UNDERSTANDING THE KNOWN

A9 : USING COMPONENTS WITH KNOWN VULNERABILITIES

BY

ANANT SHRIVASTAVA

ANANT SHRIVASTAVA

- Information Security Consultant
- Admin - Dev - Security
- null + OWASP + G4H
- http://anantshri.info and @anantshri
- Trainer / Speaker : Blackhat USA, NullCon, g0s, c0c0n, Clubhack, RootConf

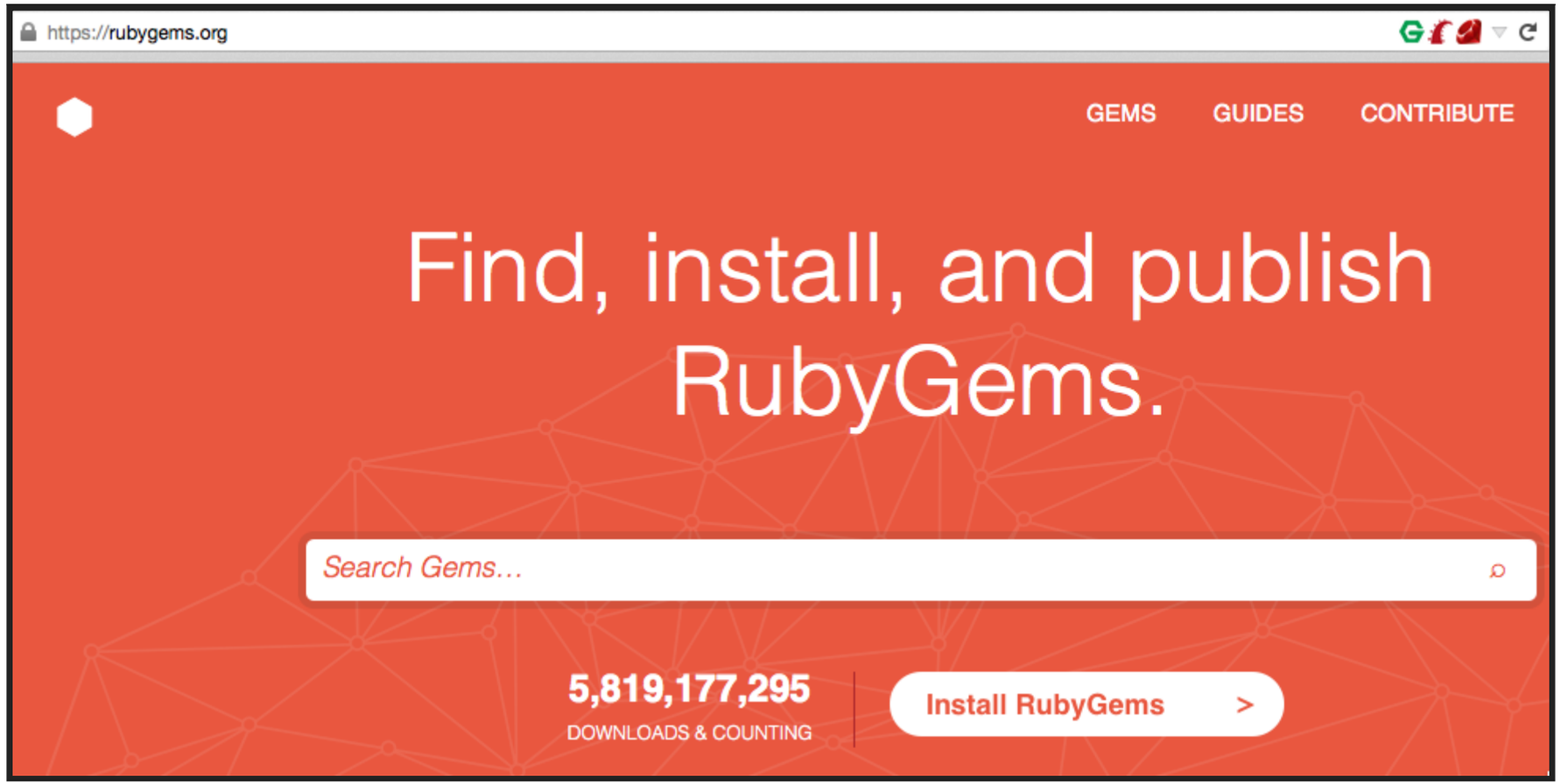
WHAT IS A COMPONENT

- Any piece of code that is reusable
- Paid or OpenSource
- Either by same developer or other developers
- Its lot more then what you know

