Kings of Dorking

Cracking has been a huge field over the previous years, fluctuating in size as legal concerns becoming greater as more information is divulged by the elites. Building an infamous name for manual injections and completing complex attacks. Unfortunately, the world of cracking is not that intimate. Now days all you need is a basic supplier of Dorks, some proxies and some premade cracked tools from 2014, if you think I'm kidding, half of you buying are probably using something like that.

The reason I call this unfortunate is now skids and leechers have invaded the lands. And they're learning quick, moving their way up from the traditional pastebin scrapers to steal other people's dumps and now even actually learning what's required. Databases are becoming harder to obtain. Especially private and High Quality ones. I'm here to improve the quality of your Dorks, teaching you from scratch the logic of Dorking and it's uses in the field of cracking. Failed security attempt 1. This was white

!!! DISCLOSER !!!

! ! SEVERAL GOOGLE TABS MAY BE REQUIRED ! !

(and a dictionary, a lot of you are illiterate)

Contents

Chapter 1. Dorking…

Dorking is the art of understanding and utilizing a search engine to emit the desired results. If I wanted to find a file on anonfile; I can go on Google and use this search query, inurl:anonfile.com + Target File

I can find a Tweet with the exact same syntax, and I can repeat this for almost any target that is at public discretion and isn't banned from Google.

What is a Dork?

A Dork is a search query that a search engine can read and interpret to provide the most precise URLs that correlate to that query.

Why do we use Dorks?

Cracking is, at its simplest form, finding basic, unprotected sites, compromising its security measures or lack of; exporting information of which is desired and then use them for other purposes.

This is important because this next part will be complementary to our injection criteria.

There are 3 PRIMARY forms of SQL Injection (SQLi)

Put

Post

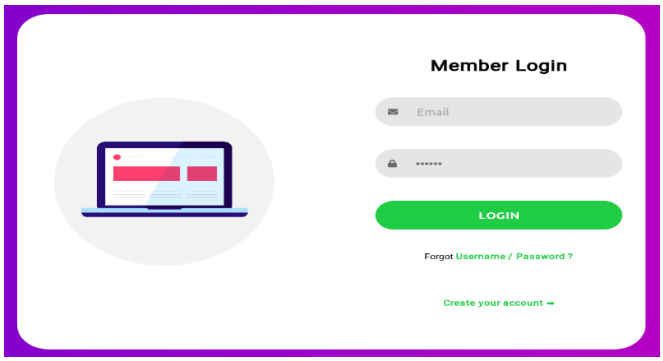
Get

Get being the simplest to search for in a Search Engine such as Google or Bing. Our usually primary Search Engines.

Get is when data is put from a "form" into the URL as a form of temporary data storage.

FORMS

http://desarrolloweb.dlsi.ua.es/cursos/2012/web-programming-with-php/example-of-getting-data-from-a-web-form



Please look at this article I found.

The concept to be taken is that data is input from some variable on a site, being user completed or automatic in the functionality of the site. Data will be sent to the server (what the website is being hosted from) and complete a task, storage or site functionality to your session or others.

The reason this information is important is SQLi is the implementation of malicious code to manipulate the backend storage server to send a request to the Database and "Exploit" data from said database or complete other malicious attacks (Database = $$$). This manipulation can stereotypically only occur with an abstract Parameter (A Parameter is the session information of data input that can be used to locally store where the user is on the site).

With this essential FORM of information, this will allow us to understand the basis of how our Dorks will achieve a desired goal.

The reason we use Get injection and do not primarily target the other Parameter functions is because, you can't search for the other ones. It's just not time efficient to be manually going through sites, 1 by 1 to eventually find a vulnerability. To all SQLi studiers to understand the joke in that, respects to you. I hope it's consolidation for the amount of your time I just wasted.

The reason that introduction to parameters was required, is now you understand WTF a Page Parameter is, and now understanding the process of Dorking onward will be like learning how to fix a speech impediment.

Keywords

Words key to finding results of a certain type.

Every single time you go on Google and search your loli porn. You are using a keyword.. -Or two.

The logic behind our keywords is we want to use the same words people are using to search for the same thing we want to get combos for.

Now as we all have experienced, putting one word into google emits millions of results. For Dorking, this is not targeted enough nor efficient. So, we usually use a target keywords (Amazon, Netflix, Hulu, Steam) and an extension keyword (events, new shows, new games, price)

Our extension keywords or secondary keywords will be explained further down the track. Right now, all you need to know is, try make things detailed, or else results that are undesired will appear, corrupting the efficiency of cracking.

Page Extensions

Page Extensions can occur in URL's after the naming of a File Page (as seen below).

We have our misconceived page types, which are actually just inbuilt with the domain, which come with the Domain on purchase by the site holder, or else the site would go nowhere.

Examples:

Amazon.com

Netflix.com

Idk.org

Wikipedia.org

And so on so forth. (gov, com, org, shop, online, xyz, waytomuchshittobehere)

Files Page Types and Page Extensions

File Pages are the

Amazon.com/pagenameinuse

And for some sites, they also include page types, (**php, asp, aspx, cfm, etc**) Which are known as the page extensions.

Amazon.com/pagenameinuse.php

These are what we call our Page Extensions, completely ignoring the domain types because it's completely irrelevant to our Dorks.

Why is **THIS** IMPORTANT???

When Dorking, I've experienced almost all sites with page parameters in the URL also have page type. Becoming a useful asset to restrict unlikely injectable page param extensions to more likely to succeed ones.

We primarily use:

php

asp

This is because these are most commonly used languages to connect to the database through a page, like shown in that form article above.

Chapter 2. Syntax (Google)

The most important thing to keep universal in the creation of ALL DORKS, is the concept of syntax. If I spoke English with a Chinese sentence structure, it wouldn't make any sense. It's the exact same to Google or Bing. If you don't speak their language, they won't understand you. **ALWAYS CHECK WHAT SYNTAX YOUR DESIRED SEARCH ENGINE OPERATES ON**

Syntax is the way that something is interpreted. Google reads phrases and independent terms depending on its "smart read" features. Which we can't disable.

What this means:

If we were to search

Gaming shop.com

Google will search for a site with shop.com in the domain and Gaming being somewhat of importance to that site. This is an issue because, very few sites will contain [shop.com]

The way we can improve on this syntax error is by using a space:

Gaming shop .com

Hopefully, this highlighting has distinguished how google will now interpret this search.

