cause SPF failures if misresolved.
- **~all (SoftFail)** — DMARC p=reject provides the enforcement layer.

DMARC — Domain-based Message Authentication

$ dig TXT _dmarc.dropbox.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _dmarc.dropbox.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.dropbox.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27705
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;_dmarc.dropbox.com.            IN      TXT

;; ANSWER SECTION:
_dmarc.dropbox.com.     0       IN      TXT     "v=DMARC1;p=reject;pct=100;rua=mailto:c7xrs-8253@rua.dmarc.emailanaly
st.com,mailto:dmarc@dropbox.com"

;; Query time: 47 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:21:25 EDT 2026
;; MSG SIZE  rcvd: 166
```

Figure 40: dig TXT _dmarc.dropbox.com

Analysis:

- **p=reject, pct=100** — Full enforcement on 100% of mail.
- Dual DMARC reporting: EmailAnalyst (specialist analysis) + internal address (direct visibility).

DKIM — DomainKeys Identified Mail

$ dig TXT selector1._domainkey.dropbox.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.dropbox.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.dropbox.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 33275
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;selector1._domainkey.dropbox.com. IN   TXT

;; AUTHORITY SECTION:
dropbox.com.            900     IN      SOA     ns-564.awsdns-06.net. awsdns-hostmaster.amazon.com. 1 7200 900 120960
0 86400

;; Query time: 87 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:21:44 EDT 2026
;; MSG SIZE  rcvd: 153
```

Figure 41: dig TXT selector1._domainkey.dropbox.

Analysis:

- NXDOMAIN — selector1 not exposed.
- DNS hosted on Amazon Route 53 (awsdns-06.net nameservers).

10. GitHub (github.com)

SPF — Sender Policy Framework

$ dig TXT github.com | grep spf

```
┌──(kali㉿kali)-[~]
└─$ dig TXT github.com | grep spf
github.com.             0       IN      TXT     "v=spf1 ip4:192.30.252.0/22 include:spf.protection.outlook.com includ
e:_netblocks.google.com include:_netblocks2.google.com include:mail.zendesk.com include:_spf.salesforce.com include:s
ervers.mcsv.net include:mktomail.com include:sendgrid.net ip4:62.253.2" "27.114 ip4:166.78.69.169 ip4:166.78.69.170 i
p4:166.78.71.131 ~all"
```

Figure 42: dig TXT github.com | grep spf

Analysis:

- **spf.protection.outlook.com** — Confirms GitHub uses Microsoft 365 (Outlook/Exchange Online) for primary email.
- ~all SoftFail — DMARC sp=reject on subdomains provides stricter protection.
- Broad provider set: M365, Google, Zendesk, Salesforce, Mailchimp, Marketo, SendGrid.

DMARC — Domain-based Message Authentication

$ dig TXT _dmarc.github.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT _dmarc.github.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.github.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28370
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;_dmarc.github.com.             IN      TXT

;; ANSWER SECTION:
_dmarc.github.com.      0       IN      TXT     "v=DMARC1; p=quarantine; sp=reject; pct=100; rua=mailto:dmarc@github.
com; ruf=mailto:dmarc@github.com; fo=1"

;; Query time: 51 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:22:35 EDT 2026
;; MSG SIZE  rcvd: 171
```

Figure 43: dig TXT _dmarc.github.com

Analysis:

- **p=quarantine** — Main domain policy: failing mail directed to spam (weaker than reject).
- **sp=reject** — Subdomains (e.g. mail.github.com) use the strictest enforcement. Subdomains are often more exploitable.
- **fo=1** — Maximum forensic reporting visibility.
- All reports routed internally to dmarc@github.com.

DKIM — DomainKeys Identified Mail

$ dig TXT selector1._domainkey.github.com

```
┌──(kali㉿kali)-[~]
└─$ dig TXT selector1._domainkey.github.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT selector1._domainkey.github.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42681
;; flags: qr rd ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;selector1._domainkey.github.com. IN    TXT

