Join the Discussion
Connect

Dradis: Effective Information Sharing for Pentest Teams

By Russ McRee – ISSA member, Puget Sound (Seattle), USA Chapter

Prerequisites

For Linux installations: Ruby interpreter, SQLite3 libraries, SQLite3 Ruby gem

Who amongst you braves the toils and tribulations of penetration and vulnerability testing? Should this be your true calling, do you undertake yon efforts alone? Methinks not. Youthinks enough of the olde English, too, I'll bet. Seriously though, most penetration/vulnerability testing efforts are team driven. And those teams need to ensure precise, thorough documentation and tracking, yes?

Dradis will serve you in this cause as a self-contained web application that provides a centralized repository for information acquired during testing in order to work completed and pending. It is the Dradis developer's contention (and I agree) that failure to share "information available in an effective way will result in exploitation opportunities lost and the overlapping of efforts." Testing teams face multiple challenges specific to information sharing, including a variety of output types from all the tools utilized. Testers gather results in different ways. Each team generates different reports, and so on.

Failure to centralize information sharing will result in individual sets of notes per tester used to track their findings, and those notes are often stored locally, or on a shared resource, but not updating in real time for use by the rest of the team.[1]

Figure 1 provides a basic Dradis architectural overview.

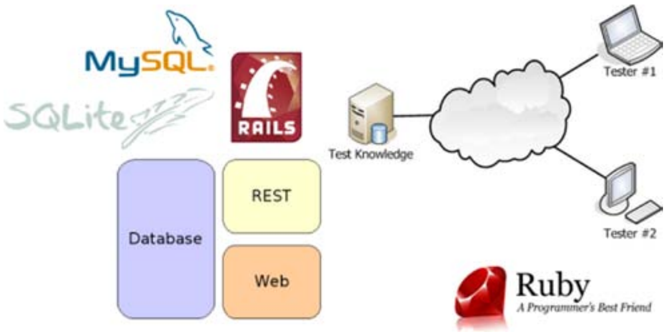


Figure 1 – Dradis architecture

Developed under GPLv2 using Ruby on Rails and platform independent, Dradis uses a simple client/server model where the client communicates with the server via REST web services over SSL. You can choose either a console or browser GUI. We'll focus on the browser client for the sake of convenience and graphic representation.

I asked Daniel Martín Gómez for insight on Dradis, including the name:

"It all started watching *Battlestar Galactica*, the 2004 remake. Without windows in battlestars, they need to rely on their instruments to know what is going on. DRADIS is an onboard information system where each person in the command center has access to a DRADIS screen where they all share the same picture of what is going on. When nothing interesting is going on, everybody is busy minding their own business, ensuring that the ship keeps ticking. However, as soon as something noteworthy appears in the DRADIS screen, they all engage and instantly know what is going on. I loved the concept, the idea that a team of security testers could share the same picture; everybody adding information to the repository, everybody sharing the insights obtained by the rest of the team.

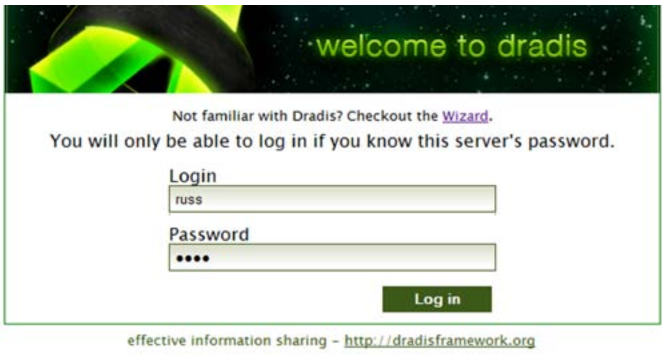
Going from sci-fi to reality hasn't been as easy as we'd like, our goals are:

1. Effective information sharing
2. Ease of use and adoption: we are proposing a new way of working; it better be easy for security teams to give it a try
3. Flexibility: Every user has different needs; we need to build a platform that users (and companies) can adapt to their needs; Everybody can easily extend Dradis using plugins
4. Small and portable: You should be able to use it while onsite (no outside connectivity); It should be OS independent (no two testers use the same OS)

It is evident that the goals we set for the project are quite high, and hitting them hasn't been an easy feat, but with every new release, and with a growing user base, we are getting closer. Today Dradis is being used by pentest companies around the world. We are getting feedback from people who felt that there was not a good tool for security professionals to collaborate effectively. Those same people are trying to convince their companies to embrace our framework. Our community is growing fast and we are trying to keep up with the challenge; more users means more feedback, more churning, more releases, and better results. Our hope is to make Dradis Framework the tool of choice for security professionals and enthusiasts to structure and share information effectively."