Quotation Syntax

Most search engines allow the user to input "" (Quotation marks) to help indicate this word/phrase must be written on the site. This is considered a Search Operator but is not usually disclosed on services like Google and Bing as it serves as common knowledge to the informed. Use it to your advantage. It can be used to String words together to create phrases.

Remember search engines have their own criteria on syntax, which leads without saying that it's most ideal to test out all possible syntax uses to help assist with the usage of these search operators when creating our Dorks.

Ordering & Capitalization

Google's recent Search Algorithm Enhancement Update entailed updates towards their search query targeting and response algorithm. Those poses both an opportunity and a threat. Presenting the avid concern of rewriting all my Dork Types multiple times and Disabling some Exploits on Google that were previously able to be utilized; but also allowing for new techniques to improve results.

This can be described under the usage of Capitalization and Ordering.

I tried

Revision 3 and this had their name ***Kings of Dorking*** Please Don't Leak

Ordering

Order matters. Easiest and most conceptualised explanation of this change is that. When utilizing the targeting of page extensions, parameters or directories, consider the **order of the URL** that can be the results of this target.

What does this mean? Well a site doesn't put .com after the parameter, simply because it's the domain and that's not possible unless it's a redirection of the site. Which **hint** is exactly what happens if you do this. Same can be said for page extensions and directories.

RULES WHEN PUTTING STUFFS TOGETHER (which I don't suggest to be clear):

Page Extension CANNOT go AFTER Parameter
Page Extension CANNOT go BEFORE Domain Extension

Parameter CANNOT go BEFORE Page Extension

Parameter CANNOT go BEFORE Directories

Parameter CANNOT go BEFORE Domain Extension

Domain Extension **GOES BEFORE ALL** Page Extension, Directories and Parameters!

General Syntax Rules:

Operators do not go directly before Search Functions

e.g.

+ intext:

All Search Functions **EXCEPT** inurl: operate with Operators

e.g.

intext:shop + online

Don't link/string/connect Text Targets with URL Targets

e.g.

game website **+** ?id=

Quotes **CANNOT** work on inurl:
inurl: will **NOT** allow spaces.

IF you add a regex operator to inurl, a wildcard will not work.

IF you add a wildcard to inurl, a regex operator will not work.

I tried

Extended Search Operators Syntax

When Dorking, we will sometimes face issues with syntax. An easy way to avoid phrases linking we don't want them to and linking phrases we want to, we do this by using Search Operators and Search Functions.

Below is a list of Search Functions and Operators for Google. This will become vital in creating your OWN unique Dork types and Dorking techniques. Making the quality of your Databases you find increase drastically. Please enjoy this Archive of relevant information.

Search Operator List:

Operator	Function	Example
.	Allows a following character to be connected to the primary input.	Usage: .php Result: .php/whateverthesubdomain OR Usage: 12.3 Result: 12.31 11.3 OR Usage: .php? Result: .php?whateverisnext
+	Connect Queries	Usage: Gaming + Shop
&	AND	Gaming & Shop
~	Synonym	E.g: ~happy Results: Happy, joyful, ecstatic. Same theory with keywords
""	String together input.	Usage: **"hello world"** Confused?

		The search engine reads it as separate requests when not in quotes linked together: "hello" "world" We want the search engine to find them together **"hello world"**
()	String data together	Usage: (Gaming Health)
|	OR	Usage: gaming|health
[x-x] Or [xxx]	Data range	Usage: [0-11] [A-Z] [12345] How Google See's It" [0,1,2,3,4,5,6,7,8,9,10,11] [a,b,c,d,e,f,g,h,etc] [1,2,3,4,5] It is treated as a string array; this means each piece of data is able to be search separately. Unfortunately, this usage is rather broken in a lot of ways.

Search Functions:

Search Function	Function	Example
cache:	Cached version of site on google	cache:site.com
inanchor:	Find pages on sites with that name	Usage: inanchor:how to basic Result: site | How Basic Site | how to basic site | how to
allinanchor:	Makes the entire following phrase be in a site page name.	Usage: inanchor:how to basic Result: "how to basic" as the anchor for the page
intext:	Finds text on page	Usage: intext:Good morning ladies and gentlemen Result: will get sites with most those words in a phrase
allintext:	Finds all text on page	Usage: intext:Good morning ladies and gentlemen Result: exact copy of text
intitle:	Explanatory	intitle:Gay Rights Matter
inurl:	Explanatory	
allinurl:	Same logic as above. Just adding the demand for all characters given.	
ext:	Extension to file, does not need a .	ext:php
filetype:	File Type	
site:	Searches for the site provided	E.g: it, au, us, gov, etc We can use it for specific sites if we plan domain takeovers or just regular site searching site:amazon.com

Chapter 3. Let's get Started

You now understand the major uses of logic in Dorking. Let's start showing you how to use it.

Basic Dorks

Keyword PageType Page Parameter

TargetKeyword(we will use xWord from now on) [XW]
Keyword [KW]

PageType [PT]

PageParameter [PP]

Example:

KW .PT?PP=
Shopping .php?cartID=

Now you may be asking.
"Where the fuck did the: . ? and = come from?"

The . is used on all devices to identify a file type. Which when we access via the web we call page type.

The ? is the most commonly used divider used by web servers to identify that a parameter is directly after. Others include: **?, &, amp**;

And the = identified the data attached to the parameter.

I hope that has cleared any confusion in a lesson to basic Dorks. Another way we can construct basic Dorks is with several keywords

XW KW .PE?PP=

Amazon shopping .php?cartID=

OR

KW KW2 .PE?PP=

KW2 being the second keyword

Makeup Shopping .php?cartID=

This can help in finding non site-specific combos and instead attempting to find generalised combos for this field of topic.

This concludes the Basic Dorks section, the introduction to a world of Dorking.

Questions that are expected:

How do I find Keywords?

How do I find Page Parameters?

These questions will be answered after we complete the introduction to all the standpoints on Dorking for beginners and what is most commonly provided by Dork providers. The reason we are making you wait is we want you to understand how and why Dorks work before we throw other legends of theory at you including how to obtain keywords and page parameters. Please just take notes of what you can learn without thinking about how you will get your variables.