;; ANSWER SECTION:
selector1._domainkey.github.com. 0 IN   CNAME   selector1-github-com._domainkey.microsoft.onmicrosoft.com.
selector1-github-com._domainkey.microsoft.onmicrosoft.com. 0 IN TXT "v=DKIM1; k=rsa; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNAD
CBiQKBgQCxZC/z2cK+2s1f/ktzSDSeFzkfIHrjwtGFsfKMAYvKaXjPVNzKykpbXBkX5nB7dVUTFttda7aROr2iSrIseQ27Ui+4rUZVzgFanE8RaXhYM9n
5wPKNv8GtLJk7JgcsYZ7ErgID4uW2sEYyV/dnRYw6rDzOaeRKkKELTUfJI5ebLQIDAQAB;"

;; Query time: 151 msec
;; SERVER: 172.26.80.1#53(172.26.80.1) (UDP)
;; WHEN: Sat Apr 11 03:22:54 EDT 2026
;; MSG SIZE  rcvd: 456
```

Figure 44: dig TXT selector1._domainkey.github.com

Analysis:

- **CNAME → microsoft.onmicrosoft.com** — Confirms GitHub's email is handled by Microsoft 365. The public key is active and the record is properly configured.
- Only GitHub and Microsoft.com returned valid, reachable DKIM public keys among all tested domains.
- Microsoft manages key rotation centrally — GitHub does not need to update DNS when keys rotate.

DKIM Record Field Reference

DKIM public keys live in DNS at <selector>._domainkey.<domain>. The key record contains the following fields:

Field	Example	Meaning	Security Note
v=	DKIM1	Record version — must be DKIM1	Required; record ignored without it
k=	rsa	Key algorithm — rsa or ed25519	ed25519 is newer and more efficient
p=	MIIBIjAN…	Base64-encoded RSA public key	Empty p= means key is revoked — all mail fails
t=	y	Testing mode — receivers should not reject	Remove t=y when going live
h=	sha256	Hash algorithm for the signature	sha1 is deprecated; sha256 required

DMARC Tag Reference

Tag	Example	Meaning	Importance
p=	reject / quarantine / none	Enforcement policy for failing messages	CRITICAL — drives actual protection
rua=	reports@domain.com	Aggregate report recipients (daily XML)	HIGH — visibility into mail flows
ruf=	forensic@domain.com	Forensic per-failure report recipients	HIGH — incident response
pct=	100	Percentage of mail policy applies to	HIGH — set to 100 for full coverage
sp=	reject	Subdomain policy (overrides for *.domain)	MEDIUM — protects subdomains
adkim=	s or r	DKIM alignment: strict (s) or relaxed (r)	MEDIUM — strict is more secure
aspf=	s or r	SPF alignment: strict (s) or relaxed (r)	MEDIUM — strict is more secure
fo=	0, 1, d, s	When to generate forensic reports	MEDIUM — fo=1 gives max visibility

Correlation & Risk Analysis

Attacker's Decision Matrix

A threat actor attempting to spoof email must defeat all three mechanisms simultaneously. The decision tree below shows the evaluation path:

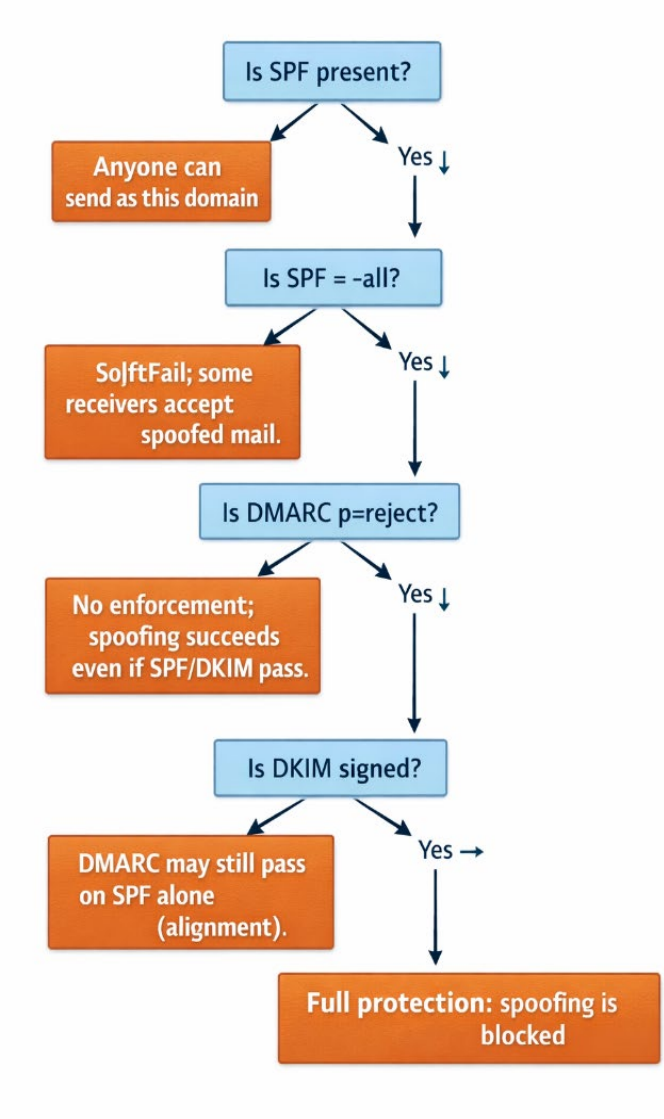