PYTHON PACKAGES

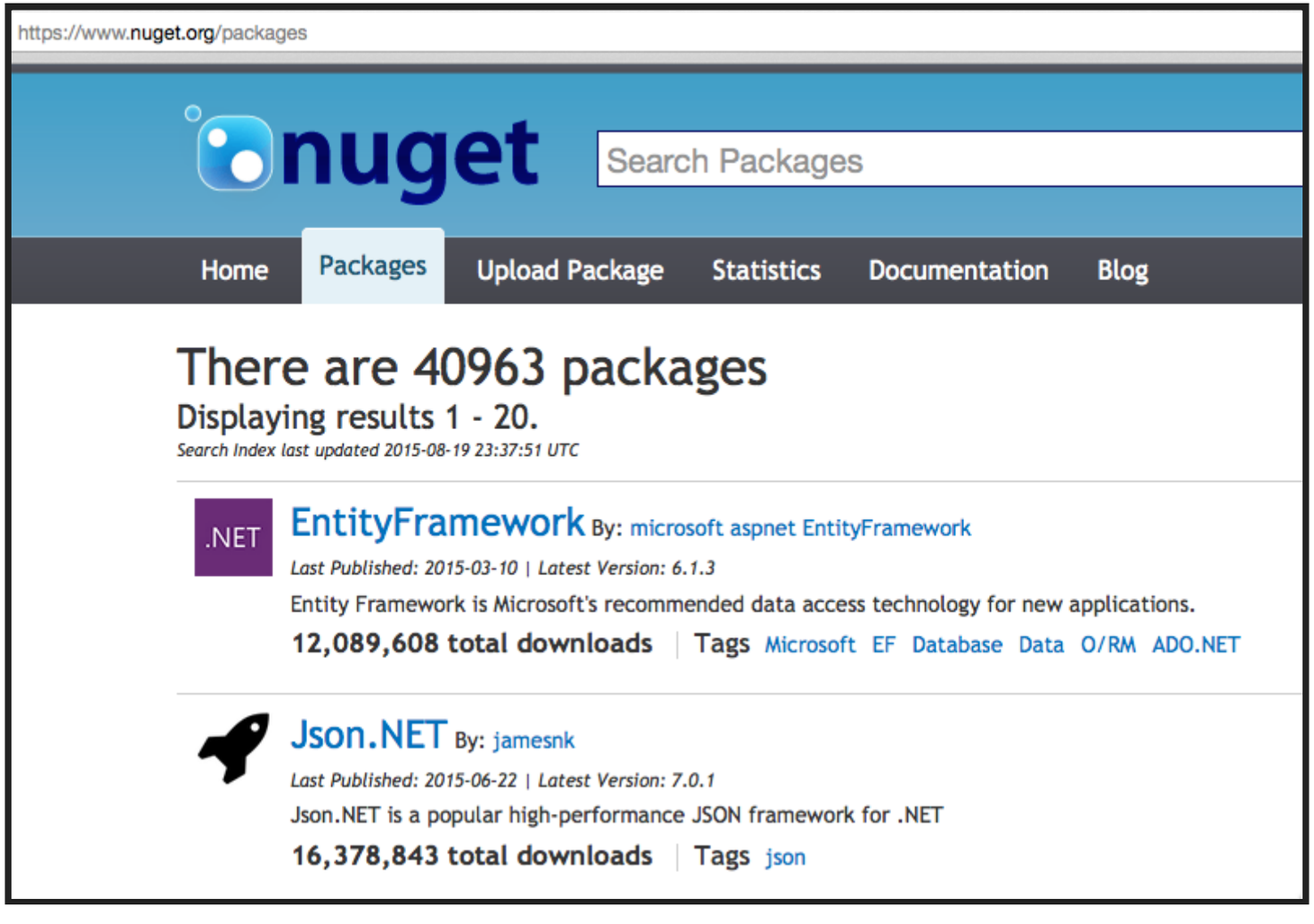
Programming Language

RUBY GEMS



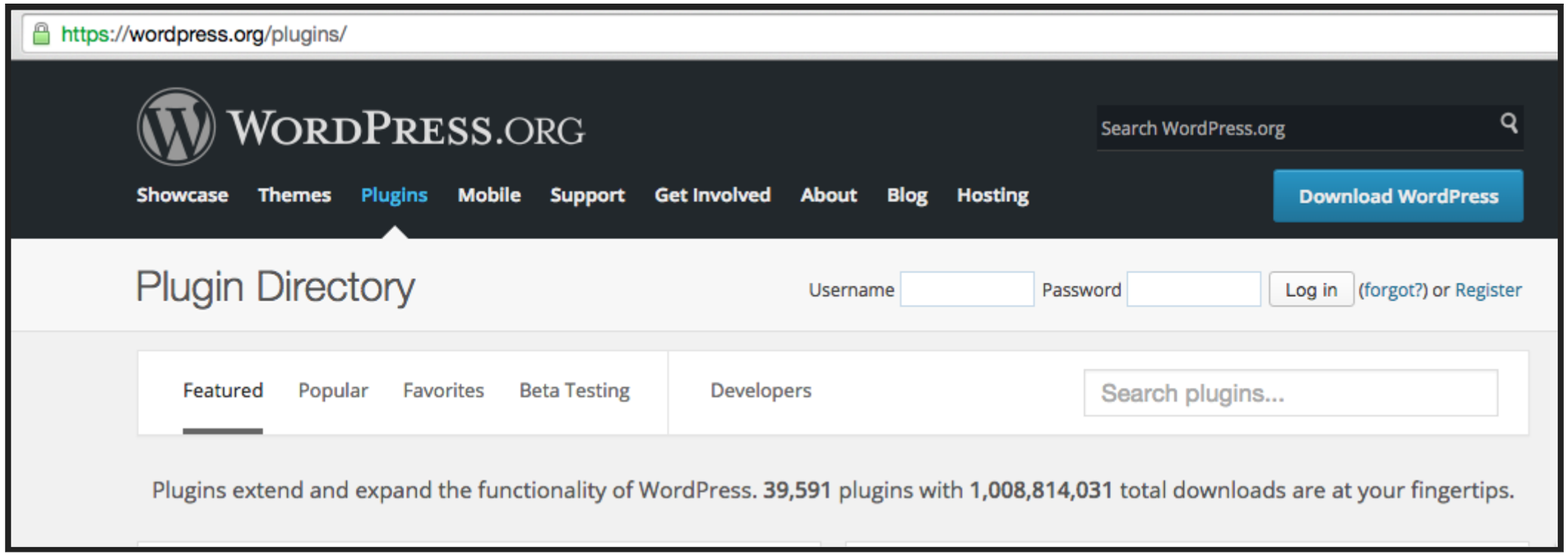
Programming Language

MICROSOFT .NET PACKAGES



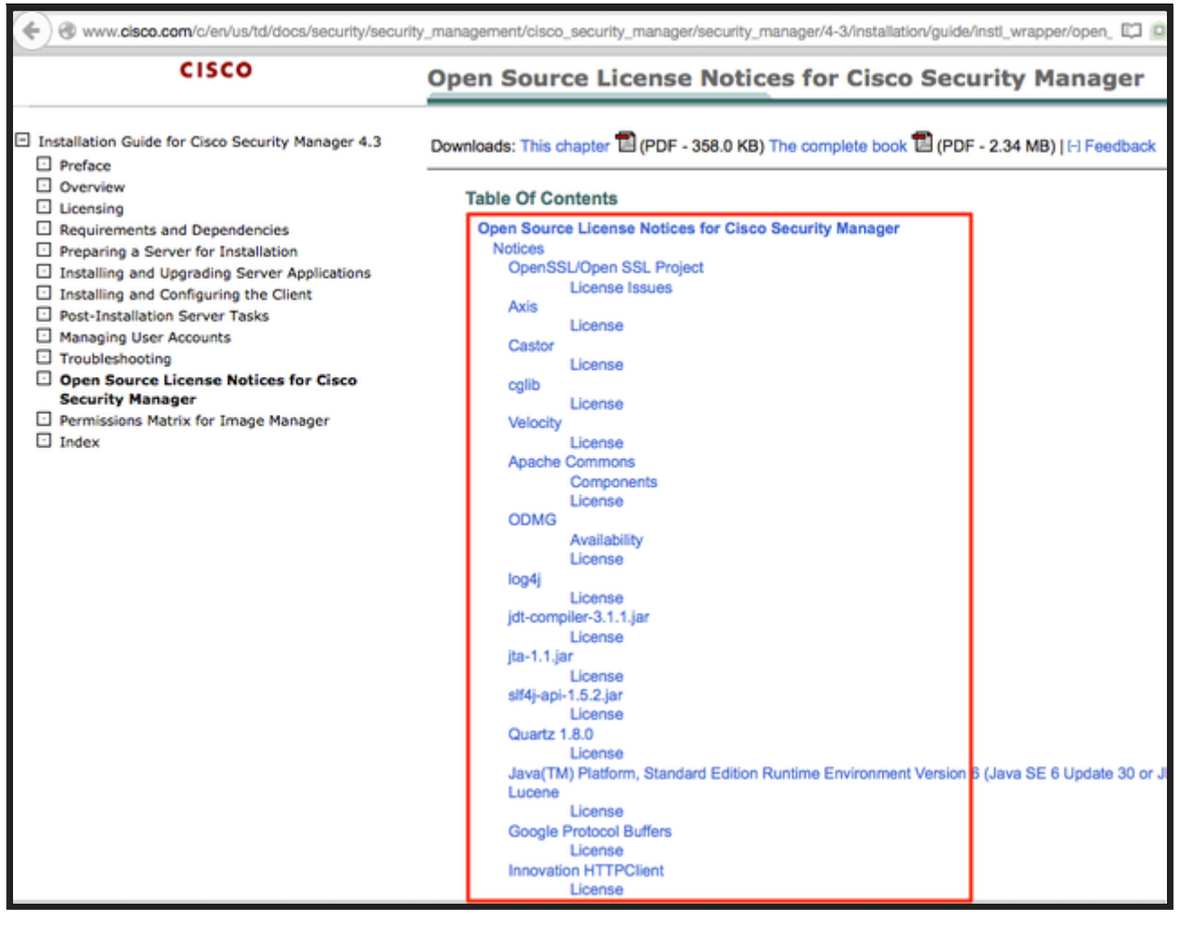
Programming Language

