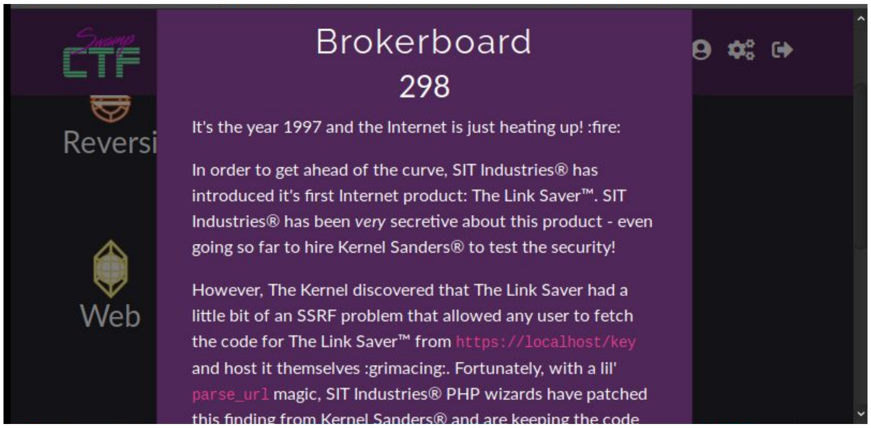
This is my write up for Brokerboard a web challenge at SwampCtf 2019. Kudos to UF and the team that hosted this competition. It was an excellent CTF and I got to learn a lot within the 48 hour time span of the competition.

Problem:



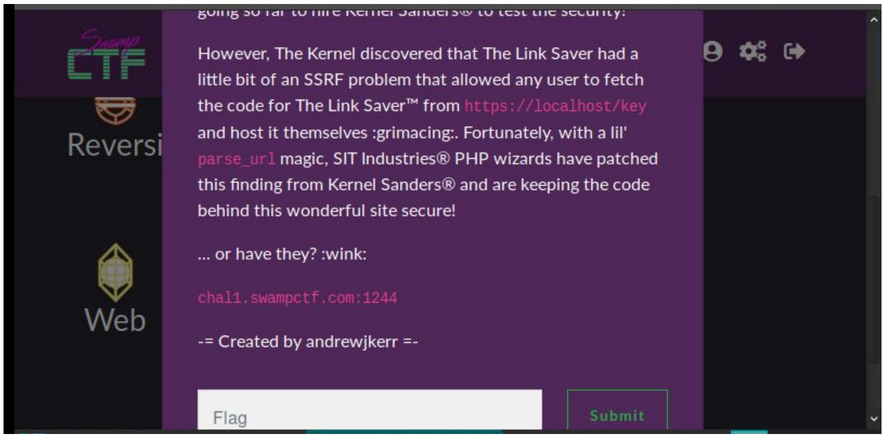
Brokerboard

298

It's the year 1997 and the Internet is just heating up! :fire:

In order to get ahead of the curve, SIT Industries® has introduced it's first Internet product: The Link Saver™. SIT Industries® has been *very* secretive about this product - even going so far to hire Kernel Sanders® to test the security!

However, The Kernel discovered that The Link Saver had a little bit of an SSRF problem that allowed any user to fetch the code for The Link Saver™ from `https://localhost/key` and host it themselves :grimacing:. Fortunately, with a lil' `parse_url` magic, SIT Industries® PHP wizards have patched this finding from Kernel Sanders® and are keeping the code



going so far to hire Kernel Sanders® to test the security!

However, The Kernel discovered that The Link Saver had a little bit of an SSRF problem that allowed any user to fetch the code for The Link Saver™ from `https://localhost/key` and host it themselves :grimacing:. Fortunately, with a lil' `parse_url` magic, SIT Industries® PHP wizards have patched this finding from Kernel Sanders® and are keeping the code behind this wonderful site secure!

