INTRODUCTION TO HACKING

VINCENZO CIASCHINI

CNAF/CERN TALKS

10/6/2020

SUMMARY

- Introduction
- Caveats
- Instructions

WHAT IS HACKING?

- In this context, is testing.
  - Testing applications to determine the presence (or absence) of security issues

WHY?

- Because applications are complex.
  - Many small moving parts (dependencies, protocols, apis, interfaces, services, etc…) Because security is complex.
- Because security is complex.
  - In part because of application complexity.
  - But also as a consequence of developer's lack of focus on security.
    - Including also knowledge of ways of attack.
- It is also system property.
  - Or a single unsecure component or setup which breaks everything else.
    - Like a root:root username passwords (or admin:admin)

WHY? (CONT.D)

- Security issues are bugs and can be found during testing!
  - Yes, but…
- Testing usually deals with functional issues.
  - Can be done with a checklist.
- Testing for security issues requires specialized knowledge, experimentation and constant studying
  - There are no useful checklists. Exploits are very finicky.

WHY (FINAL)

- But why spend time, effort and resources in it? What's the worst that could happen?
  - Answer: Disruption of service, Loss of money, Loss of reputation, Penal responsibilities…
  - See Stefan's slides for more details

CAVEATS

- Things to watch out for

CAVEAT 1

- Hacking your way in a system is **ILLEGAL!**
  - If done without proper authorization.
  - Hacking random services is a penal crime under most (all?) jurisdictions.
  - So be sure to always be authorized to perform whatever tests you want.
    - And follow instructions on what you are or are not authorized to do.
    - E.g: The application has a database. Can you attempt deleting it?

CAVEAT 2

- Hacking is not a riskless activity.
  - It is an exploratory activity. If the bugs were known, and the consequences understood, they would not be there
  - Testing can crash a service, delete data, or make it unusable.
    - Even with apparently harmless exploits.
    - Even when you do not want to cause damage.

CAVEAT 3

- Privacy
- It is possible that a successful attack would expose a user's private information.
  - If at all possible, try to attack your own user.
  - Or a user who has explicitly given you permission to do so (and keep documentation of that permission)
  - Seek advice from the administrators in case of doubt
    - They may even create dummy users for you to attack.

INSTRUCTIONS

- Basic steps

HOW TO?

- So, how to do hacking?
- First, gather information.
- Second, study the environment.
- Third, study known attacks
- Fourth, attack

1. GATHER INFORMATION

- Is there a manual for the program/service you intend to attack?
- Is there any kind of documentation?
- Sources available?
- Dependencies are discoverable?
- Maybe a version you can install and test yourself?

IS THERE A MANUAL?

- A manual is useful because it will explicitly document all user-facing (or admin-facing) endpoints where input can be given.
- It may also explain that what appears as a single program is a set of programs working together
  - More targets! Possible communication weakpoints!

DOCUMENTATION IN GENERAL

- All kinds of documentation is useful.
- It may list dependencies.
- It may document deployment choices
- It may tell you which programming language is used

SOURCES

- The sources tell the internal workings of the application.
- It may be used as a guide to exploitation.
- Can describe endpoints that are otherwise non-discoverable.
- Can give you ideas on what to attack.
  - E.g: If there is the concepts of "users" and "permissions" the code implementing it is especially interesting

DEPENDENCIES

- (Almost) No program is self-contained.
- They all depend on libraries.
- Libraries themselves may have bugs. Maybe even *known* bugs. Maybe with *known* exploits.

SELF-CONTAINED VERSION AVAILABLE FOR PERSONAL USE?

- Can perform all kinds of testing on your own machine without needing permission.
  - (but make sure it is REALLY self-contained)

STUDY THE ENVIRONMENT

- Very few things work by themselves.
  - Applications run on an operating system.
  - Web services run on web server.
  - Plugins run on a framework.
  - Maybe there is a firewall?
- This are all potentially vulnerable.
  - And therefore, they are all of interest.
  - And they may all impact the service you are testing.