In 2009 Dradis was included in BackTrack 4, featured in Offensive Security's Metasploit Unleashed, presented at DEFCON 17, and lined up for inclusion in Pentoo.

1 http://dradisframework.org/overview.html.



Not familiar with Dradis? Checkout the Wizard.
You will only be able to log in if you know this server's password.
Login: russ
Password: ••••
effective information sharing - http://dradisframework.org

Figure 2 – Dradis login

Installing and configuring Dradis

Dradis can be installed on both Windows and Linux. For Windows users, all the dependencies are installed for you; installation is an extremely simple click-through process.

On Linux installations, you can utilize verify.sh[2] which is, as you can imagine, a Dradis dependencies verification script. Execute `sh verify.sh` and you'll be advised what, if anything, you're missing. I lacked the Ruby development libraries; I ran `sudo apt-get install ruby-dev libopenssl-ruby` and was quickly in business.

For this article I focused only on a Windows installation. Once Dradis is installed, getting down to work is as simple as Start → Dradis → Start Dradis Server, then browsing to https://localhost:3004/. There's a nice wizard to get you up to speed at https://localhost:3004/wizard.

Dradis uses an authentication mechanism I struggle to accept, but I understand the developer's intentions. It's based on the knowledge of a shared password. Given that Dradis is designed for small dynamic teams, the authentication scheme seeks to avoid the hassle of creating new users and assign passwords; the team agrees on a password to be shared during the engagement and changed only if need be.

Dradis offers a number of useful plugins for use to import, export, or upload.

In order to make use of the OSVDB Import Plugin you must place your OSVDB API key in `C:\<your Windows installation hierarchy>\dradis-2.5\server\config\osvdb_import.yml`.

I'll cover exports when I discuss reporting later.

Upload plugins include:

- NessusUpload: vulnerability scanner
- NmapUpload: network mapper
- NiktoUpload: web server scanner
- BurpUpload: web application scanner

Each of these allows you to upload results from the related tools; you simply need to be able to generate said results to do so.

The Nessus plugin will upload results exported from Nessus in the .nessus format; the Burp, nmap, and nikto plugins import XML results.

I have recommended to the development team that similar plugins be added for commercial pentest tools such as Core and Canvas.

Using Dradis

I'll walk through a real vulnerability testing scenario and use Dradis as I go.

In February I analyzed a shopping cart application (DFD Cart) that resulted in responsible disclosure, repair, and advisories. DFD Cart is a fairly new project written by an attentive and diligent developer who was very responsive to the bug report. To test DFD Cart for the bug hunt I installed the latest version on my test server (192.168.248.102). After allowing the requested amount of time necessary for the developer to make repairs, Secunia issued SA 38635[3] and I issued HIO-2010-0207.[4] Keep this in mind as I import OSVDB details on these advisories.

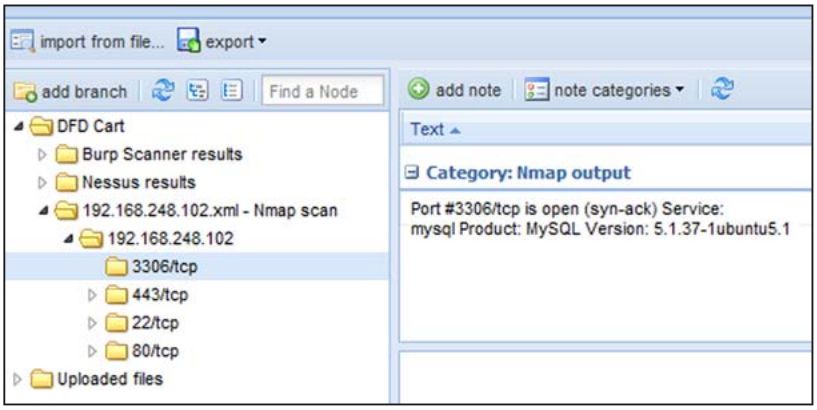
I'll offer some oversimplified generalizations here as this is an article about Dradis, not penetration testing methodology.