WORDPRESS PLUGINS



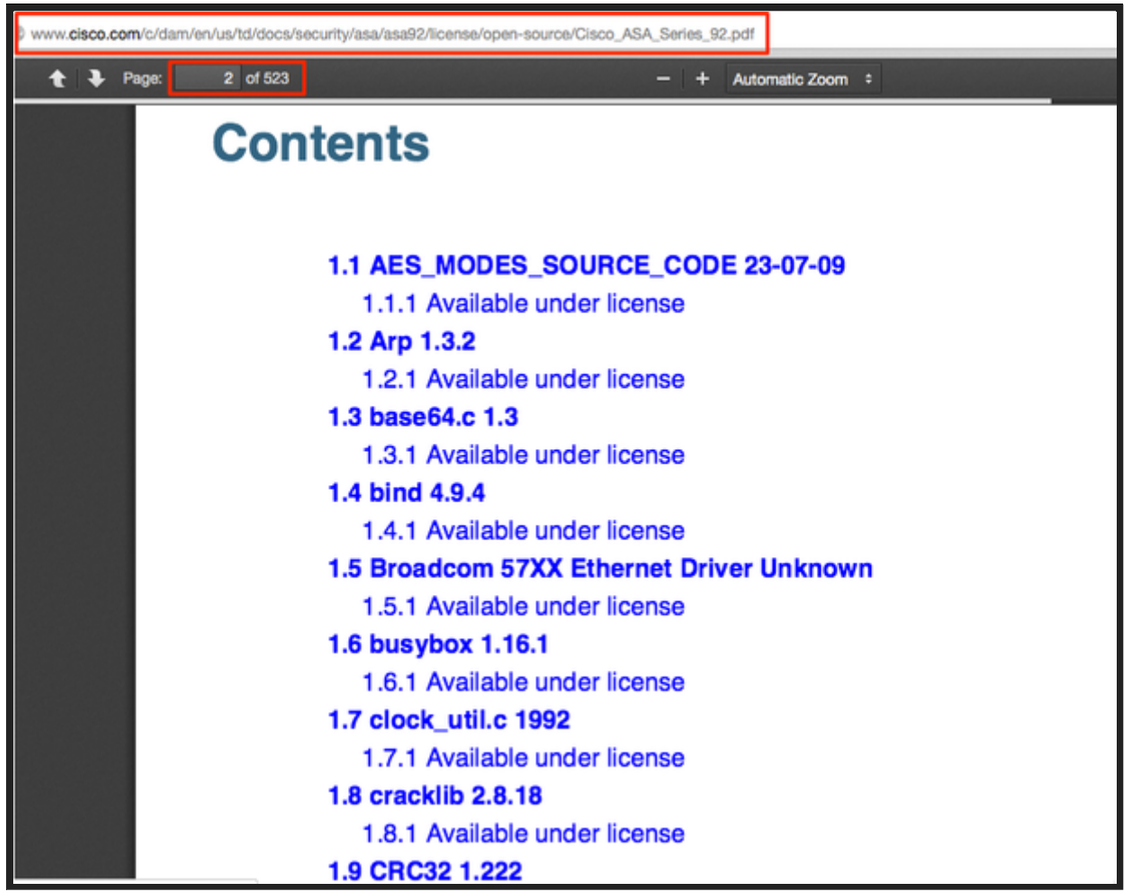
Web Application

CISCO SECURITY MANAGER



Cisco Security Manager

CISCO ASA



Cisco ASA Hardware

AND MANY MORE

WHY COMPONENTS

- Unix Philosophy : **Do one thing and do it well**
