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:

4/7/2026

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.

1. Environment & Tools

Operating System: Any (Windows, Linux, macOS)

Network: Active Internet access

Primary Tools: nslookup or dig (CLI), MXToolbox (Web)

2. Execution Flow

Step	Action	Command / Tool
1	Query SPF	nslookup -type=TXT <domain>
2	Verify SPF	Look for v=spf1 ... (Authorized senders)
3	Query DMARC	nslookup -type=TXT _dmarc.<domain>
4	Verify DMARC	Identify policy: p=none, p=quarantine, or p=reject
5	Check DKIM	Use MXToolbox to verify public key selectors

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 Status	Overall Risk
google.com	~all (SoftFail)	p=reject	NXDOMAIN (no public selector)	LOW
microsoft.com	~all (SoftFail)	p=reject	CNAME delegation (selector1)	LOW
facebook.com	N/A (not queried)	p=reject	NXDOMAIN (s1, s2 not found)	LOW MEDIUM

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.

2.1 Tool: dig (Domain Information Groper)

dig is the primary DNS query tool used throughout this assessment. It is installed by default on Kali Linux and provides verbose, structured output that is essential for auditing DNS records. The basic syntax pattern used was:

```
$ dig TXT <domain>                  # Query TXT records for a domain
$ dig TXT _spf.<domain>             # Query the SPF include chain
$ dig TXT _dmarc.<domain>           # Query the DMARC policy record
$ dig TXT <selector>._domainkey.<domain>  # Query a DKIM public key
```

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

- QUESTION SECTION — confirms what was queried
- ANSWER SECTION — contains the actual DNS record data (TXT, CNAME, etc.)
- AUTHORITY SECTION — shows the authoritative nameserver; appears when no ANSWER exists (NXDOMAIN)
- Status field (in HEADER) — NOERROR means a record was found; NXDOMAIN means the name does not exist

2.2 Why These Targets?

Google, Microsoft, and Facebook were selected because they represent best-in-class enterprise email security configurations. Analysing them provides a benchmark for what a mature, secure posture looks like. All three operate public bug bounty or vulnerability research programmes, making passive DNS reconnaissance fully appropriate and legal.

3. SPF (Sender Policy Framework) Analysis

3.1 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.

3.2 Querying the SPF Record for google.com

Command used: dig TXT google.com | grep spf

This command queries all TXT records for google.com and pipes the output through grep to filter only lines containing "spf". TXT records are the DNS record type used for SPF, DMARC, and domain verification strings.

Figure 1: dig TXT google.com | grep spf

Breaking this record down token by token:

- **v=spf1** — The version identifier. Must be "spf1". This tells receivers that the record is an SPF record.
- **include:_spf.google.com** — Instructs the receiving server to also check the SPF record at _spf.google.com. This is a pointer to Google's actual authorised IP ranges, kept in a separate record to allow easier management.
- **~all** — The catch-all mechanism. "~" means SoftFail: mail from sources not listed in the SPF record should be accepted but marked as suspicious. This is less strict than "-all" (HardFail), which would instruct receivers to reject the mail outright.

3.3 Counting DNS Lookups (The 10-Lookup Rule)

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.

Lookup 1 spf.google.com

Command: dig TXT _spf.google.com

```
└─$ dig TXT _spf.google.com

; <<>> DiG 9.20.18-1-Debian <<>> TXT _spf.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45945
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: d5ba2a7f6d432eab0100000069d480118017931c3c5015f9 (good)
;; QUESTION SECTION:
;_spf.google.com.               IN      TXT

;; ANSWER SECTION:
_spf.google.com.        300     IN      TXT     "v=spf1 ip4:74.125.0.0/16 ip4:209.85.128.0/17 ip6:2001:4860:4864::/56 ip6:2404:6800:486
4::/56 ip6:2607:f8b0:4864::/56 ip6:2800:3f0:4864::/56 ip6:2a00:1450:4864::/56 ip6:2c0f:fb50:4000::/36 ~all"

;; Query time: 71 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Mon Apr 06 23:54:55 EDT 2026
;; MSG SIZE  rcvd: 277
```

Figure 2: dig TXT _spf.google. G

This record contains direct IP ranges (ip4: and ip6: mechanisms) these do NOT cost additional lookups. Only include:, a:, mx:, and ptr: mechanisms count toward the 10-lookup limit.