Complex Dorks

Complex Dorks are the next level of Dorking. We can use Search Functions to improve the accuracy of our search. I will only be providing the Search Functions (SF) from Bing and Google.

Complex Dorking is something very useful when creating Dorks. It's the usage of Search Functions to optimize the results of a search to meet a specific criterion.

Complex Dorks

Keyword (KW)
Page Type (PT)
Page Parameter (PP)

And new to the list

Search Functions (SF)

Usage of Search Functions can be debated. But I don't care how you want to debate it.

inurl:?PP= or .PT or Directory

intext:KW

ext:PT

filetype:PT

site:.DE

For any questions about the usage of **site:**

I tried

We are able to add our domain extensions that came with the domain into the Dork, enabling us to choose a country code (us, it, kr, au, br, nl) OR our regular sites domain type (com, org, net). This is what we consider to be our targeted country Search Function. Which can help. Other aspects that help is changing the location of the Parser to where you'd like

to extract URLs for as Google likes to optimize it's results with the location of the user. *Helpful Right?*

The amazing thing about google is it has such a large amount of capabilities; it is genuinely hard to keep up. Thankfully I've comprised the most detailed guide to using Google optimally right here.

(KW) .(PT)?(PP)=

Transforming this Basic Dork into a Complex Dork is as using a Search Function

For Example:

intext:"mario bros" .php?id=

Now is a basic example of how we can use Search Functions.

Donjuji. A cracker to initiate the UHQ mainstream of Combo Cloud's. Released a very Basic Guide to Dorks several years ago. "https://pastebin.com/raw/39wTesCS"

In this guide he outlines the usage of Search Functions and demonstrates the usage of Search Functions to target his page types and parameters. Very interestingly he also exposed a common accessibility vulnerability of lot of databases have. That being the ".sql" file exposed on the search engine as an anchor page. A rather large mistake by the developers. This out of the box usage of Search Functions as well as knowledge of file types and exposure vulnerabilities is exactly what hacking stands for.

The definition for hacking it trying to get around something, find a fix. Or in our case, find a fix to our Dorks, finding these vulnerabilities like how Donjuji experimented. I **HIGHLY SUGGEST** trying to find your own methods as there are plenty undiscovered. One day you may become the next Donjuji.

Search Function Dorks:

intext:"mario bros" .php?id=

mario bros inurl:.php?id=

mario bros .php?id= site:.com

mario bros site:.com ext:php ?id=

mario bros .php?id= site:it

I tried

mario bros site:it ext:php ?id=

mario bros ?id= ext:php

mario bros ext:php ?id=

Hopefully this helps with your understanding of Search Functions. It's a pretty basic tool which, once utilized becomes an amazing tool in the field of Dorking.

The Big Questions

Now as I expect most of you to be asking. "How do I know how to make a Dork"

The way of the Dork is orientated around Dork Types. Dork Types are just ways to order and compile your Dorks. Nothing surprising to the name but it does require some creativity, some experimenting and some time.

"How to use a Dork and what is a Parser?"

Dorks are used on a Parser. This is a tool that completes the queries on the desired search engine and then scrapes the results. It's rather simple and tools suggested can vary.

How to make Dork Types

Dork Types are just the format that you can generate Dorks from.

I will be using my own formatting for declaring each part of a Dork. This is because I'm not happy with how the community labels them, so I'm making my own. This will hopefully become a branding of sorts towards my Dorks but I'll accommodate however you want to interact with these words and what you want to use as an abbreviation.

Keyword (Target) = xKW

Keywords 2 = KW2

Page Extension = PE

Parameter = xP

Search Function = SF

Directories = Dir

Keyword 3 = KW3

Making Dork Types requires you to FOLLOW the rules of Dorking. For Google this is inclusive of the rules in Google Syntax of Chapter 2.

With this logic we can use our Operators, Search Functions and knowledge of Dorking to create Dork Types.

Some examples will be
KW + *KW2 \ ext:PE inurl:?xP=

"KW" / *KW2 \ ext:PE \ inurl:?xP=

KW2 + *KW ext:PE inurl:?xP=

These basics of Dorking will not change. So worth with the logic I've provided to trial Dork Types and create a list of successful Dork Types to use when generating. This will allow you the advantage of your own custom Dork Types to yourself instead of sharing with your peers by me sending mine.

Keywords and Parameters

Our hunt for ideal Keywords and Parameters has left people on multiple sides of the scales.

But here's the secret; you cannot target something that doesn't exist in order to obtain results, this makes our life too easy. I say this because we can use the same search engine to find sites operating with SEO (search engine optimization) and use their keywords and other sites parameters from my initial keyword list results. To complete this simple process, begin by creating a list of random keywords for your target.

e.g. Netflix

initial keyword list:

Netflix shows Actors Reviews

Netflix Original trending Shows

Netflix discount Dates Codes

Netflix mistakes and bloopers

Netflix new shows Release Date

And complete this till you have a solid list of keywords. Then parse it on your search engine of choice and runs the URL results on a tool like METAKOSTRIV, which will extract the metadata from the site including keywords, parameters and other valuable information for later on in this E-Book.

More Advanced Keywords

Keywords 1 and 2 are different. Primary when we discuss them, we have our Target Keywords (Keywords 1) and our Unique Identifier Keywords (Keywords 2). **THIS DOES NOT MEAN THERE IS ONLY 2 KEYWORDS!** Methods to improve the results can include choosing Brands, Actors or other Key Features about your Target to keep as a **Primary Keyword**. Leaving other Keywords to be your **Unique Identifier Keywords**. It's rather simple and can help to use.

It's important to diversify your Initial Keywords to get unique results, try adding actors, brands, games and other unique identifiers to expand your results and give you tailored keywords and parameters. I also suggest prioritizing more than just your target product or brand, try finding similar products as well to expand the range of targeting, some of your Primary Keywords can also be just related to your category rather than the target itself.

Parameters

Once you complete this task, simply remove language parameters, long keywords and duplicated. Another skill you will learn with experience is what parameters are worthless and never show up as a vulnerable parameter. In my experience I've derived that language orientated parameters and extremely long ones that are EXTREMELY specific never obtain results or sometimes URL's at all. Then feel free to begin sending your keywords and parameters into battle and start testing your target.