- Code Reuse : "**Less Development Overhead**"
- "**Potentially**" Combined and Faster evolution
- Higher cost to develop from scratch

IN SHORT

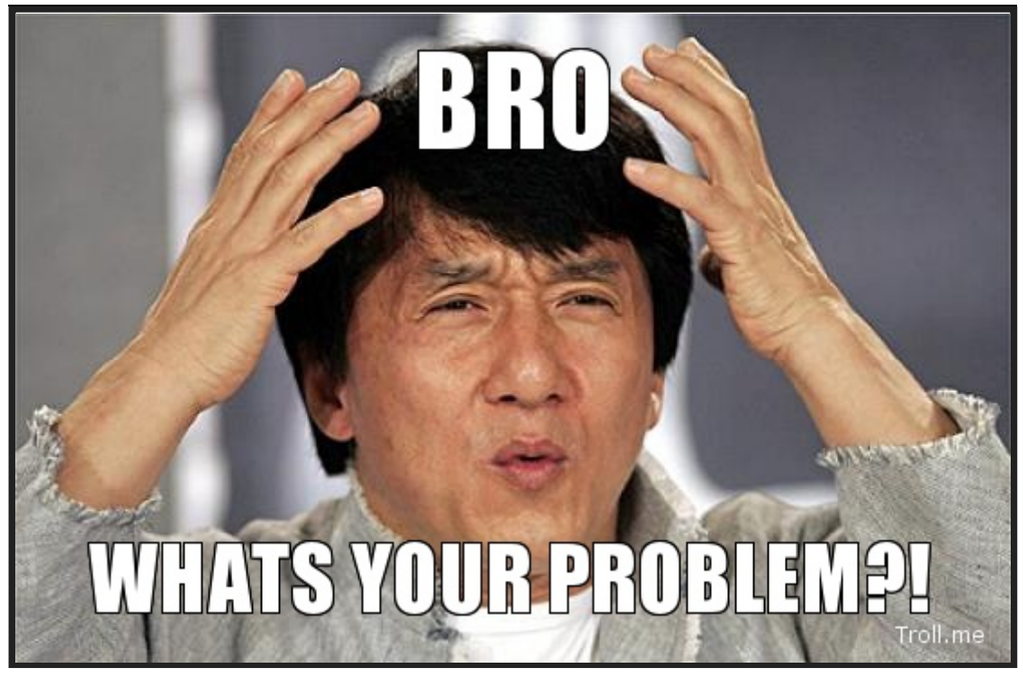
Any component which is not developed by you is a 3rd party package in use