STUDY THE ENVIRONMENT

- Discover the OS (nmap –O).
- Discover the framework (Apache? Tomcat? Wordpress?)
- Check for services running on the server (nmap)

- For each object found, check for known vulnerabilities (google, www.exploit-db.com, cvedetails.com, etc…)
  - And test them! (warning! Some exploits may need to be rejiggled)

https://www.cvedetails.com/vulnerability-list/vendor_id-10048/product_id-17956/Nginx-Nginx.html

CVE Details

The ultimate security vulnerability datasource

(e.g.: CVE-2009-1234 or 2010-1234 or 20101234) Search View CVE

Log In Register

Vulnerability Feeds & Widgets New www.itsecdb.com

Home
Browse :
Vendors
Products
Vulnerabilities By Date
Vulnerabilities By Type
Reports :
CVSS Score Report
CVSS Score Distribution
Search :
Vendor Search
Product Search
Version Search
Vulnerability Search
By Microsoft References
Top 50 :
Vendors
Vendor Cvss Scores
Products
Product Cvss Scores
Versions
Other :
Microsoft Bulletins
Bugtraq Entries
CWE Definitions
About & Contact
Feedback
CVE Help
FAQ
Articles
External Links :
NVD Website
CWE Web Site
View CVE :
Go
(e.g.: CVE-2009-1234 or 2010-1234 or 20101234)
View BID :
Go
(e.g.: 12345)
Search By Microsoft Reference ID:
Go
(e.g.: ms10-001 or 979352)

Nginx » Nginx : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending
Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2018-16845	835			2018-11-07	2019-10-02	5.8	None	Remote	Medium	Not required	Partial	None	Partial
nginx before versions 1.15.6, 1.14.1 has a vulnerability in the ngx_http_mp4_module, which might allow an attacker to cause infinite loop in a worker process, cause a worker process crash, or might result in worker process memory disclosure by using a specially crafted mp4 file. The issue only affects nginx if it is built with the ngx_http_mp4_module (the module is not built by default) and the .mp4. directive is used in the configuration file. Further, the attack is only possible if an attacker is able to trigger processing of a specially crafted mp4 file with the ngx_http_mp4_module.														
2	CVE-2018-16844	400			2018-11-07	2019-09-10	7.8	None	Remote	Low	Not required	None	None	Complete
nginx before versions 1.15.6 and 1.14.1 has a vulnerability in the implementation of HTTP/2 that can allow for excessive CPU usage. This issue affects nginx compiled with the ngx_http_v2_module (not compiled by default) if the 'http2' option of the 'listen' directive is used in a configuration file.														
3	CVE-2018-16843	400			2018-11-07	2019-09-10	7.8	None	Remote	Low	Not required	None	None	Complete
nginx before versions 1.15.6 and 1.14.1 has a vulnerability in the implementation of HTTP/2 that can allow for excessive memory consumption. This issue affects nginx compiled with the ngx_http_v2_module (not compiled by default) if the 'http2' option of the 'listen' directive is used in a configuration file.														
4	CVE-2017-7529	190		Overflow +Info	2017-07-13	2018-01-04	5.0	None	Remote	Low	Not required	Partial	None	None
Nginx versions since 0.5.6 up to and including 1.13.2 are vulnerable to integer overflow vulnerability in nginx range filter module resulting into leak of potentially sensitive information triggered by specially crafted request.														
5	CVE-2016-4450			DoS	2016-06-07	2018-01-04	5.0	None	Remote	Low	Not required	None	None	Partial
os/unix/ngx_files.c in nginx before 1.10.1 and 1.11.x before 1.11.1 allows remote attackers to cause a denial of service (NULL pointer dereference and worker process crash) via a crafted request, involving writing a client request body to a temporary file.														
6	CVE-2016-1247	59		+Priv	2016-11-29	2018-10-09	7.2	Admin	Local	Low	Not required	Complete	Complete	Complete
The nginx package before 1.6.2-5+deb8u3 on Debian jessie, the nginx packages before 1.4.6-1ubuntu3.6 on Ubuntu 14.04 LTS, before 1.10.0-0ubuntu0.16.04.3 on Ubuntu 16.04 LTS, and before 1.10.1-0ubuntu1.1 on Ubuntu 16.10, and the nginx ebuild before 1.10.2-r3 on Gentoo allow local users with access to the web server user account to gain root privileges via a symlink attack on the error log.														
7	CVE-2016-0747	399		DoS	2016-02-15	2018-10-30	5.0	None	Remote	Low	Not required	None	None	Partial
The resolver in nginx before 1.8.1 and 1.9.x before 1.9.10 does not properly limit CNAME resolution, which allows remote attackers to cause a denial of service (worker process resource consumption) via vectors related to arbitrary name resolution.														
8	CVE-2016-0746			DoS	2016-02-15	2018-10-30	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
Use-after-free vulnerability in the resolver in nginx 0.6.18 through 1.8.0 and 1.9.x before 1.9.10 allows remote attackers to cause a denial of service (worker process crash) or possibly have unspecified other impact via a crafted DNS response related to CNAME response processing.														
9	CVE-2016-0742			DoS	2016-02-15	2018-10-30	5.0	None	Remote	Low	Not required	None	None	Partial
The resolver in nginx before 1.8.1 and 1.9.x before 1.9.10 allows remote attackers to cause a denial of service (invalid pointer dereference and worker process crash) via a crafted UDP DNS response.														
10	CVE-2014-3616	284			2014-12-08	2014-12-08	4.3	None	Remote	Medium	Not required	None	Partial	None
nginx 0.5.6 through 1.7.4, when using the same shared ssl_session_cache or ssl_session_ticket_key for multiple servers, can reuse a cached SSL session for an unrelated context, which allows remote attackers with certain privileges to conduct "virtual host confusion" attacks.														
11	CVE-2014-3556	77			2014-12-29	2015-03-16	4.3	None	Remote	Medium	Not required	Partial	None	None
The STARTTLS implementation in mail/ngx_mail_smtp_handler.c in the SMTP proxy in nginx 1.5.x and 1.6.x before 1.6.1 and 1.7.x before 1.7.4 does not properly restrict I/O buffering, which allows man-in-the-middle attackers to insert commands into encrypted SMTP sessions by sending a cleartext command that is processed after TLS is in place, related to a "plaintext command injection" attack, a similar issue to CVE-2011-0411.														
12	CVE-2014-0133	119		Exec Code Overflow	2014-03-28	2018-10-30	5.1	None	Remote	High	Not required	Partial	Partial	Partial