Parameters Detailed

Parameters have 2 types, Dynamic and Static. Static parameters cannot be changed, Dynamic can change. You may wonder what this is referencing to. Static means that it cannot change until the webpage is reset, and a Dynamic can be altered on the go without resetting the webpage. This is significant because static parameters cannot go to anything that the server doesn't give it, so our basic **Get** parameters will not be vulnerable to such targeting methods. This means that we need to target dynamic parameters to increase our vulnerability figures. How does this work?

Avoid language parameters,

Avoid country parameters,

Avoid "li" parameters or other parameters which will only gain data from a set list on the server. Some may argue that page is a static parameter, well, not always. So, these aren't set in stone, but from experience, it's suggested to avoid regularly static parameters.

Another way to look at this is to consider what will be stored in the Database and what will be unique for the page. A list of ID's will be stored in a Database because it's a list and should be accessible globally amongst the site, but a list directly for one page would only be for that individual page. It's a logical theory and can help clear confusion about Parameters.

Testing Parameters

Testing Parameters is rather simple, you cannot test if they're dynamic or static usually, but you can check if the parameter is popular enough that you can add keywords and other variables to the

search query and still obtain results. This is important to save bandwidth, reduce invalid Dorks ratios and make life easier. The easiest way to test this is to grab 2 simple Keywords correlating with your target and add 'inurl:PP='. If this result gets several hundred thousand URL's or more then the Parameter is rather simple to Dork with, if it's less be cautious about the Dork Types you use. This method is useful for people without experience in Dorking and it's also useful if you just haven't experimented with Parameters before.

Parameter Methods

Some Parameter Methods can be the usage of targeting Parameters of Plugins and Drivers. A perfect example of this is the "ODBC" Driver used on SQL Servers. (https://dev.mysql.com/doc/connector-odbc/en/connector-odbc-configuration-connection-parameters.html) Plugins and Drivers that operate Directly with the Database will increase your success rates by reducing the exposure to Parameters not involved with the Database. Good sources for these are things like the main Database Provider (DBMS) site:

https://www.mysql.com/products/connector/
https://dev.mysql.com/downloads/

ETC

This method is simple and logical, and I incline everyone to utilize it by finding your own targeted resources.

This is always fun finding new methods. This one is extremely useful and I'm happy I figured it out. If you find any that satisfy you that you'd feel comfortable sharing with me. I'd appreciate it.

Another method is altering the Initial Keywords you created and add 'ext:php' to them. This is mainly because a lot of sites with exposed page extensions usually have parameters to store data, this can give you plenty of parameters to play with and expand your targeting range. It's a useful tool and should be utilized.

Advanced Dorking

Advanced Dorking…

Advanced Dorking is including Search Operators into your Dorks. This is pretty straight forward. The way we do this is simply by using our big brains to link phrases we want to connect. If we want certain keywords to combine, we will use Search Operators to do that. If we want to prioritize a certain keyword, we can do that. This is about making your Dorks as optimized as they can get.

Example:

Dog Bed

Google will receive this as two separate words, separated or one not even being in the results. The big issue is we usually don't put in our phrases to be separated by Google.

The way we could complete this is to use one of our Search Operators

& **+** **“”** **()**

E.g.

I tried

Revision 3 and this had their name ***Kings of Dorking*** Please Don't Leak

Dog & Bed Dog + Bed "Dog Bed" (Dog Bed)

Highlighted are the usage examples of these Search Operators. All completing the same / similar task. By using Search Operators. Getting sites following your request of stringed queries can become easier by simply adding these to your Dork types. Use them efficiently and you will expect nothing less than results alongside your Search Functions when required.

To be clear. I don't always use Search Functions or Search Operators. I always like to consider my target demographic and what I'm really going to be expecting.

What do I mean?

Well, if I'm using a target with an extremely small demographic, it makes no sense for me to restrict my search query with a Search Function or requiring keywords to be directly linked on the site.

BUT

When targeting something without a target keyword, (Amazon, League of Legends, Roblox) then it's a good idea to include multiple strings. This is SOULY because of the sheer number of URLs you get when you don't have a designated target for your Dork. With an excessive number of URLs, you're less likely to find Databases (DB) that are vulnerable to SQLi.

The usage of Search Operators helps us in targeting our DB of choice.

Before: Amazon Iphone .php ?cartID=

After: "Amazon" + "Iphone" .php ?cartID=

OR

After: Amazon + Iphone .php ?cartID=

Or any experimental variation of Search Operators. The syntax has been provided. For you to become fluent in Dorks. I suggest testing out ways to input keywords into google with Search Operators and keeping track of the results as you change how you use Operators. This will help you in understanding why things work and when they do such.

Chapter 4, Syntax.

Now in the previous chapter on syntax we made it aware that Google speaks its own language. Being able to understand how google will receive separate queries and how to optimize that system while Dorking.

Now we are going to bring light to a method of which people perceive as "HQ". Which is simply just understanding how google will interpret its regex.

Regex is a very complex topic, for a basic rundown, Regex is shorter terms/characters of which register to an AI and translate to create a formula, this formula could be to identify certain characters or skip characters; or how we will use it.

We will be using the Google Regex System (**GRS**) to optimize our Dorks. As I mentioned earlier in this book, Google has its smart query reading AI. Well, we can actually do something similar to turning it off. We can inform it on how to read a Query.

You may be asking, "But it already understands me doesn't it?". And the truth is, I have no clue. I'm not a developer. But what I can be certain of is by using these Regex "Formulas" you will be increasing the accuracy of your search once again.

Google Regex System

Now this may be confusing. And trust me it should be. I've reused data from earlier with some minor adjustments to make it completely accurate. The reason I've done this is we need to start somewhere. And it's certainly not going to be at the complete top, so you lose context of how things work. In my experience I've been able to utilize the Google Regex in almost every Dork I currently produce, with few exceptions of certain methods of targeting sites.

The Google Regex System is comprising of characters with plenty of use that can change, alter, optimize, randomize and other unique features for Dorks. It's an important tool to preserve the quality of Dorks that obtain unique results, and it should be used with an understanding of purpose. It's vital to understand the usage of these in order to get the best results possible out of them.

