ATHENE
Nationales Forschungszentrum
für angewandte Cybersicherheit

Injection Attacks Reloaded: Tunneling Malicious Payloads over DNS

Philipp Jeitner and Haya Shulman

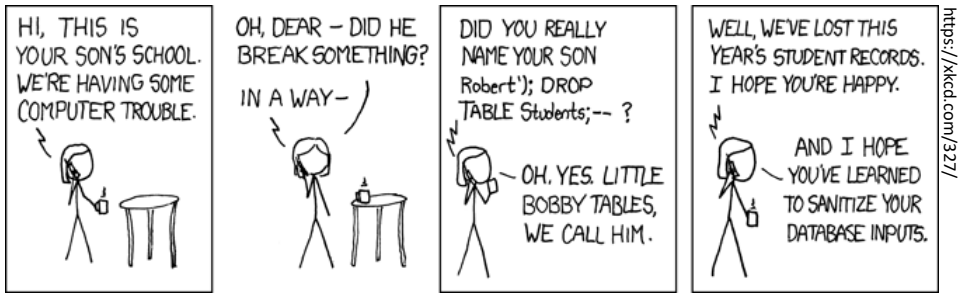
German National Research Center for Applied Cybersecurity ATHENE
Technical University of Darmstadt
Fraunhofer Institute for Secure Information Technology SIT

Motivation

“Be strict when sending and tolerant when receiving” [RFC1958]

DNS follows the general internet end-to-end principle

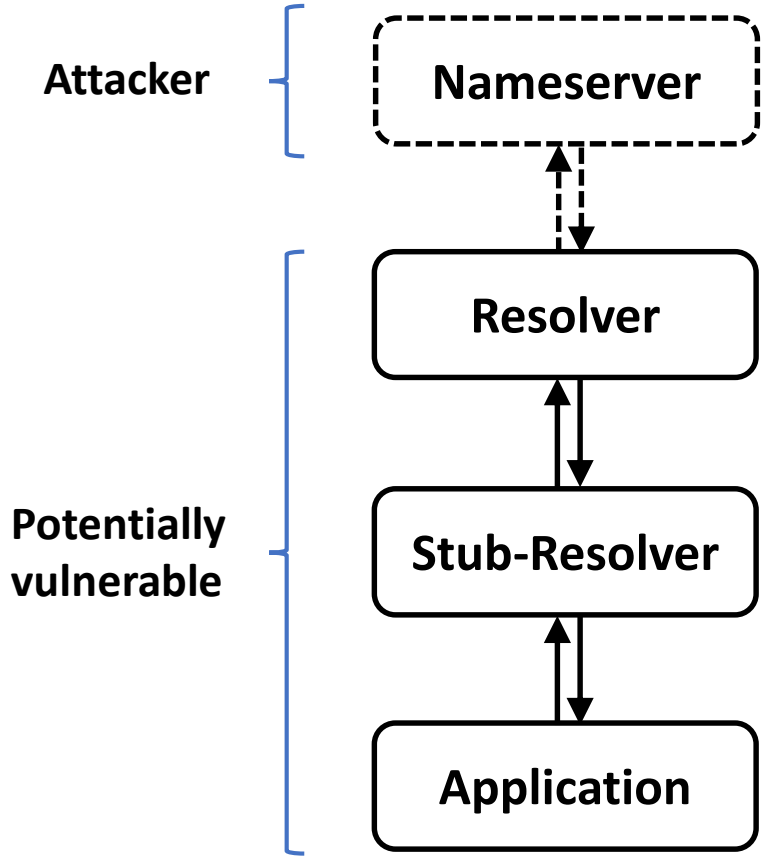
Validation must be handled by endpoints, ie. Applications



Can we abuse the DNS transparency for attacks?