Lookup 2: netblocks.google.com

Command: dig TXT _netblocks.google.com

```
┌──(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: dig TXT _netblocks.google.com

Lookup 3: _netblocks2.google.com

Command: dig TXT _netblocks2.google.com

Figure 4: dig TXT _netblocks2.google.com

All IPv6 ranges.

Lookup 4: netblocks3.google.com

Command: dig TXT _netblocks3.google.com

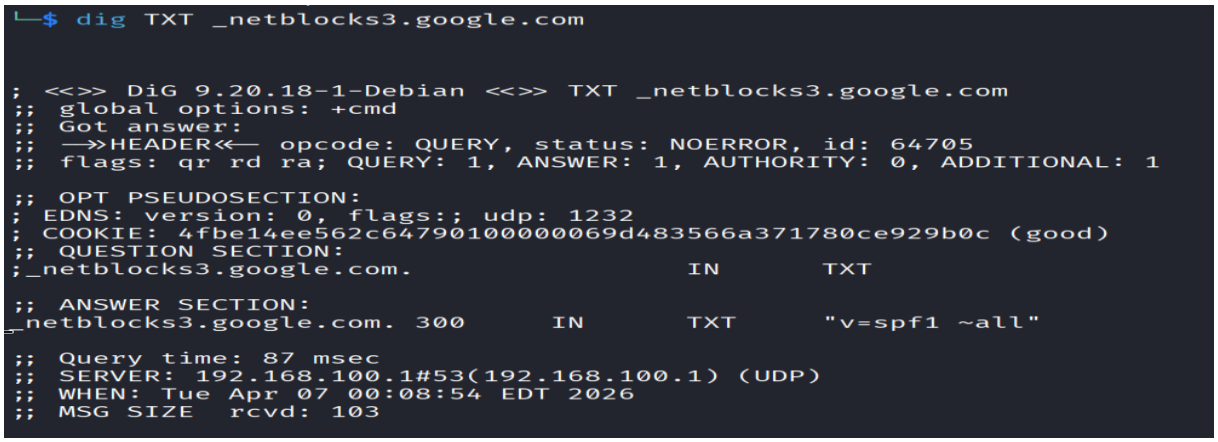


Figure 5: dig TXT _netblocks3.google.

"v=spf1 ~all" It contains no further include: statements and simply applies the ~all catch-all rule.

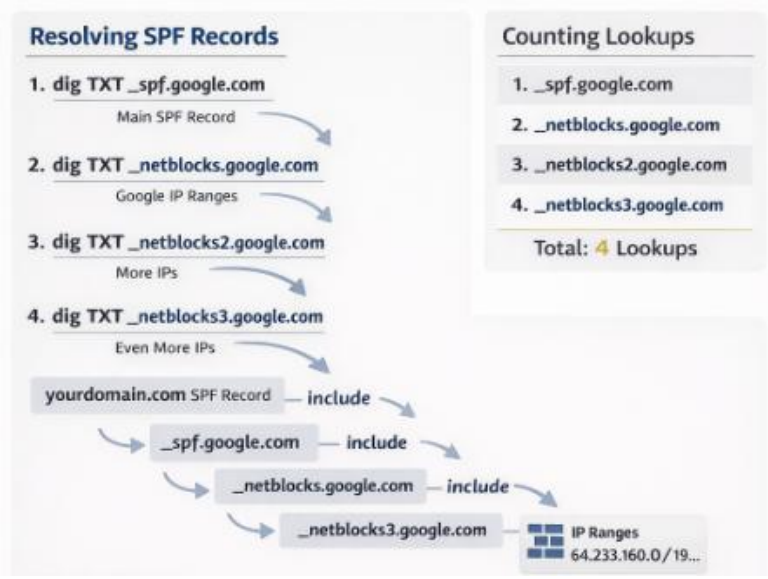


Figure 6 SPF lookup chain diagram

Lookup	Record Queried	Type	Result
1	_spf.google.com	include:	Contains IP ranges + implicit lookup to sub-records
2	_netblocks.google.com	include:	IPv4 ranges — 74.125.0.0/16, 209.85.128.0/17
3	_netblocks2.google.com	include:	IPv6 ranges — 2001:4860, 2404:6800 series
4	_netblocks3.google.com	include:	Terminal — "v=spf1 ~all" — no further lookups