Regex Wildcards

Operator	Function	Example
.	Matches any single character (letter, number or symbol)	Usage: 1. Result: 10, 1A
?	Matches proceeding character 0 or 1 times	Usage: hi? Result: h his hi
+	Matches the preceding character 1 or more times	Usage: 10+ Results: 10, 100
*	Matches the preceding character 0 or more times	Usage: 1* Results: 1, 10
|	Creates an OR match Do not use at the end of an expression	E.g: 1|10 Results: 1, 10

Regex Groups

Operator	Function	Example
( )	Matches the enclosed characters in exact order anywhere in a string Also used to group other expressions	(10) matches 10, 101, 1011 ~~([0-9]|[a-z]) matches~~ ← GONE. any number or lower-case letter
[ ]	Matches the enclosed characters in any order anywhere in a string	[10] matches 012, 123, 202, 120, 210
-	Creates a range of characters within brackets to match anywhere in a string. Or avoids targeting.	[0-9] matches any number 0 through 9

Regex Escape

Operator	Function	Example
\	Indicates that the adjacent character should be interpreted literally rather than as a continuation of strings. Another example and usage that's different to how other perceive. It's like the following characters after the \ are a new "string" of data in the query, all that follows is a new target. Otherwise, it's all one singular target string.	\. indicates that the adjacent dot should be interpreted as a separate section of the query to anything prior. Gaming \ Shop \ inurl:id=

This stuff can even go more complex. But fortunately, that will not be necessary. I copied all these definitions directly from google. Using these in according ways can help.

GRS

The way we would use GRS is by already including Search Operators or Search Functions.

E.g.

Shopping + Cart .php?cartID=

Now, we have our Search Operator +

There are many ways we can edit this Dork to include GRS.

Shopping + Cart .php ?cartID=[0-10]

We're now targeting a method we will explain further in the book called Numeric Dorks. Fix, just add a random number anywhere in the Dork except the parameter or in search functions to reduce the chance of invalidating the Dork. Enjoy!

All that you need to know about that method is that, almost every single tutorial you have ever seen before was wrong. And I can't explain it shortly here.

E.g. 2

Shopping + Cart \ purchase .php ?cartID=

~~In this example I have exited the query stringing from the + operator. This usage has made, Shopping Cart become a target and purchase being a separate string. I've also added a space between .php and the parameter. This is to avoid over intense targeting. I've also used a backslash before the =, a perfect example of taking the next character literally, this is important because **Google has their own Regex** for their own parameters for each user, including their images and search data that is referenced the same as any other site using Get. By using the backslash we've told Google to use this next character literally instead of a Google Parameter.~~

Patched and I'm mad. Separating each part of the query is completed with a backslash with a space before it. Meaning you can do anything after that first space backslash ' **\ inurl:**'. This is extremely useful and I use it in almost every Dork.

E.g 2a

("Shopping" + cart) \ ext:php \ inurl:?cartID=

Revision: You can now separate the importance of each section of the query with backslashes to expand the search query. This was a recent development from the Google Update. This allows for unique results and is extremely useful.

E.g. 2a (patched…)

("Shopping" + cart) ext:php inurl:?cartID\=

Revision: You can no longer target specific characters because Google patched it. Meanieeeee!

E.g. 3

Base Dork

Shoe + Shop .php ?cartID=

Regex Dork

Shoe? + Shop .php ?cartID=

This Dork will be used if you're using a keyword that you **KNOW** for a **FACT** can be commonly replaced with that word with an "ed" or "s" on the end. *(or any other extension)* This will allow you to find those results as well. Expanding your target range.

Advanced Explanation and Usage for GRS

Now a big part of this book is teaching you how these Dorking solutions **actually achieve** results. Now it's a reasonable time to explain most of these Regex Options. Now our best place to start is the order I've listed them all in.

Wildcards

Wildcards are our way to expand our search results, avoiding our failed Dorks while generating or mistargeting. Human Error is a large concern for Dorking. Whether it's been from the Dorker (us) or other users of the internet (**BOOMERS**). So, it's vital to optimize our chances of success. In this following section of the book I will be explaining each wildcard and usage.

.

Our period is a useful tool when targeting a numeric target or a string of characters towards a subject. Whether your target be for a game, event, year. Anything like this can be optimized to assist to randomize that result to be anything.

e.g.

Zelda Event 201.

This usage of the period will now target results with 201[SOMETHING]

Results can give you a year, a number of an item or anything in this usage technique.

?

Usage of a Question Mark can assist in the reverse searching for context and stringed keywords or something else.

e.g.

?games events London

In this context, characters before games could be linked to the word, could be strung together in a phrase or simply be in the URL: top-ps4-games

e.g.

games? events London

In this context, the "s" on the end of game will be non-required when scanning through sites, I'm unsure of the extent this is capable in Google.

+

The potential uses of this operator are flexible. Being able to complete multiple tasks, either adding a dynamic random character to the end of a specific word or by linking multiple words to be together or as a priority in the search query itself.

e.g.

How to multithread in C# with 10+ at the same time

This usage of the operator will now search for queries of 10 with another character after it, being a number or letter. Either way can assist in searching legal documents or just attempting to expand the search boundaries.

e.g.

Game + popular events

This use of the operator will result in stringing both queries together, Google has a hidden way of completing this as it's a rather complex algorithm they've developed but it's a very useful and commonly used practice by Dorkers.

*

Usage that follows this operator is similar to the previous but instead of colliding the queries but can also randomize the results, with an increase to the beginning or the end of a numeric query or just randomize the strings in the query.

e.g.

How to multithread in C# with 10* at the same time

Result is simply just randomizing that next character, also and most commonly used is to put it before and to randomize the figures after that number / query as well.

e.g.

*Game popular events

This will simply just make all characters following that word be randomized. But with Google sometimes it can do it for the entire query, so please be sure to use some of the operators I will be explaining further on to solve syntax mistakes.

|

This operator means, OR, which means it will choose one or the other. This is the same syntax usage of "+".

e.g.

Game | popular events

Meaning, "Game OR popular events"

This is not an overly used operator. I won't spend time explaining this.

Regex Groups

Regex groups are how we can organise our search queries within a query to achieve searching for multiple pieces of information in a singular query. We can also use Wildcards to assist the accuracy of these searches, increase the diversity and improve the results. Now this is what really makes the advanced Dorkers good at what they do.

()

Parenthesis are a useful tool to increase priority in searching or link queries together. I personally use these most commonly in searching. It's a vital component when linking multiple queries together and making them a priority along side your Search Functions and other Regex Usages.

e.g.