Many a pentest likely begins with nmap scans; I prefer a slow comprehensive scan if I'm using Zenmap, which for this test translates to `nmap -sS -sU -T4 -A -v -PE -PP -PS21,22,23,25,80,113,31339 -PA80,113,443,10042 -PO --script all 192.168.248.102` at the command line. Results were then saved as `192.168.248.102.xml`.

In the Dradis UI I first clicked *add branch*, and added a node called DFD Cart.

Note: after submitting content to Dradis, I recommend making a habit of hitting F5 to refresh the UI.

I then clicked *import from file* and selected Nmap upload under *Available Formats* while selecting `192.168.248.102.xml`. I then dragged the resulting node under the DFD Cart branch. Figure 3 describe how Dradis populated the UI with Nmap results.



Category: Nmap output

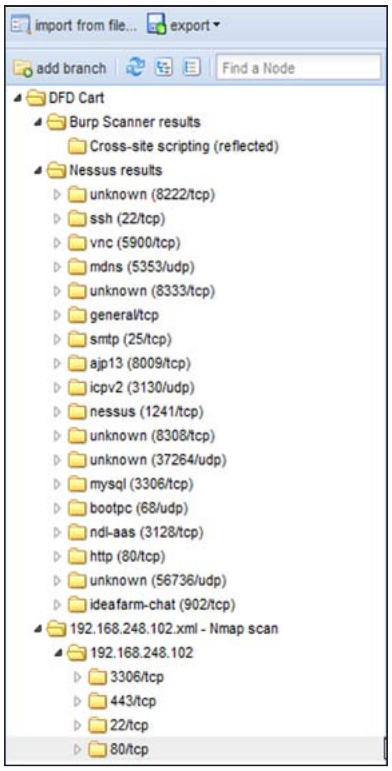
Port #3306/tcp is open (syn-ack) Service: mysql Product: MySQL Version: 5.1.37-1ubuntu5.1

Figure 3 – Dradis imports Nmap results

2 http://dradisframework.org/install.html.

3 http://secunia.com/advisories/38635.

4 https://holisticinfosec.org/content/view/135/45.



```
import from file...  export ▾
add branch  Find a Node
◢ DFD Cart
  ◢ Burp Scanner results
      Cross-site scripting (reflected)
  ◢ Nessus results
    ▷ unknown (8222/tcp)
    ▷ ssh (22/tcp)
    ▷ vnc (5900/tcp)
    ▷ mdns (5353/udp)
    ▷ unknown (8333/tcp)
    ▷ general/tcp
    ▷ smtp (25/tcp)
    ▷ ajp13 (8009/tcp)
    ▷ icpv2 (3130/udp)
    ▷ nessus (1241/tcp)
    ▷ unknown (8308/tcp)
    ▷ unknown (37264/udp)
    ▷ mysql (3306/tcp)
    ▷ bootpc (68/udp)
    ▷ ndl-aas (3128/tcp)
    ▷ http (80/tcp)
    ▷ unknown (56736/udp)
    ▷ ideafarm-chat (902/tcp)
  ◢ 192.168.248.102.xml - Nmap scan
    ◢ 192.168.248.102
      ▷ 3306/tcp
      ▷ 443/tcp
      ▷ 22/tcp
      ▷ 80/tcp
```

Figure 4 – Dradis branch/node list

Nmap scans are great for host-level analysis, but DFD Cart is a web application, so I made use of Burp Suite Professional and saved the test results to `dfd_cart.xml`. Again, I followed the import procedures described above but opted for the Burp upload format. The same held true for Nessus results. Once uploaded, one need only drill in to the applicable node in the UI's left pane (see Figure 4). For each imported finding, the relevant uploaded content will populate in the *Notes* pane.

With the UI focused on the DFD Cart node I also opted to *Import Note*. This includes the above mentioned OSVDB import feature. I selected *General Search* under *Filter* and provided DFD Cart under *Search for*. The seven available OSVDB IDs were returned via the query; I right-clicked them and chose *Import this*. In this case, as I'd already reported the XSS and CSRF bugs at the top of the list (see Figure 5), the import was simply to validate the OSVDB import feature functionality for this discussion.

Testers can also assign their own note categories and apply notes to any node as they see fit.

Remember, as each tester adds content, it's returned to the UI in real time; just remember to hit F5 to keep current.

Notes are attributed to the relevant author with a *Last Updated* timestamp.