3.4 Interpreting the "all" Mechanism

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

Note: google.com uses ~all (SoftFail), not -all (HardFail). This means mail from unauthorised sources is accepted and flagged rather than rejected. Combined with a DMARC p=reject policy, this is compensated for but SPF alone does not provide hard rejection..

4. DMARC Analysis

4.1 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 introduces alignment — it 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.

4.2 Querying DMARC Records

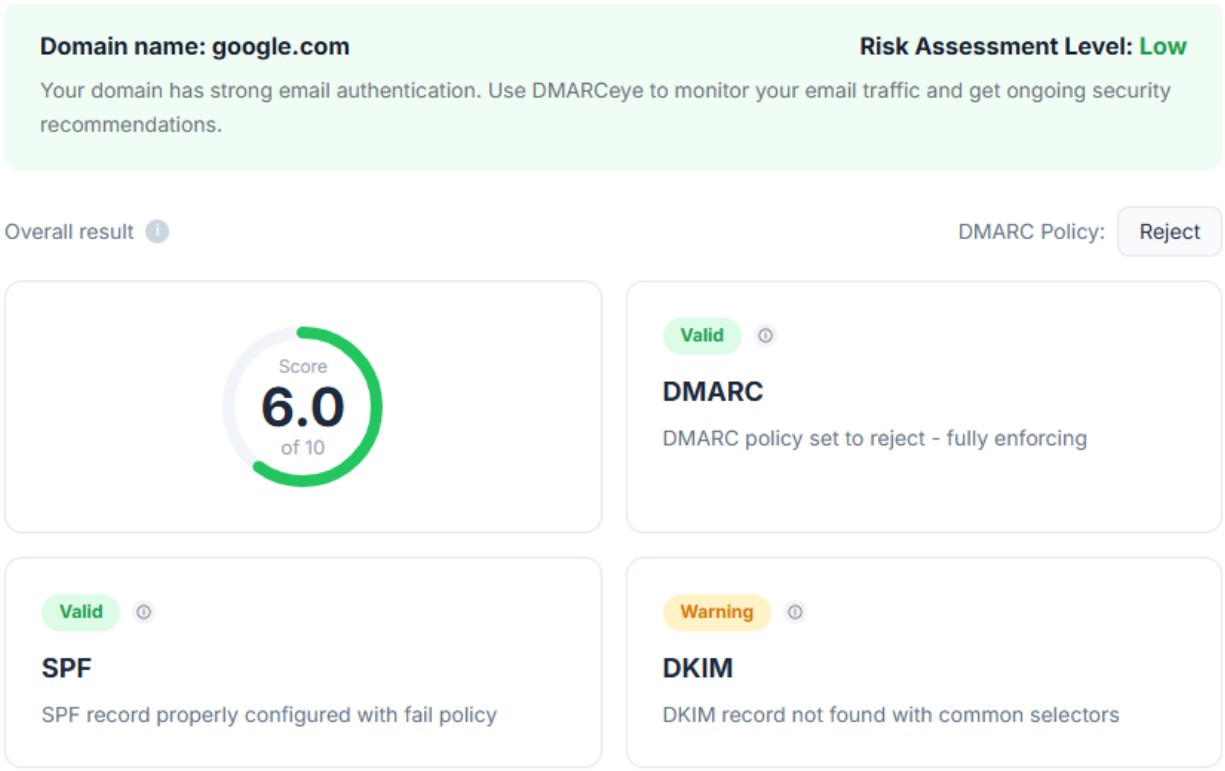
google.com

Command: 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 dig TXT _dmarc.google.com

- **p=reject** — The strictest enforcement level. Mail that fails both SPF and DKIM alignment is rejected outright at the receiving server. No exceptions.
- **rua=mailto:mailauth-reports@google.com** — Aggregate reports (summaries of all mail flows) are sent to this address. Google actively monitors their own authentication ecosystem.