NOT DEVELOPED BY YOU

NOT DEVELOPED BY YOU

1. OpenSSL
2. Bash
3. Apache
4. NGINX

and many more


BRO
WHATS YOUR PROBLEM?!
Troll.me

UNDERSTANDING THE KNOWN USING COMPONENTS WITH KNOWN VULNERABILITIES

TWO DISTINCT PROBLEMS

1. Component has known vulnerability
2. Licensing Policies

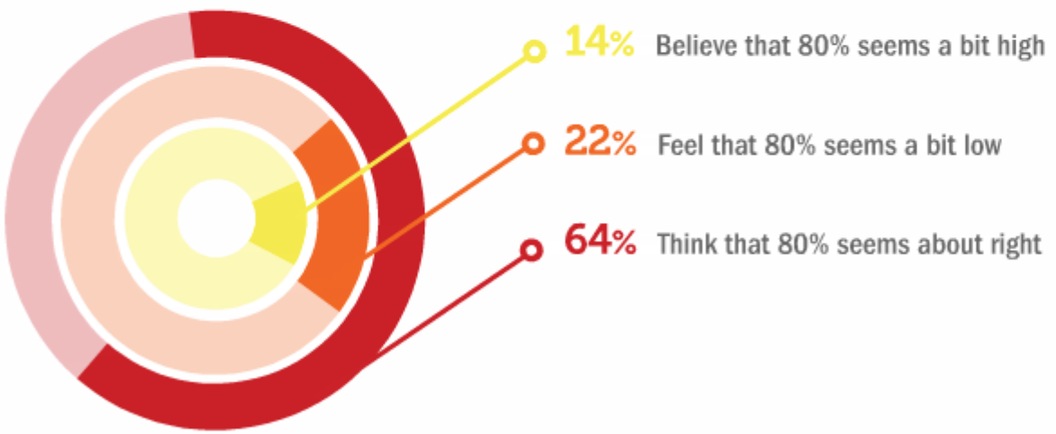
Talk focus only on the first part

COMPONENT WITH KNOWN VULNERABILITY

- Marked as 9/10 in OWASP Top 10 Vulnerabilities in 2013
- Attacks can range from basic web attacks to Remote Code Execution

At least 80% of a typical java application is assembled from open source components and frameworks.

Question: Would it surprise you to know that 80% of a typical Java application is now assembled from open source components and frameworks?



Yes, It's True!

Many of the applications you use today are now assembled from hundreds of open source components.

Source: 2013 Sonatype OSS survey of 3,500 developers, architects and managers

SOME EXAMPLES

HEARTBLEED

DOMAIN	VULNERABLE SITES	SAFE SITES	TOTAL NO. OF SITES USING SSL	TOTAL NO. OF SITES	PERCENTAGE
KR	57	45	102	2839	56%
JP	534	661	1195	17852	45%
RU	2708	3590	6298	38573	43%
CN	66	98	164	10430	40%
GOV	26	43	69	829	38%
BR	866	1782	2648	16328	33%
AU	553	1190	1743	7911	32%
UK	1073	2692	3765	19062	28%
DE	1544	4780	6324	34275	24%
FR	594	2474	3068	13033	19%
IN	611	2851	3462	13204	18%
Total	8632	20206	28838	174336	30%

TREND MICRO

VULNERABLE VENDOR

The Document Foundation	Open SSL	Isode Ltd	Extreme Networks, Inc.
Splunk Inc.	Red Hat, Inc	OpenBSD	Barracuda Networks, Inc.
IBM Corporation	Cerberus, LLC	ADTRAN, Inc.	NoMachine S.à r.l.
BalaBit IT Security	Michal Trojnara	F-Secure Corporation	Rapid7
VMware, Inc.	Netwin Ltd.	McAfee, Inc.	Huawei Technologies Co., Ltd.
ABB	Juniper Networks, Inc.	Novell, Inc.	Digi International Inc.
Certec EDV GmbH	Aruba Networks, Inc.	Fortinet, Inc.	Proofpoint, Inc.
Electric Sheep Fencing LLC.	OpenVPN Technologies, Inc.	MarkLogic Corporation	Apple Inc.
Eucalyptus Systems, Inc.	mod_spdy	Chef Software, Inc.	Hitachi, Ltd.
NVIDIA Corporation	Cisco Systems, Inc.	SonicWALL L.L.C.	WinSCP
Nginx Inc.	Blue Coat Systems, Inc.	Dell	Sybase, Inc.
Pivotal Software, Inc.	Bitcoin Project	BlackBerry	Sébastien Jodogne
Tenable Network Security	Google, Inc.	CA	Python Software Foundation
The FreeBSD Project	The Tor Project, Inc.	Intel Corporation	Kaspersky Lab ZAO
The NetBSD Foundation, Inc.	Joyent, Inc.	Siemens AG	Invensys Inc.
Tor-ramdisk	opensource.dyc.edu	RUCKUS WIRELESS, INC.	Sophos Ltd.
		Innominate Security	