(Gaming Shops Local) ext:php

This usage is linking the entire phrase / query into a priority queue, instead of Google searching for each independent word, instead it will now attempt to find these words strung together, or close to it.

Be aware that once you incorporate Search Operators / Wildcards you will be drastically optimizing your Dorks to complete multiple tasks at the same time. You will be making Google process multiple requests, also improving the accuracy of your search.

[]

Data arrays, for anyone who's been a developer or participated in the numeration techniques of coding. Will know that arrays are pieces of data of which can be called, and then find a certain entry to that array and use that data, as a static piece of information. We can use this to randomize our searches with numeric orders or to simply incorporate random characters into the search. Be aware Google's array inputs each unique character as a separate entry to the array. No commas to be required and adding them may even corrupt the search intent.

-

This operator is used inside of the Data array in the previous explanation. This is just to define the distance between two characters. Like when counting, instead of putting 1, 2, 3, etc into the array. Simply putting the starting number followed by this operator and then the finishing number to have it automatically targeted. Can also be used to avoid targets. Such as github.com. But it isn't much of a concern for us and will waste our space for a query.

Breaking Regex & Syntax

In the event that you will use an operator, function, or regex. You will need to separate it from the remainder of the search or ones prior. The best way to do this and the only way to do this is to use a backslash. \. This is pretty straight forward. ~~Either use it connected to something you want to target or~~ use it with spaces to separate different parts (strings) of the search. This singular operator is how to make your Dorks become some of the highest quality you could imagine. It makes targeting so much more precise and controllable and it is my number one **MOST IMPORTANT** method to improving Dorks.

That's all you need to know to be at the top of your Dorking game. No exaggeration, no bullshit, no time wasting. If you understand everything above. **YOU CAN OFFICIALLY DORK.** And not only can you Dork. You can understand why things will work.

After recent revision it's also capable by a space on search operators and functions, but it's still recommended to use backlash in the event of a complex situation.

Chapter 5, Extensive Google Syntax.

Extensive Google Syntax, EGS. EGS is about the detailing in Dorking that can be used on Google and why. I will be covering how to string queries together, how using operators with search functions is not possible because it will make your eyes water and make your nose bleed. Let's get right into the thrill of GOOGLE.

So, to make this extremely simple and easy to understand.

Google optimizes its results in relation to what's searched, how often that or a similar search is made, and what is clicked from those searches.

This is important because we can utilize GRS and EGS to create an expansive result count and therefore improve final product results.

Basics and Rules

When Dorking on Google, everything is simple, easy and all syntax. Consider how Google interprets your query. It treats everything without a Search Function as a keyword with minor priority to site name, title name and other considerably important information, with this information you can formulate queries around this factor.

If you want to target a page extension, use
ext:

If you want to target a directory, page, parameter or other information in the URL of the target, use

inurl:

If you want to target a domain attribute (your .com, .net, .us, forum.), use,

site:

These are your URL targets, our primary targets only in the URL of the results that we want to target. By targeting these properly your results are **100% guaranteed** to be relative to those targeting features due to the Search Function. This will reduce results, purely because it's a more refined query, quality over quantity. Other Search Functions have their own purpose, these are primarily for current syntax and targeting.

Testing Dorks

Another important skill to Advanced Dorking is being able to tell what's a good, and what's a bad Dork. The ideal Dork for SQLi Get is having the targeted parameter highlighted in the URL of the sites shown to you by Google and the relevance to your target.

Easiest and most logical way legitimately put it into Google manually and check a few pages. Ensure that "Big Brands" aren't the only results, like Facebook, Amazon, or other major sites that can be never vulnerable. This is one form of results poisoning or soft banning on Google and is a huge flaw. Another thing to watch out for is URL's that look either; completely random or are IP's. This is because they're usually honeypots and pose a **threat** to getting any results. Another valid point to take avid consideration of is if you get a static 8 URL's on that Dork, even if you change some of the variables. This is restriction of results by Google to stop what we do, Bing does it as well, but Google is much easier to notice. Another method of this is when the first page states there is lots of results,

then once you go to the next pages it says there is no more or that they're repeated and to show the omitted results (it stops u several pages after). All these tests are important to avoid creating invalid or result poisoning Dorks.

Regex Dorking

Regex Dorking is built around the Regex provided by Google. Understanding how to use our Regex Wildcards and other Regex Operators are extremely useful. Regex Dorking is containing of multiple Search Exploits and other private methods of Dorking which are **NEW** and **NEVER USED BEFORE**. But these will not come yet. The core concept of Regex Dorking is understanding how Search Functions operate.

inurl:
has no spaces, quotes and equal signs do not get read.

ext:php

is the **ONLY WAY** to use ext:

YOU CANNOT PUT SEARCH OPERATORS DIRECTLY BEFORE OR AFTER SEARCH FUNCTIONS.

Targeting Parameters

Unless something is specified implicitly to be targeted elsewhere than the text with a Search Function, what characters you put there are keywords.

inurl:?param=

Be sure to consider if the target will be considered in the DB of your target, what will be stored and will be it be called onto the page from that parameter. It's the most important logical query to consider when choosing parameters.

Page Extensions | When to Target

Page Extensions are useful when target specific web pages that could potentially or are commonly vulnerable. This is usually a partial link towards a common directory Dorking method.

In respects of the common directory Dorking method there is no space between a page name and a page extension, being the only instance there is no space before the page extension.

E.g.

cart.php

but in all other valid instances it's not proper to target in this manner, this is because this is just a directory. Also, this is without considering the **NEED** to use Search Functions to define where this is targeting. This is simply an example of what is being targeted.

Bypassing Google Bot-Detection

Bypassing Google Bot-Detection is as simple as using:
1. Correct Syntax
2. Real Keywords
3. Realistic Parameters

That's as simple as the demand for Google Bot-Detection Bypassing.

Chapter 6, Google Search Settings INURL

Google operates on Queries, Operations and User Settings. It's literally built around your most evident to click, your user data (country, time, location, cache, etc), the query you enter and the operations you use to optimize that search. This knowledge is how some Google Regex System mistakes are made.