- **No ruf= tag** — Forensic (per-message failure) reports are not configured. This is common for large organisations due to the privacy and volume implications of per-message reporting.
- **No pct= tag** — Defaults to pct=100, meaning the policy applies to 100% of mail.



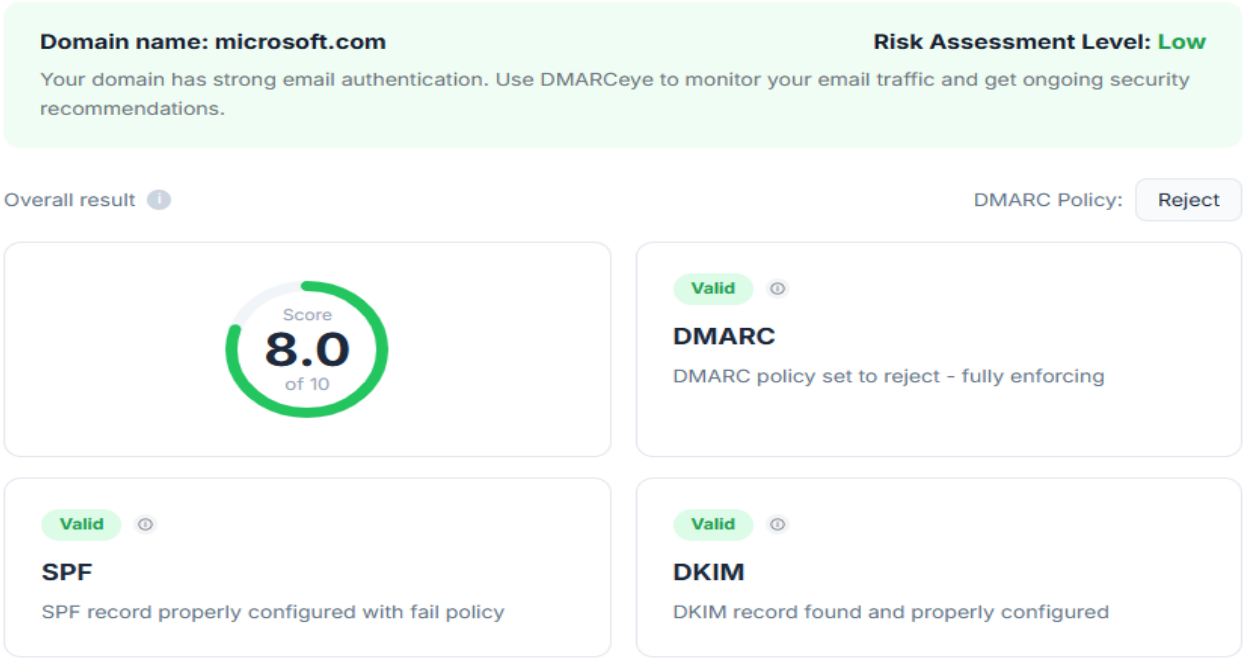
microsoft.com

Command: 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: 60938
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 65676aa49fb8ffe70100000069d483fed325597b7a34dbba (good)
;; QUESTION SECTION:
;_dmarc.microsoft.com.          IN      TXT

;; ANSWER SECTION:
_dmarc.microsoft.com.   3129    IN      TXT     "v=DMARC1; p=reject; pct=100; rua=mailto:itex-rua@microsoft.com; ruf=mailto:itex-ruf@mi
crosoft.com; fo=1"

;; Query time: 63 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 00:11:42 EDT 2026
;; MSG SIZE  rcvd: 193
```