Credits: Jake & Kymberlee : Stranger Danger! What Is The Risk From 3rd Party Libraries? : Blackhat USA 2015

MORE

Library	Vuln Count	Vulns Per Year	Releases Per Year	Average CVSS
OpenSSL	90	10-11	3	5.49
the FreeType Project	50	6	2	7.43
libpng	28	3	2-3	6.65
Apache Tomcat	100	12	5	4.72
*2009-to present	522	80	11	8.96
Java *2010-to present	539	98	4	7.07

Credits: Jake & Kymberlee : Stranger Danger! What Is The Risk From 3rd Party Libraries? : Blackhat USA 2015


VULNERABILITIES
VULNERABILITIES EVERYWHERE
rator.net

REMEMBER

We rely on 3rd party to

1. patch
2. maintain security
3. accept security issues
4. in short "**NOT SCREWUP**"

WHAT ARE THE CONCERNS

1. **Open Source Software**
    1. Developer has scratched his itch and will not want to work on it
    2. Developer doesn't understand security implications and ignore reports
    3. Developer is genuinely not in a position to work on project
2. **Closed Source Software**
    1. Company shifted focus
    2. Not enough money

WHAT IF THEY DO ALL THE FIXES IN TIME

PATCH PROCESS

1. Someone disclosed a vulnerability
2. 3rd party vendor fixes code
3. A public advisory is released informing about the update and hopefully security issue
4. Developer has to update the dependencies in actual project (believe me when i say its not easy task) (backword compatibility, regression, feature support etc)
5. Sysadmin / user has to update the software to receive the update

LOOKS COMPLEX

ANDROID OTA PROCESS

1. Google released PDK to Vendor for evaluation
2. Google Announces new version
3. Google send source code to Chipset manufacturer and Vendor
4. Chipset manufactures provides drivers and BSP or stops support
5. Vendor evaluates requirement for device if no driver then no update
6. Vendor updates its own softwares (SENSE, TouchWiz etc)

Cont.

ANDROID OTA PROCESS...

1. Vendor works with carrier for modification
2. Final build is submitted for Lab Entry and testing
3. If bug found patch and resubmit.
4. Take approvals from
    1. Regulatory
    2. Industry
    3. Google
5. Prepare OTA for the Device
6. User Downloads OTA and updates the device

BIGGEST QUESTION WHAT WE CAN DO

3 KEY PLAYERS

1. Component Code Developer
2. Programmer reusing component
3. Enduser/sysadmin using the final program

THEN THERE IS PENTESTER

LETS EVALUATE ONE BY ONE

SYSADMIN / ENDUSER

- Monitor your software feeds to ensure you do not miss security updates
- never ignore update from shared library
- Keep an eye on how shared resources are holding up

DEVELOPERS

(SOFTWARE AND 3RD PARTY)

1. Identify and catalogue your components
2. Never ignore pull requests and security issue bug report
3. Proactively test software and at-least if a fix is released publicly accept security issue

