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

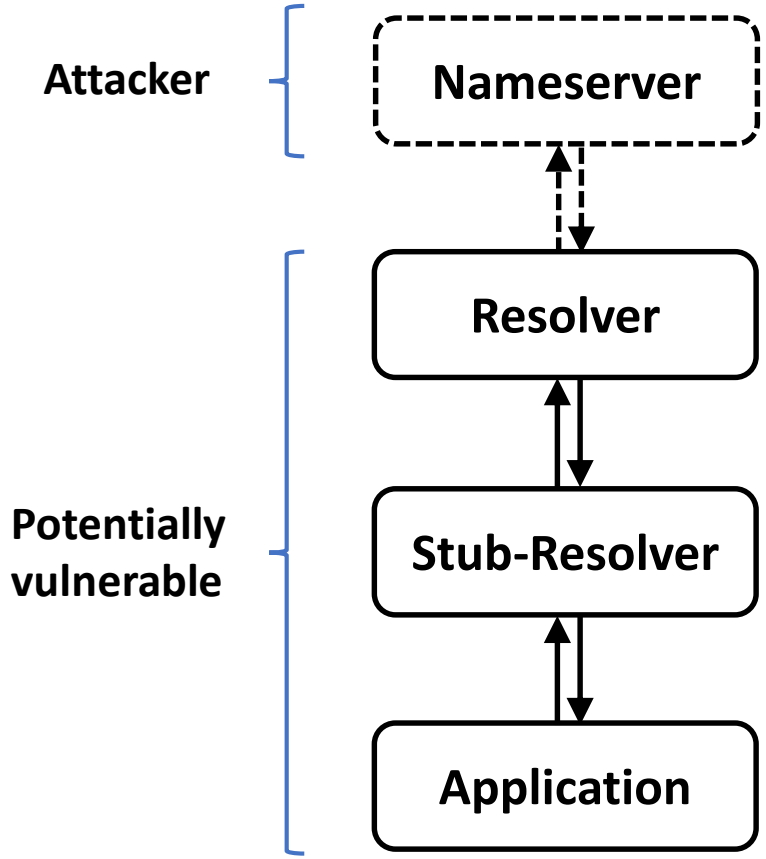
“Be strict when sending and tolerant when receiving”

[RFC1958]

- DNS follows the general internet end-to-end principle
  - Resolvers should pass unknown data unchanged [RFC3597]
- Validation must be handled by endpoints, ie. applications
  - Applications are bad at handling unexpected inputs
- **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 the line-format DNS data into textual form
5. **Application**
   - Handles the data



Handling in DNS resolvers (1)

- **Resolvers should handle data transparently**
  - so what is the problem?
- **2 Formats for domain names**
  - Text format: Labels are separated with period (.)
  - Line format: List of labels, length of each label is prepended
- **What happens if …**
  - Labels contain non-printable chars (ie. NULL)
  - Labels contain periods (.) ?

03	61	2e	62	02	3c	3e	03	63	6f	6d	00
label	a	.	b	label	<	>	label	c	o	m	label

Domain Name in **line-format**

Handling in DNS resolvers (2)