... or have they? :wink:

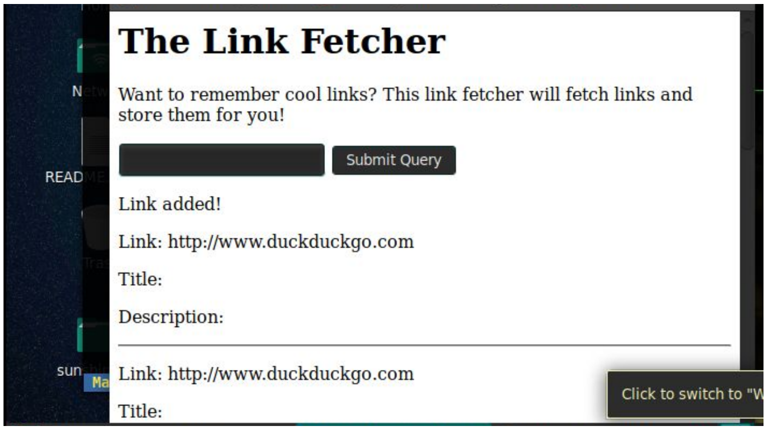
`chal1.swampctf.com:1244`

-= Created by andrewjkerr =-

Flag

Submit

So first we are greeted with all the hints required to solve this problem.   We know that we have a SSRF vulnerability somewhere in the application that will allow us to access this location : https://localhost/key. We know that the application is built with PHP and that it is using parse_url() as some sort of blacklisting mechanism.



The Link Fetcher

Want to remember cool links? This link fetcher will fetch links and store them for you!

Link added!

Link: http://www.duckduckgo.com

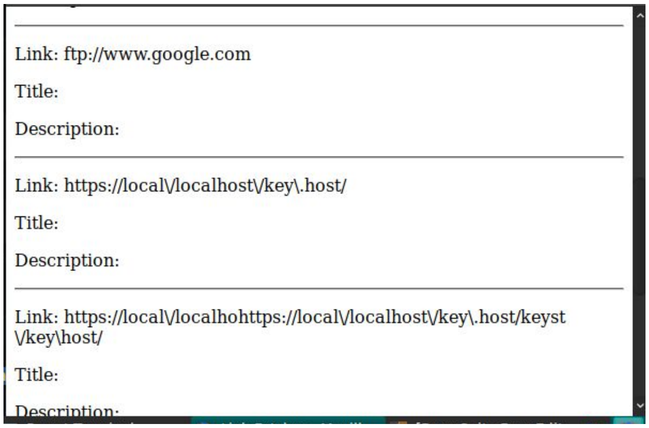
Title:

Description:

Link: http://www.duckduckgo.com

Title:

Navigating to the challenges site we find a form that takes a URL and breaks up the URL with the parse_url() function in order to look for the string localhost/key. If localhost/key is found it is blacklisted, else it passes and allows you to add the link to the page.



Link: ftp://www.google.com

Title:

Description:

Link: https://local\/localhost\/key\.host/

Title:

Description:

Link: https://local\/localhohttps://local\/localhost\/key\.host/keyst
\/key\host/

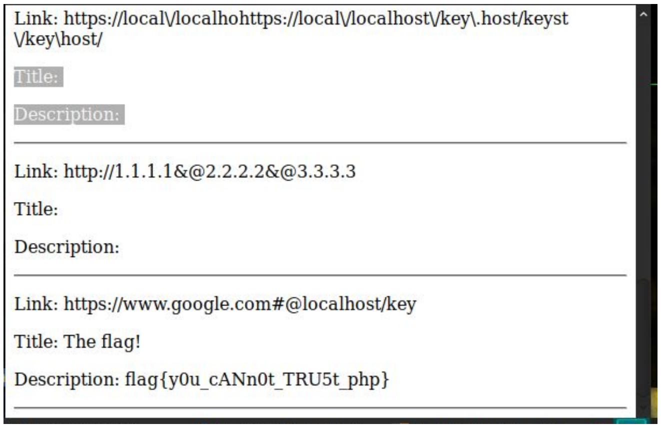
Title:

Description:

After attempting some known character escapes within the authority section of the URL, testing different protocols, and sending different encoding schemes… I decided to break out my google fu skills and learn something new.

This led me to a very talented security researcher from Taiwan named Orange Tsai.[1][2] His talk and slide deck go into parsing issues with php and curl but for the sake of the competition all we needed is the php section. The bug occurs in how parse_url() recognizes the authority for the URL. So by inputting a known valid authority (i.e. https://www.google.com/) we can trick the server to loading localhost/key with the payload shown below.

Solution:



Link: https://local\/localhohttps://local\/localhost\/key\.host/keyst
\/key\host/

Title:

Description:

Link: http://1.1.1.1&@2.2.2.2&@3.3.3.3

Title:

Description:

Link: https://www.google.com#@localhost/key

Title: The flag!

Description: flag{y0u_cANn0t_TRU5t_php}

Sweet we successfully loaded the flag onto the page as shown and then scored the points.

[1] https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf

[2] https://youtu.be/2MslLrPinm0