Typical DNS resolution chain

1. **Application triggers a query**
   - Forwarded to the nameserver
2. **Nameserver provides record in line-format**
   - Record data can contain any value
3. **Resolver**
   - Forwards the record - treats data transparently
4. **Stub-resolvers / DNS-library**
   - Translates wire-format into textual form
5. **Application**
   - Handles the data

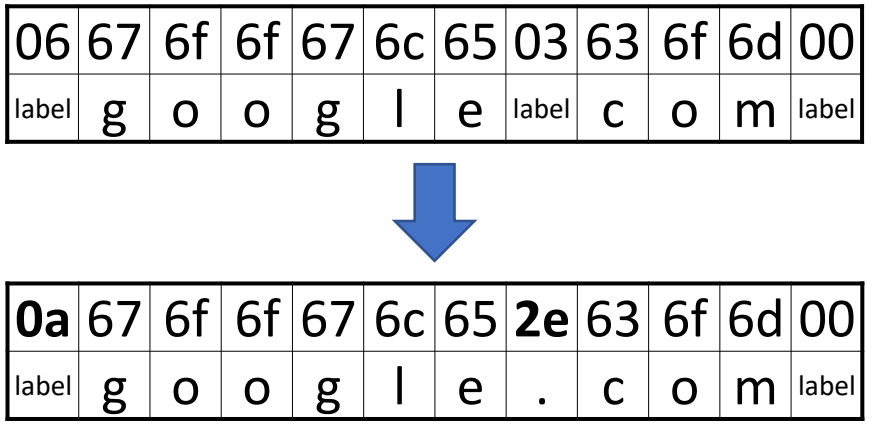


DNS resolvers

Resolvers should handle data transparently

2 Formats for domain names

- Text format: Labels separated with “.”
- Wire format: List of labels, length prepended



What happens if …

- Labels contain NULL (“\000”) ?
- Labels contain periods (“.”) ?

Attacking resolvers

```
attacker.com                    IN CNAME victim.com\000.attacker.com
victim.com\000.attacker.com     IN A     6.6.6.6

victim.com                      IN A     1.1.1.1
```

1. **Injection: Ask for attacker.com**

Record is processed, cached

2. **Validation: Ask for victim.com**

Resolver will answer with `victim.com IN A 6.6.6.6`

Vulnerable resolvers

Well-known implementations

Not vulnerable (0/11)

Public resolvers

One vulnerable (1/11, now defunct)

* Follow-up research – Not in the paper

Vulnerable resolvers

Well-known implementations

Not vulnerable (0/11)

Public resolvers

One vulnerable (1/11, now defunct)

Internet

8% vulnerable ?!

Data suggests the problem is in forwarders
- stay tuned* …

* Follow-up research – Not in the paper

Stub resolvers & Applications

Domain names vs. hostnames

- POSIX: stubs operate on “hostnames”
- Hostnames: [a-z0-9-.] [RFC952]
- **Stub-resolvers should validate!**

Do they?

- Only **1 out of 10** validates
- **7 out of 10** misinterpret zero or period

Test	Base	/	@	\.	\000	XSS	SQL	ANSI
Payload (Fig.9)	1.1.1.1	2.2.2.2	3.3.3.3	5.5.5.5	4.4.4.4	6.6.6.6	7.7.7.7	8.8.8.8
glibc	✓	✗	✗	✗	✗	✗	✗	✗
musl	✓	✓	✓	✓	✓	✓	✓	✓
dietlibc	✓	✓	✓	✓	✓	✓	✓	✓
uclibc	✓	✓	✓	✓	✓	✓	✓	✓
windows	✓	✓	✓	✓	✓	✓	✓	✓
netbsd	✓	✓	(✓)[2]	(✓)[2]	(✓)[2]	✓	✓	(✓)[2]
mac os x	✓	✓	✓	(✓)[2]	✓	✓	✓	✓
go*	✓	✓	✓	✓	(✓)[3]	✓	✓	✓
openjdk8*	✓	✗	✓	(✓)[2]	(✓)[3]	✓[4]	✓	✓
node	✓	✓	✓	(✓)[2]	✓	✓	✓	✓

✓: Vulnerable. [2]: output was escaped. [3]: Zero-byte did not stop output.
[4]: Alternative XSS payload with " " instead of " / ".
∗: Uses system stub resolver by default but offers a builtin-one.