- **Do resolvers handle DNS data transparently?**
  - In-lab: **yes**
  - Public resolvers: **1 out of 11** misinterpret period-in-label
  - In-the-wild: **8%** vulnerable to cache-poisoning due to misinterpretation (zero or period)

```
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`

Resolvers tested:

In lab:
7 recursive, 4 forwarders

Public:
11 public resolvers

In-the-wild:
1.3 million open resolvers from censys dataset

Handling in stub resolvers

- **Same problems as in resolvers**
- **Domain names vs. hostnames**
  - Domain names can contain any data
    - And resolvers do not filter
  - Hostnames can only contain [a-z0-9-.] [RFC952]
  - POSIX: libc resolvers operate on hostnames
  - **So 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)

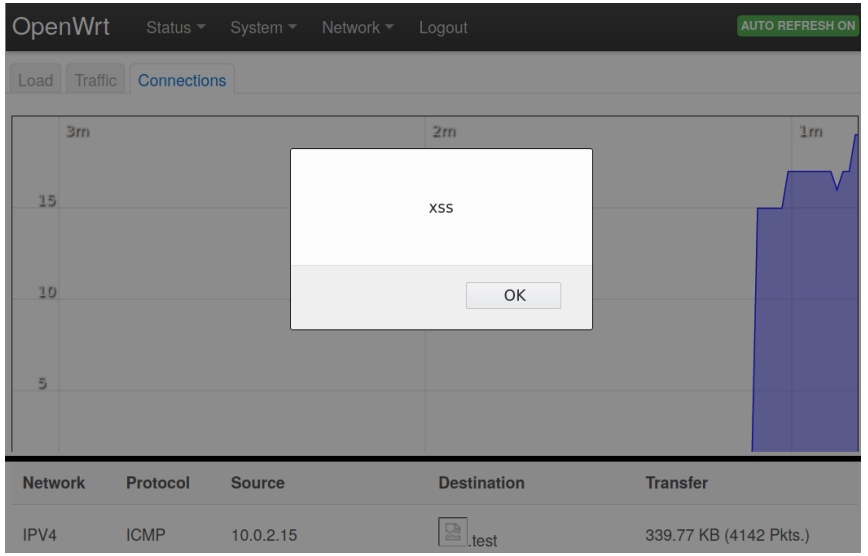
- **Stub-resolvers do not validate…**
- **So applications have to do it. However, …**
  - DNS data seems to come from the OS
    - might lead the developer to think it is trusted
  - Application developers are not DNS developers
    - they might not be aware that DNS records can contain any value
- **So do applications validate DNS input?**
  - **No.** None of the applications did validate.
  - **8** applications have vulnerabilities due to this: XSS, Stack overflow, Buffer Overflow, Config injection, …

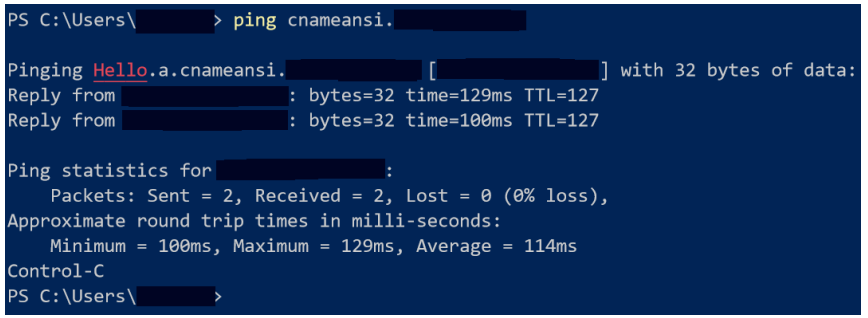
DNS Use-Case	Application	Trigger Query	Set	Uses libc	Validates	Input use	Attack found
Address lookups (A, CNAME)	Chrome	js,html		yes	no	cache	no
	Firefox	js,html		yes	no	cache	no
	Opera	js,html		yes	no	cache	no
	Edge	js,html		yes	no	cache	no
	unscd	client app		yes	no	cache	no
	java	client app		both	no	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 vulnerable?**
  - Attacker must trigger a DNS query
  - Query result must be used somewhere where injection attacks are possible
    - HTML
    - Caches
    - Inter-process communication
    - …
- **Example vulnerabilities** (right)
  - XSS in OpenWRT
  - ANSI escape code injection into ping





Conclusions

- **Misinterpretations and wrong processing all the way**
- **Standardization for stub-resolvers is lacking**
  - POSIX only defines “hostnames”, but no format
- **Missing knowledge**
  - Differences between DNS line format, text format and hostnames are not widely known
  - Developers are not are that input from DNS is untrusted
- **Mitigations:**
  - Fix application vulnerabilities: CVE-2021-20314, CVE-2021-32019
  - Stub-resolver developers agree to apply hostname checking
  - Resolvers: 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