Things that can be optimized with the INURL settings of Google.com
Google Location|Geographic Location (gl={countryCode})
Language (hl={langCode})

Start Search (q={Dork URL encoded and other user data})

Search info starter (search)

Filters (filter={0})
Connector of commands/settings (&)
Amount of URL's per page (num={number})

EG.

Google.com/search?hl={langCode}&gl={countryCode}&q={URL Encoded Dork}

Utilizing this can allow you to improve your Dorks without actually using Dorks.

Chapter 7, Numeric Dorks.

Numeric Dorks are simple at first and only glance. I'm disappointed that it's been misconstrued by society for the previous years. As a parameter for searching, as some other pathetic methods have proven to be constructed.

Now I'm here to provide the really basic and simplistic explanation to this method.

Parameters need a piece of data / information to follow it. Post or Get. This is a useful piece of information, because now we just need to know what most sites will use to order their data in Get parameters, and we can randomize targeting patterns. Well, as should be no surprise to the name of the method. Numeric orders are **most common** in get parameters. Not meaning it's exclusive to, but is most consistent between web developers. As it's a simple to code project, simply adding a byte count system to each page in the webserver, making storying data smaller than a byte and easy to manage. With this information, we can use randomized numbers on the end of out Page Parameters and simply reduce randomization of results for each search, but also making them random to which results will show with those keywords in mind. Numeric Dorks are easy. Just generate numbers in lengths of 1 to 3 digits and then add them to the end of each parameters' =. Enjoy this method, it has been public for a very long time as it is common knowledge, but crackers at our level have not been common practices and therefore can still receive and emit good results.

E.g. 1:

Flash ?game_id=93 ext:php

Flash ext:php ?game_id=93

Here, the results are 500 URL's, mostly containing all the information that was searched for in the query. The important thing to recognise here about this Dork, is that the results do not all mirror that numeric figure, this is because Google will broaden results to attempt to assist us in finding what we're searching for. This can work against us in moments where there are too many URL's in the results, and other times can work perfectly as needed and provide us alternatives even if our query is invalid and could not return a response matching all given criterion.

E.g. 2:

Flash ext:php ?game_id=1+

This Numeric Figure can be changed to increase or improve the results. This allows you to use GRS to target sites with an expanded view. This is because Google only shows URL's specific to your search. So, continue adding to that number and continue creating Dorks and finding unique targets. There is a large amount of opportunity in this form of Dorking. This also works in Search Functions!

It's extremely important to avoid using complex Dork types when using this method to avoid the query returning null.

Keep it simple!

Chapter 8, Database Errors | Vulnerability Method.

Database Errors have been around since the beginning of databases. Put simply, there is some reason that the code connecting to the database has caused an error that results in a potential vulnerability to exploit. These errors are a Vulnerability that allow hackers to obtain an injection point for the majority of the time. These exploitation techniques are usually one of the faster responding injection methods due the security of that section usually becoming null and void once an error has occurred. These exploits have been common knowledge since 2012 and is not a suggested targeting strategy for private and reliable results. But for the sake of being completely comprehensive we will be following the two main types of Error SQLi.

Most Known SQLi Error Vulnerability (Stupidity Error)

This is by far the most universally exploited vulnerability on the face of the planet when it comes to SQLi. Simply put; The database has an error that has been rendered and displayed prior to the indexing to the Google Index of Websites. What we call the internet for the most part. We usually use Google to find our sites and their SEO pages, some developers prematurely post updates and releases to the Google index which result in errors being publicly visible and accessible before even loading up the website. This is a critical flaw and has been exploited for almost an entire decade. This method works by multiple methods, being the most successful of them is using the search function, '**intext:**' to target the SQL error that follows this search function. These Dorks are usually accompanied by keywords and other targeting strategies like default and common page methods, to be elaborated on further into the book.

The main method to finding your initial error messages is to go find a site that has an index of them. This is usually found on the vendors site directly. From there you just need to find errors that present on the front end of the site and have some fun, find a huge list and just expand it constantly. This method of Dorking has proven useful to individuals in the community, although not perfect it can present some success. It's suggested to play with this at least once with Dorking, it's a lot of fun making a list and getting started, and it's rewarding in the end as you might find something really cool.

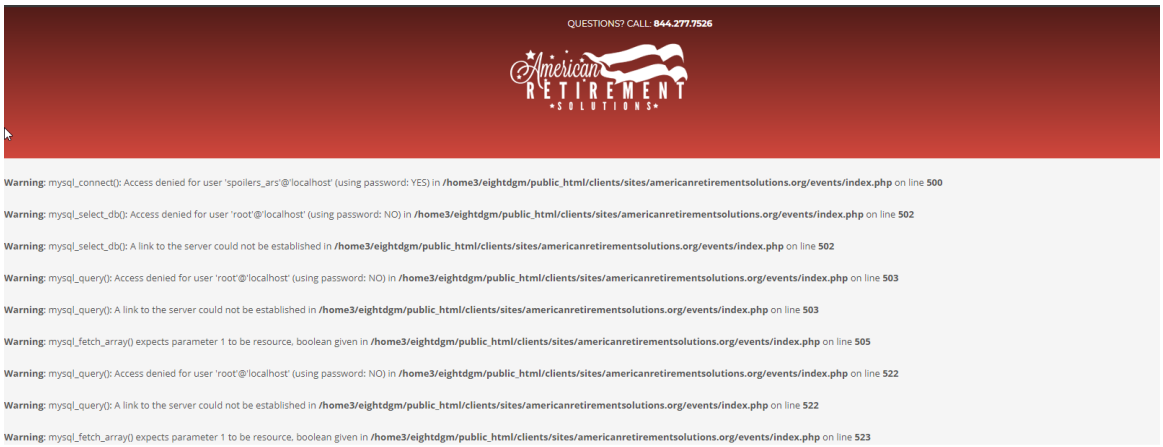
E.g. 1

As seen in this example, the get parameter, '**id=**' is being searched. The error that is being searched for on the page HTML is, "**Warning: mysql_fetch_assoc()**". With this information Google has derived almost twenty-eight thousand URL's from its index.

And with no surprise this Dork has extracted parameters called 'id' in the search and obtained targets with the error message in the text. This is a **Successful Query**.

This Dork's entire goal is to obtain URL's with the parameter and a vulnerability message. The parameter is to gain an entry point and the error message is to find **publicly known** and **vulnerable sites**.

E.g. 2, How it can look.