Stub resolver test results (PTR)

Handling in applications (1)

So applications have to validate, but …

- DNS data seems to come from the OS
- application developers are not DNS developers

Do applications validate DNS input?

- **No.** None of the applications did validate.
- **8** exploitable applications

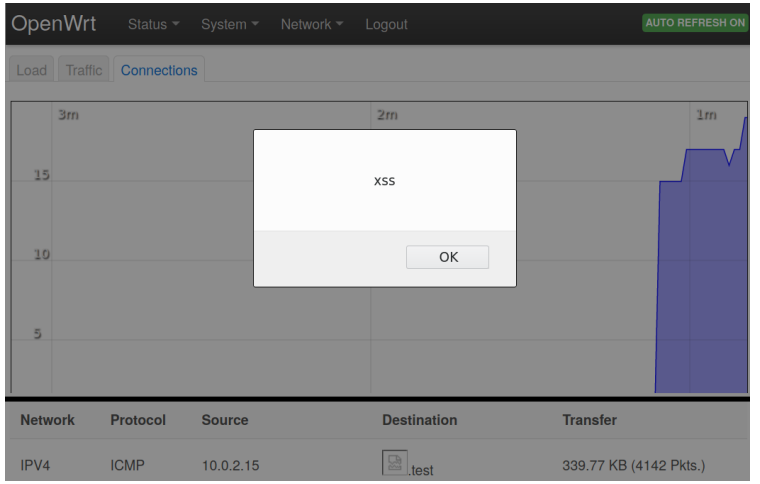
DNS Use-Case	Application	Trigger Query	Set	Uses libc	Vali-dates	Input use	Attack found
Address lookups (A, CNAME)	Chrome	js,html		yes	no	cache	no
	Firefox	js,html		yes	no	cache	no
	Opera	js,html		yes	no	cache	no
	Edge	js,html		yes	no	cache	no
	unscd	client app		yes	no	cache	no
	java	client app		both	no	cache	no
	ping(win32)	✗	✗	yes	no	display	yes
discovery (MX, SRV, NAPTR)	openjdk	login	✗	no	no	create URL	yes
	ldapsearch	login	✗	no	no	create URL	no
	radsecproxy	login		no	no	configure	yes
Reverse lookups (PTR)	ping(linux)	✗	✗	yes	no	display	yes
	trace(linux)	✗	✗	yes	no	display	yes
	OpenWRT	✗	ping	yes	no	display	yes
	openssh	login		yes	no	display,log	yes
Authentication (TXT, TLSA)	policyd-spf	SMTP		no	no	text protocol	no
	libspf2	SMTP		no	-	parse	yes
All	Resolvers	client app		no	some	cache	yes

Applications tested

Handling in applications (2)

What makes applications exploitable?

- Attacker must trigger a DNS query
- Query result used out of attack context
  - HTML, Caches, Inter-process communication …



Example vulnerabilities

- XSS in OpenWRT [CVE-2021-32019]
- Config-injection in radsecproxy [CVE-2021-32642]

Induced behaviour	**Outcome**
change dig DNS resolver	verification of vulnerability
pass `/some/file` as dig batch-file	disclose contents of `/some/file`
read `/dev/zero` as config file	100% CPU utilisation
provide malicious regex to regcomp()	radsecproxy crash
provide own RADIUS server and disable TLS-authentication	unauthorised network access

Conclusions

Standardization for stub-resolvers is lacking

- POSIX only defines “hostnames”, but no format

Missing knowledge

- Differences between DNS formats are not well known

Mitigations

- Patches: CVE-2021-{2432,32019,32642,20314,33195,3672,22931}
- Format-checking in stub-resolvers

Read the paper & test your resolver at *https://xdi-attack.net/*

Thank You!

Philipp Jeitner, TU Darmstadt/Fraunhofer SIT
philipp.jeitner@sit.fraunhofer.de

תודה רבה!

çok teşekkürler

谢谢

Merci beaucoup!

Thank you very much!

Dank je wel!

Vielen Dank!

Muchas gracias

ありがとうございました

Dziękuję!

zor spas

Grazie mille!

اشكرك