Figure 8 dig TXT _dmarc.microsoft.com

- **pct=100** — Explicitly set. Policy applies to all messages (no phased rollout).
- **ruf=mailto:itex-ruf@microsoft.com** — Forensic reports enabled. Microsoft receives failure reports on individual messages, which aids incident response.
- **fo=1** — Failure reporting option "1": generate a forensic report when any authentication check (SPF or DKIM) fails, even if DMARC alignment as a whole passes. This maximises visibility.



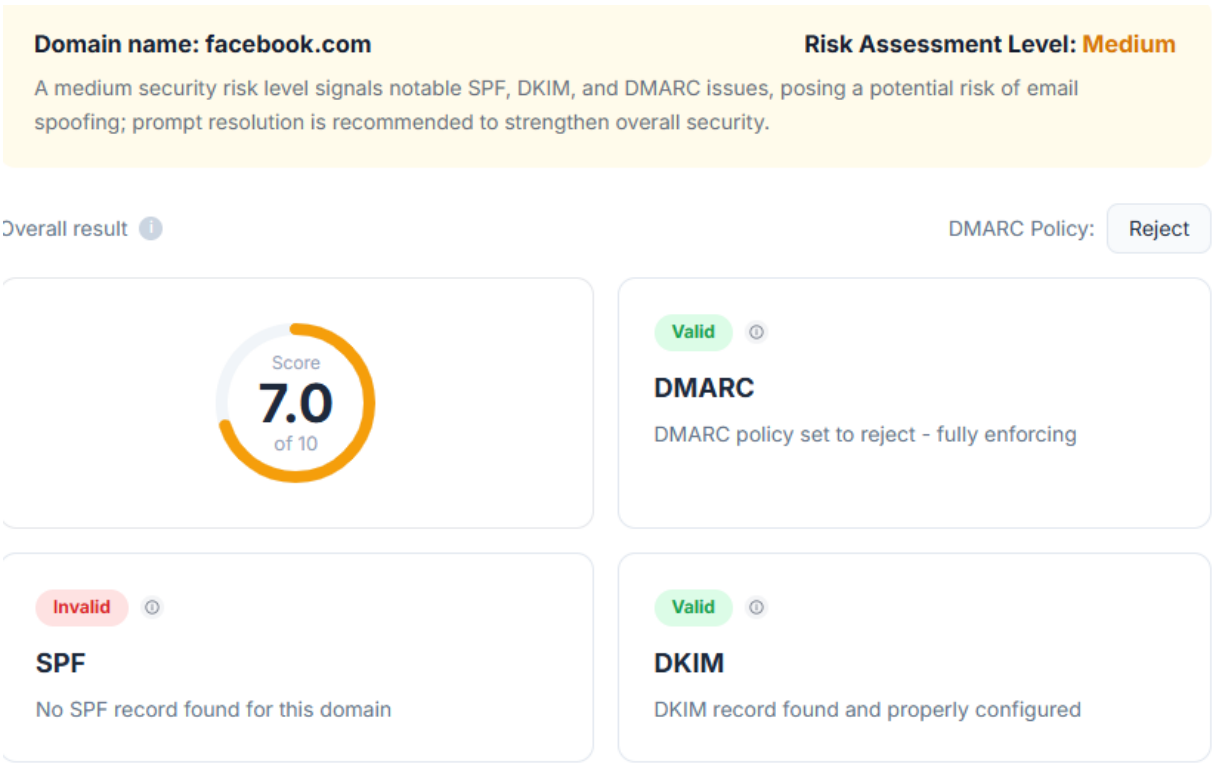
facebook.com

Command: dig TXT _dmarc.facebook.com

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

; <<>> DiG 9.20.18-1-Debian <<>> TXT _dmarc.facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25704
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 2d0624ad8d7b9a200100000069d48405f27a753a7eeb1c09 (good)
;; QUESTION SECTION:
;_dmarc.facebook.com.           IN      TXT