ANY AVAILABLE TOOLS

VULNERABLE COMPONENT IDENTIFICATION

IDENTIFICATION

https://retirejs.github.io/retire.js/

View on GitHub

Retire.js

What you require you must also retire

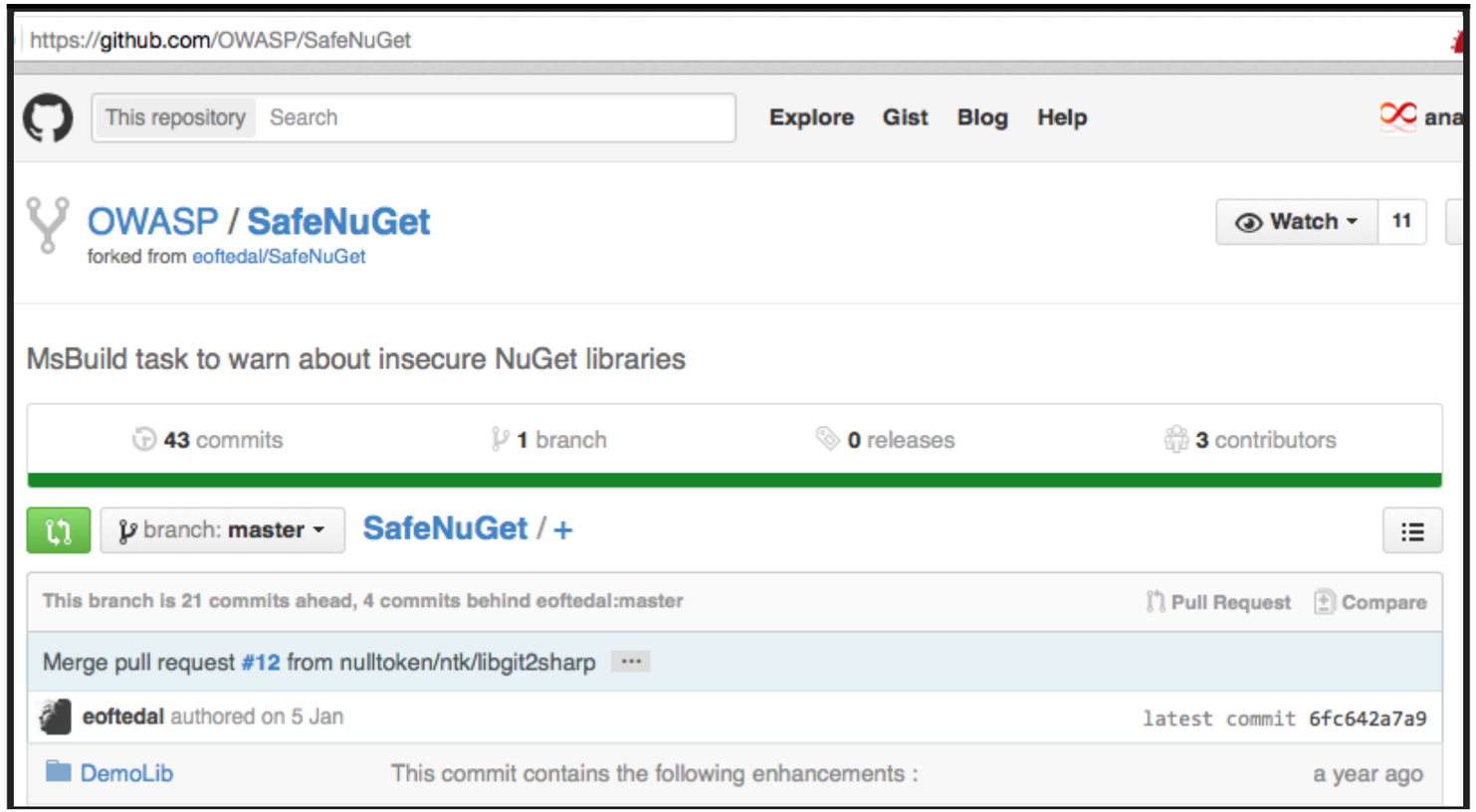
tar.gz .zip

There is a plethora of JavaScript libraries for use on the web and in node.js apps out there. This greatly simplifies, but we need to stay update on security fixes. "Using Components with Known Vulnerabilities" is now a part of the OWASP Top 10 and insecure can libraries can pose a huge risk for your webapp. The goal of Retire.js is to help you detect use of version with known vulnerabilities.

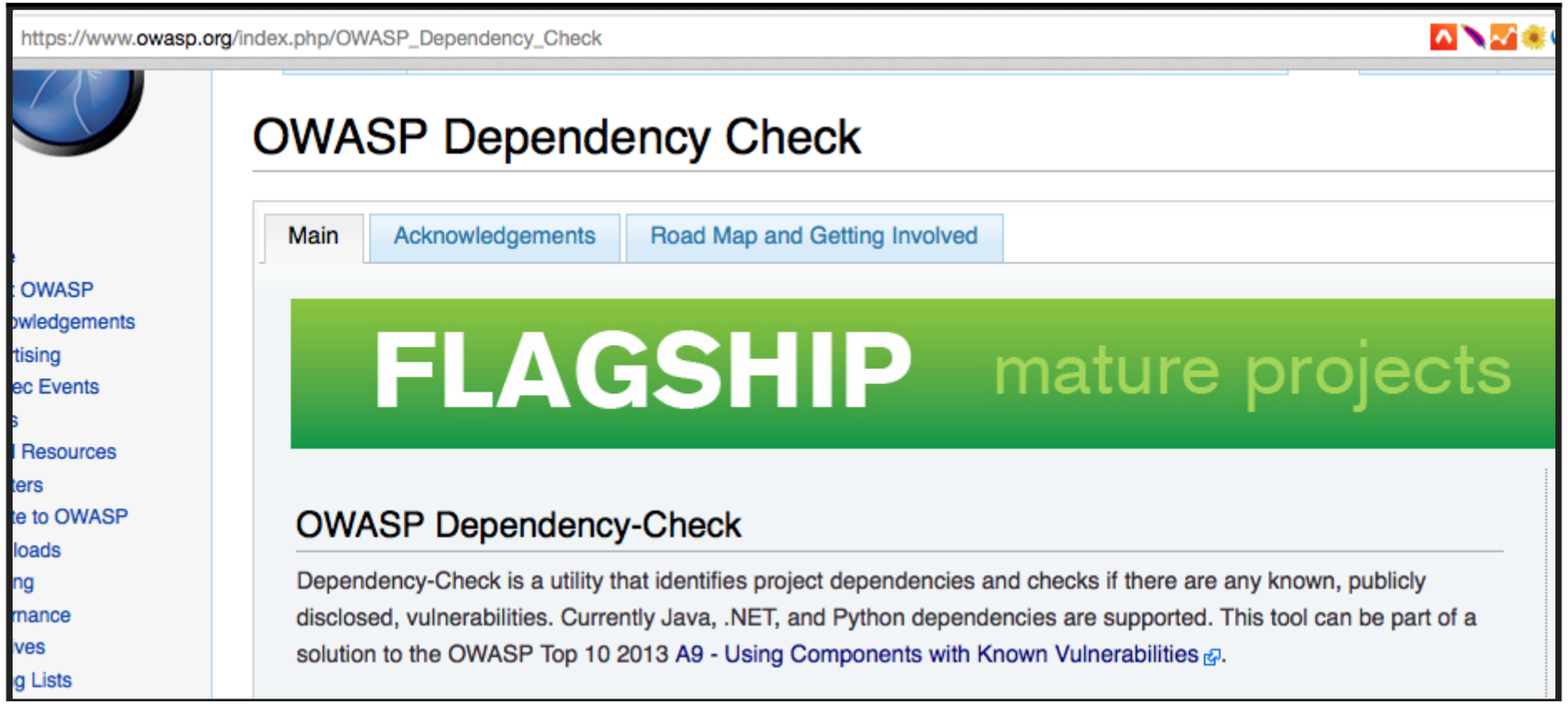
Retire.js has these parts:

1. A command line scanner
2. A grunt plugin
3. A Chrome plugin
4. A Firefox plugin
5. Burp and OWASP Zap plugin

IDENTIFICATION



IDENTIFICATION



IS THIS ENOUGH

1. Not yet
2. We still lack method to track it for every third party library
3. Manual tracking is still required

COMPONENT CODE DEVELOPER

1. Be clear about support status
2. If out of support, release and updated version clearly stating support status
3. Clearly accept the security issues and inform about fix

www.jobberbase.com

/// A bit of history

Jobberbase was born in October 2007.

Filip had previously created http://www.jobber.ro, a tech-only job board that quickly became known and loved in the Romanian tech community.

As an experiment and being influenced by how Ruby on Rails (framework) came out of Basecamp (product), Filip decided to open-source jobber, 3 months after launch.

Jobberbase was the first open-source job board platform, a breath of fresh air in a world where all other similar software was legacy, ugly and paid-for, thus challenging the status quo.

Over the next few years, Jobberbase gained popularity and a few core contributors, most notably Lavi & Cosmin Mendrea and Radu Lucaciu who were also heavily involved in the community, helping people out. Other developers started to make a living by customising and extending Jobberbase for their clients. New visual themes were created...

Good times!

Unfortunately for Jobberbase, life happened and the project slowly fell into oblivion. It was nice, it had potential, but we just didn't have time...

/// Status quo (December 2014)

Having received unexpected help from @rimas-kudelis, we've since launched version 2.0 with significant changes and many additions.

www.jobberbase.com

/// A bit of history

Jobberbase was born in October 2007.

Filip had previously created http://www.jobber.ro, a tech-only job board that quickly became known and loved in the Romanian tech community.

As an experiment and being influenced by how Ruby on Rails (framework) came out of Basecamp (product), Filip decided to open-source jobber, 3 months after launch.

Jobberbase was the first open-source job board platform, a breath of fresh air in a world where all other similar software was legacy, ugly and paid-for, thus challenging the status quo.

Over the next few years, Jobberbase gained popularity and a few core contributors, most notably Lavi & Cosmin Mendrea and Radu Lucaciu who were also heavily involved in the community, helping people out. Other developers started to make a living by customising and extending Jobberbase for their clients. New visual themes were created...

Good times!

Unfortunately for Jobberbase, life happened and the project slowly fell into oblivion. It was nice, it had potential, but we just didn't have time...

/// Status quo (December 2014)

Having received unexpected help from @rimas-kudelis, we've since launched version 2.0 with significant changes and many additions.

PENTESTER

1. Follow steps for Admin to identify all components
2. Cross reference with known disclosures (use dependency trackers)
3. Profit

REFERENCES

1. BlackHat 2015 : Stranger Danger! What Is The Risk From 3rd Party Libraries? : **DO CHECK VTEM**
2. The Unfortunate Reality of Insecure Libraries: Jeff Williams,Arshan Dabirsiaghi : March 2012
3. https://www.gov.uk/service-manual/making-software/dependency-management.html
4. http://swreflections.blogspot.in/2013/10/dont-let-somebody-elses-technical-debt.html

REFERENCES

1. https://prezi.com/g-01vdbth1co/sonatype-survey-2013/
2. http://blog.softfluent.com/2011/08/19/leveraging-third-party-components-or-reducing-dependencies/
3. https://img.en25.com/Web/SonatypeInc/%7Bb2fa5ed8-938d-4bce-8a9c-d08ebeba826d%7D_Executive_Brief_-_Study-_Understanding_Security_Risks_in_OSS_Components-1.pdf

ANY QUESTIONS

ANANT SHRIVASTAVA

- Information Security Consultant
- Admin - Dev - Security
- null + OWASP + G4H
- http://anantshri.info and @anantshri
- Trainer / Speaker : Blackhat USA, NullCon, g0s, c0c0n, Clubhack, RootConf

THANK YOU