Reporting

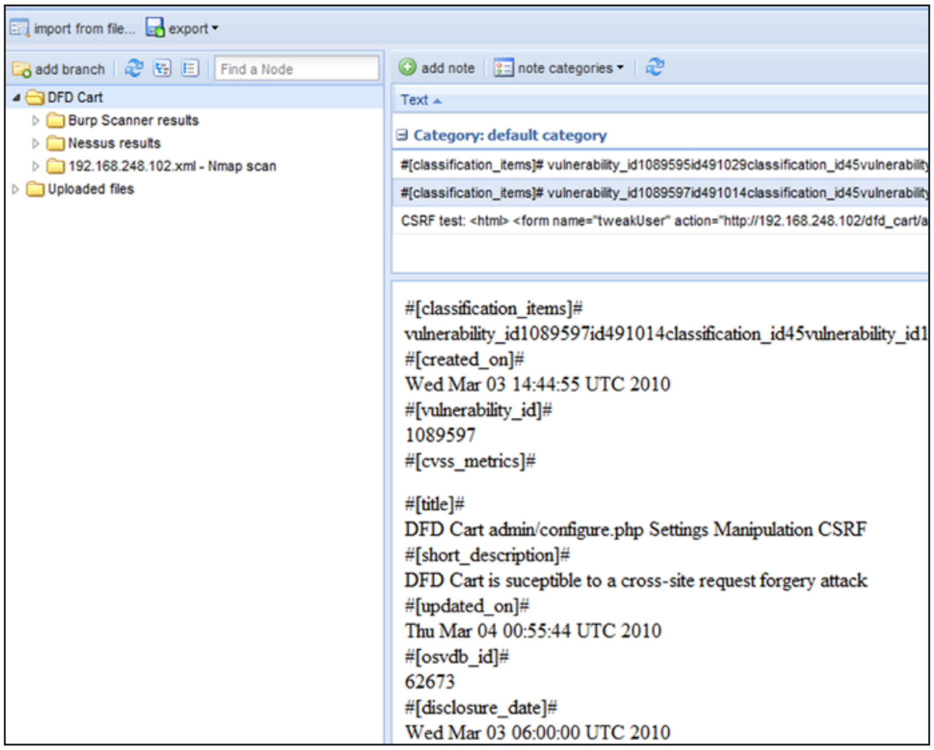
No pentest engagement is of much value without a resulting report, and Dradis includes export functionality to assist in this cause as well.

In order to generate reports all branches/nodes that you wish to report must be assigned to the applicable category. In the Notes UI, double click the category associated (default is *Default category*) with each finding/note and choose *HTMLExport ready*. Results are quite utilitarian by default but can be customized via template modifications. The same reporting can be generated using custom Word templates as well.

In conclusion

For teams that facilitate many penetration/vulnerability tests that require uniform documentation and organization, Dradis is quite useful. Consider this a young project; it's under dynamic development and is a bit buggy, but getting better all the time. I've incorporated Dradis into my testing process as I was pleased with the benefits discovered while writing this. Give it a good close look and provide feedback to the development team; they are attentive and very interested in continuously improving Dradis.

Cheers…until next month.



```
import from file...  export ▾
add branch  Find a Node            add note  note categories ▾
◢ DFD Cart                         Text ▴
  ▷ Burp Scanner results           ⊟ Category: default category
  ▷ Nessus results                 #[classification_items]# vulnerability_id1089595id491029classification_id45vulnerabilit
  ▷ 192.168.248.102.xml - Nmap scan  #[classification_items]# vulnerability_id1089597id491014classification_id45vulnerabilit
▷ Uploaded files                   CSRF test: <html> <form name="tweakUser" action="http://192.168.248.102/dfd_cart/a

#[classification_items]#
vulnerability_id1089597id491014classification_id45vulnerability_id1
#[created_on]#
Wed Mar 03 14:44:55 UTC 2010
#[vulnerability_id]#
1089597
#[cvss_metrics]#

#[title]#
DFD Cart admin/configure.php Settings Manipulation CSRF
#[short_description]#
DFD Cart is suceptible to a cross-site request forgery attack
#[updated_on]#
Thu Mar 04 00:55:44 UTC 2010
#[osvdb_id]#
62673
#[disclosure_date]#
Wed Mar 03 06:00:00 UTC 2010
```

Figure 5 – Dradis imports OSVDB advisory notes

Acknowledgments

— Daniel Martín Gómez, Dradis developer

About the Author

Russ McRee, GCIH, GCFA, GPEN, CISSP, is team leader and senior security analyst for Microsoft's Online Services Security Incident Management team. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at russ@holisticinfosec.org.