;; ANSWER SECTION:
_dmarc.facebook.com.    3284    IN      TXT     "v=DMARC1; p=reject; rua=mailto:a@dmarc.facebookmail.com; ruf=mailto:fb-dmarc@datafeeds
.phishlabs.com; pct=100"

;; Query time: 67 msec
;; SERVER: 192.168.100.1#53(192.168.100.1) (UDP)
;; WHEN: Tue Apr 07 00:11:49 EDT 2026
;; MSG SIZE  rcvd: 198
```

Figure 9 dig TXT _dmarc.facebook.com

Note: Facebook routes forensic DMARC failure reports to PhishLabs, a third-party threat intelligence and anti-phishing vendor. This indicates Facebook has an active programme where external specialists monitor DMARC failure data for phishing campaigns targeting the facebook.com brand.



4.3 DMARC Tag Reference

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

5. DKIM (DomainKeys Identified Mail) Validation

5.1 What Is DKIM and Why Does It Matter?

DKIM adds a cryptographic digital signature to outgoing email messages. The sending mail server signs each message with a private key. The corresponding public key is published in DNS. Receiving servers fetch the public key and use it to verify that the message signature is valid confirming both that the message came from the stated domain and that it was not tampered with in transit.

Unlike SPF, which validates the sending server's IP address, DKIM validates the message content itself. This means DKIM survives mail forwarding (where the sending IP changes) and provides non-repudiation a message with a valid DKIM signature cannot plausibly be claimed as forged.

DKIM public keys live in DNS at: <selector>._domainkey.<domain> where the "selector" is a label chosen by the sending organisation to allow multiple keys (for key rotation, different mail services, etc.).

5.2 Discovering DKIM Selectors

Unlike SPF and DMARC, there is no standard location that lists all active DKIM selectors. Selectors must be discovered by:

- Inspecting the DKIM-Signature: header of a real email from the domain (the "s=" tag contains the selector)
- Testing common selector names used by major mail platforms
- Reading MX records to infer which mail platform is in use

```
google      → for Google Workspace / Gmail
s1, s2      → generic selectors used by many providers
selector1   → Microsoft 365 / Exchange Online
selector2   → Microsoft 365 (rotation key)
default     → generic default selector
mail        → common generic selector
```

5.3 DKIM Results google.com

google._domainkey.google.com

Command: dig TXT google._domainkey.google.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 10 dig TXT google._domainkey.google.com

Status NXDOMAIN (Non-Existent Domain). The "google" selector is not published in public DNS for google.com.

Default._domainkey.google.com

Command: dig TXT default._domainkey.google.com

```
┌──(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 11 dig TXT default._domainkey.google.com

NXDOMAIN again. The "default" selector is not published either.