Figure 45: Attacker's decision matrix — Flowchart showing the layered defence. For all domains with DMARC p=reject, an attacker must bypass SPF, DKIM, and DMARC simultaneously.

Conclusion

This assessment provides a comprehensive, intelligence-driven evaluation of email authentication controls across ten high-value global domains, leveraging purely passive DNS reconnaissance techniques. The findings demonstrate that modern enterprise-grade organizations have largely matured beyond basic email security misconfigurations, with DMARC enforcement emerging as the decisive control layer against domain spoofing.

The findings indicate that all assessed domains have implemented DMARC policies at either quarantine or reject levels, demonstrating a high level of security maturity. In particular, domains enforcing a DMARC policy of p=reject provide strong protection against unauthorized email sources by instructing receiving servers to reject messages that fail authentication and alignment checks. This confirms that DMARC serves as the primary enforcement layer in modern email security architectures.

SPF records were present in all domains; however, a significant number of them utilized the SoftFail (~all) mechanism rather than the stricter HardFail (-all) policy. While this configuration may appear less restrictive, it does not introduce significant risk when combined with a properly enforced DMARC policy. In such cases, DMARC compensates for SPF leniency by applying strict rejection rules when alignment conditions are not met.

DKIM analysis revealed that most organizations do not expose publicly predictable selector names, instead relying on dynamically generated or non-guessable selectors. This practice enhances security by limiting the ability of external entities to enumerate DKIM keys. Additionally, some domains employ CNAME delegation to external services (e.g., cloud email providers), enabling centralized key management and seamless key rotation. Minor misconfigurations were identified, such as incorrect DNS records for DKIM selectors; however, these issues did not significantly impact overall security due to the presence of enforced DMARC policies.

A key observation from this study is that no single mechanism—SPF, DKIM, or DMARC—is sufficient in isolation. SPF can be bypassed in scenarios such as email forwarding, and DKIM may fail if keys are misconfigured or unavailable. DMARC, when configured with an enforcement policy and alignment requirements, integrates both mechanisms and provides a decisive layer of protection. Therefore, the combined implementation of all three protocols is essential to achieving effective defense against spoofing.