STUDY KNOWN ATTACK TECHNIQUES

- If you do not know something is possible, you cannot test for it.
- E.g.: you can inject code in input fields, you can force a user to do a request, you can alter the messages an application uses to communicate, etc…
- Good places to keep informed a r/hacking, r/pentest, seclists.org/fulldisclosure, seclists.org/oss-sec, many other you will find in time.

FINALLY, ATTACK

- Now you can try attacking the application. How?
- First if you have identified a set of known vulnerabilities in the previous steps, try them.
  - If they at first don't work, alter them if necessary and try again.

FINALLY, ATTACK (CONT.D)

- (Following examples will assume a web application)
- Now, try to attack the application proper.
- All input fields are interesting targets
  - There are more fields than readily apparent

- Are there forms in the application?
  - How about inputing: <script>alert(1);<script> ?
  - Or: "; ls ?
  - Or: ' OR 1==1, --
  - Or: '; DROP DATABASE; COMMIT; ?
  - Or: http://other.victim.com/ ?
  - Or any combination thereof?
  - Or variants ?
  - If the application refuses a value how about bypassing it and specifying it directly?
- Also, are there hidden fields?
  - Look at the html sources to find them
  - What happens if you change the value?

- Other "hidden" fields…
  - The URL!
    - Try adding /../etc
      - Or alter the path in any other way
      - Good candidates are also log/, admin/ …
    - There are several cases where unintended paths are exposed
  - Cookies!
    - What are they? What they do contain? What if you change them?

UNINTENDED USAGE

- Think about what a feature does, not what is *supposed* to do.
  - E.g.: Avatar upload form -> To upload an image
  - What about uploading some other type of file?
  - Interesting "image" : `<?php system($_GET['cmd']); ?>`
  - Even more interesting one…: `GIF89a<?php system($_GET['cmd']); ?>`
  - Call them with: http://victim.url.com/uploaded/file.gif?cmd=/sbin/reboot

- You have the sources?
  - Read them.
  - In particular, read sections where user privileges are checked.
    - Can you control any of the data used?
    - If so, try to alter the values to obtain new privileges.

- Best of all…
  - Are you even sure you need to go through the application? Or its checks?
  - Can you bypass any?
    - If one application acts as a "gateway" to another, can you contact the second directly?

REPORTING

- So, you have found some vulnerability… what to do?
- Report it to the service owner.
- DO NOT write a blog article (unless given permission by the service owner)

HOW TO GET BETTER?

- Practice, practice, practice!
- No other way.

- There are many online services that help with that:
  - Google gruyere, vulnhub, hackthebox, etc…