Mail._domainkey.google.com

Command: dig TXT mail._domainkey.google.com

```
┌──(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 12 dig TXT mail._domainkey.google.com

5.4 DKIM Results — facebook.com

s1._domainkey.facebook.com

Command: dig TXT s1._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 13 dig TXT s1._domainkey.facebook.com

NXDOMAIN. The s1 selector is not present. The AUTHORITY SECTION shows the SOA (Start of Authority) record from Facebook's nameservers (a.ns.facebook.com), confirming the query reached the correct authoritative DNS.

s2._domainkey.facebook.com

Command: dig TXT s2._domainkey.facebook.com

Figure 14 dig TXT s2._domainkey.facebook.com

NXDOMAIN. Neither s1 nor s2 selectors are present. Same conclusion as Google Facebook does not expose DKIM selectors under guessable common names.

5.5 DKIM Results — microsoft.com

Selector: selector1._domainkey.microsoft.com

Command: dig TXT selector1._domainkey.microsoft.com

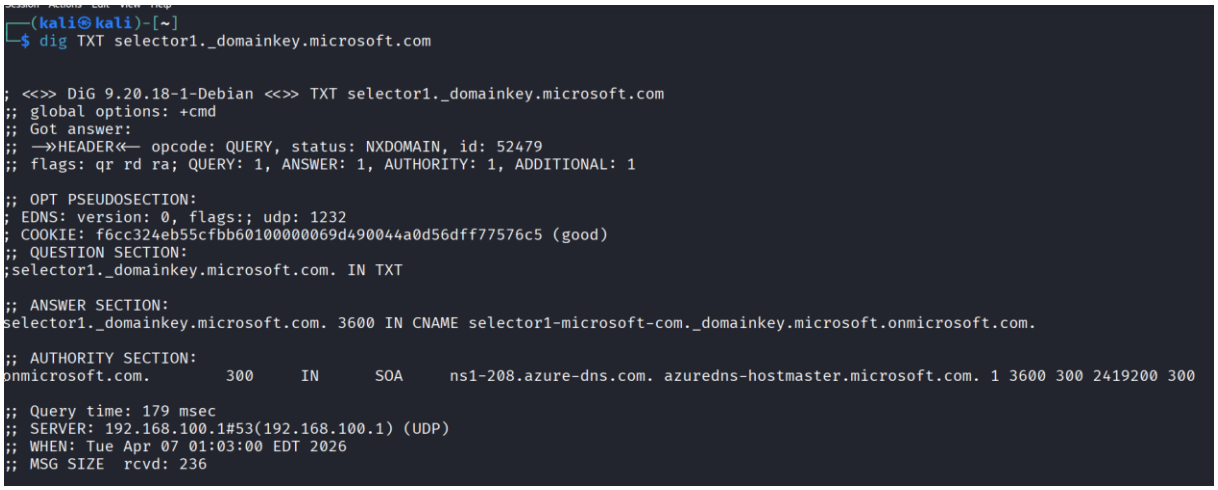


Figure 15 dig TXT selector1._domainkey.microsoft.com

CNAME to selector1-microsoft-com._domainkey.microsoft.onmicrosoft.com

This is a critically important and unique finding. Microsoft's selector1 record returned a CNAME (Canonical Name) rather than a direct TXT record. This is Microsoft 365's standard DKIM architecture:

- **CNAME delegation** — Microsoft publishes a CNAME at selector1._domainkey.<custom-domain> that points to a record under microsoft.onmicrosoft.com, which Microsoft controls.
- **Centralised key management** — By owning the target of the CNAME, Microsoft can rotate DKIM private keys at any time without requiring customers to update their DNS records. The CNAME remains static; only the underlying TXT record at microsoft.onmicrosoft.com changes.
- **Two selectors (selector1, selector2)** — Microsoft 365 provisions two selectors to support seamless key rotation: one active, one in reserve. When rotating, the reserve selector is updated and activated before the current one is decommissioned.

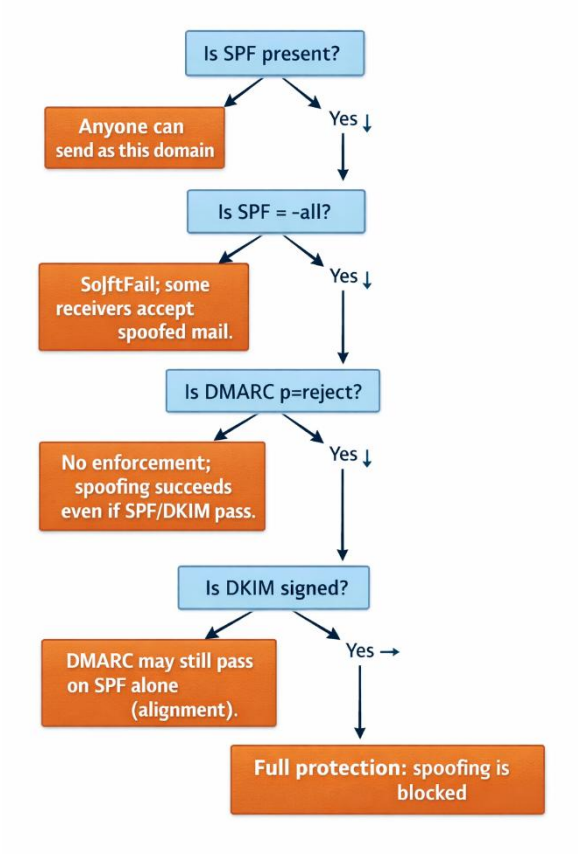
5.6 DKIM Record Field Reference

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, more efficient
p=	MIIBIjAN...	Base64-encoded public key	Empty p= means key is revoked — all mail from this selector will fail verification
t=	y	Testing mode — receivers should not reject	Remove t=y when going live
h=	sha256	Hash algorithm for signature	sha1 is deprecated; sha256 required

6. Correlation & Risk Analysis

6.1 Attacker's Decision Matrix

A threat actor attempting to spoof email from any of these domains must defeat all three mechanisms simultaneously. The following decision tree shows how an attacker would evaluate each domain:



6.2 Identifying the Weakest Mechanism

Across all three targets, the weakest element is the SPF all mechanism. Both google.com and microsoft.com use ~all (SoftFail) rather than -all (HardFail). In isolation — without DMARC — this would mean that mail from unauthorised servers gets accepted by most receivers, just with a lower reputation score. The reason this is not a critical risk here is that both domains have p=reject DMARC, which provides the hard enforcement layer that SPF's SoftFail does not.

However, this creates a dependency: if DMARC were ever misconfigured (e.g. rolled back to p=none for debugging), SPF's SoftFail would provide no meaningful protection.

Conclusion

This assessment demonstrated email security audit across three of the world's most security domains **google.com**, **microsoft.com**, and **facebook.com** using nothing more than dig on Kali Linux and passive DNS queries.

Every target has **DMARC enforced at p=reject**, which is the single most important configuration in email authentication it is what actually stops spoofing at the enforcement layer.

SPF was present and valid on all queried domains. Google's chain resolved cleanly through 4 levels of include: pointers, consuming only 4 out of the allowed 10 DNS lookups within the limit. Both Google and Microsoft use ~all (SoftFail) rather than the stricter -all (HardFail). This would be a real gap, but DMARC's p=reject compensates for it by providing the hard rejection layer that SPF SoftFail does not.

DMARC was the standout strength across all three targets. Microsoft had the most complete configuration p=reject, pct=100, both aggregate and forensic reports enabled, and fo=1 for maximum failure visibility. Facebook's use of a third-party forensic reporting vendor (PhishLabs) monitoring for phishing attempts in real time.

DKIM produced the most educational findings. All guessed selectors returned NXDOMAIN on Google and Facebook which is not a failure but a deliberate security practice. Large organisations rotate selectors frequently and do not expose them under guessable common names. Microsoft's selector1 returned a CNAME delegation to microsoft.onmicrosoft.com, which is the correct Microsoft 365 architecture it allows Microsoft to rotate private keys centrally without customers needing to touch their DNS.

None of the three is sufficient alone. SPF can be bypassed through forwarding. DKIM can be bypassed if no selector is published or the key is weak. DMARC without enforcement (p=none) provides reporting but zero protection. Only the three working together, with DMARC at p=reject, produces a genuinely spoofing resistant domain.