If this isn't clear, the error messages go where the data from the database **SHOULD HAVE** been.

This form of Google Dorking has been abused since 2010, it is not a private method and I will not spend time compiling a list of hundreds of these error messages. You have a brain; you've proven that by obtaining this book and reading this far. Google a target of yours, the ability to think independent is what makes you good at this field. Not by following a guide word for word.

Less Known Error Dorking

Less known error Dorking is the same error Dorking as above. But with a stance of logic. By going through SQL errors like the ones above, you can find a list of new errors to target, probably less abused and therefore can render some HQ results. This is extremely powerful and is suggested to experiment with.

For example:

In the above images there are lists of errors on the page. Grab one you don't have in your list like:
"Warning: mysql_fetch_array() expects parameter 1 to be resource"
then go Google this query
intext:"Warning: mysql_fetch_array() expects parameter 1 to be resource"

Then go back to step one, find new errors and do it again. Adding to your list.

This is useful form of Dorking and can still reap results.

Another way to expand this success rate is to add parameters to the search. This is because it will increase the injection rates and get unique URL's. Enjoy!

True Error | Unsearched but Practical

True Error Dorking. Truth is, you cannot target it. The best you can do is target sites with parameters, **Post** or **Get** and **attempt** to use syntax mistakes and evil queries to **result** in an error to exploit. If it were targetable like the first one, it *wouldn't be feasible* to target something so publicly accessible, it's like trying to find a dictionary when you have Google in your pocket, it's counterproductive and wastes time. I see the fascination with the first part of this chapter though, and so I've included and elaborated on it.

Chapter 9, Stringed / Extended Dorks.

Stringed Dorks

Stringing Queries has been a target for a very long time as it allows a large diversity of a search. This has recently been discovered in the mixture of using **Search Functions** and the **GRS** in combination.

Rules to be acknowledged prior to trialling this method:

1. You can only use it on Search Functions that allow for Spaces or **without** a Search Function
2. **Experiment** with the **GRS** to optimize unique results

So, how do Stringed Dorks work?

Stringed Dorks is about giving a **string** of Keywords linked together via **GRS** to create a randomized order of search to Google. Leaving it the ability to expand the results and therefore providing a more intense search. This can result in URL's that haven't been vulnerable to other methods of Dorking and also provide an experimental side to using Dorks.

The usual format that's used, but is not exclusive to:

intext:game*Forum?Link
intext:game?Forum*Link

intext:game*Forum*Link

intext:game?Forum?Link

As you can tell, this usage of **Stringing** is using Wilcards. This is because it **allows for diversity** in the results but still meets the demands of your search. From this you could increase or reduce the level of Keywords. Add **Parameters** or **Default Locations** / **Files** or **any myriad** of opportunity. These Dorks are something I've never seen publicly and intrigue me far more than any other I've seen from all of my discoveries of Dorking.

Another Method of Stringing is also utilizing our Operators to link queries together to obtain results that match that query. This is usually completed with parenthesis' and can be utilized in a myriad of ways.

(Game + "Playstation" newest release)

This is querying the keywords together, making it possible to utilize the search around the keywords in an order as well as specific requirements of being together. It's a useful tool and can be used to string certain requests together.

Extended Dorks:

This is simple and works the exact same.

This can be used to expand Parameter Searches or Directory Searches. There are other practical usages which have been proven less effective but it's certainly a topic of dabble. But for the sake of being comprehensive I will cover all angles of this method to truly be complete on the topic.

Method one of Extended Dorks:

Parameter Searching Method:
ext:php inurl:id|game_id|index

As seen above there are 3 different Parameters. Bound together by an OR Operator. This is useful because we can make Google select appropriate results. Another interesting factor is the Page Extension criterion, this just reduces the count of URLs without Parameters, it's not compulsory and it's suggested to play around with this method.

Now the extension for these extended Dorks that correlates with the information provided in a later chapter. This will allow a unique usage that provides a regex exploit for Google, spoilers I know.

ext:php inurl:**?** id|game_id|index =

You will understand the functionality of this Dork once you read the chapter on exploits for Google Searching. Just remember with this syntax decision to use a \ after the inurl: to avoid syntax errors resulting in appalling URLs.

Method two of Extended Dorks:

Anchor Conventions in URL

ext:php inurl:how+to+download+fortnite

This method is simply just putting a string of keywords together with + and using it with **inurl:** and this is utilized to find unique results although has grown popular over previous years as more people have understood its advantages.

It's highly suggested to find further methods of this avenue as they're usually undiscovered or show significant purpose or succession. It's also important to utilize the tools at your hands, you have Google; **TRY EVERYTHING!** Google is very accommodating for search availability so you may as well try everything out with the aspirations of succeeding.

Chapter 10, Dork Types | The Right, The Wrong, And The Bullshit..

Dork types has been a rather polarizing topic brought up in conversations with buyers.

The simple answer is, Dork Types are 100% how you make them, but you need to follow rules. So, in this chapter I plan to expose common bullshit in the community and also provide correct syntax examples and common Dorks and their logic. Leaving you to expand from that field of knowledge to further expand your ability to perform and surpass the regular Dorker to become a ***King of Dorking.***

Chapter 11, Google Search Exploits

Google is the largest search engine used today. Fortunately, it also has Regex Search Exploits which allow for unique usages of Regex Strings to in ways never used before. These are from the newer updates of Google and are extremely useful, there is no explanation for the logic behind these because there isn't any documentation or understanding behind it. Simply just follow along and enjoy.

Anti-Parameter Dorks

Anti-Parameter Dorks are built off a Regex Exploit for ext:.

This is iconic importance behind this is that there is no demand to target parameters anymore. E.g.

ext:php ?

The **?** is a Regex Exploit that seems to improve the rates of ? being in the URL. This is seemingly because ext: doesn't close the string when a regex exploit is after a space. It's extremely weird but I will not complain in the slightest.

Comma Dorks

Comma Dorks improve URL rates, do I know why? Nope, do I care? Nope.

This cannot be done for ext: or filetype:

e.g.

inurl:?id=, game Favourites ext:php

game, Favourites ext:php inurl:?id=

game Favourites ext:php inurl:?id=,

game, Favourites ext:php inurl:?id=,

ext:php game, Favourites, inurl:?id=

These can also work without a space after the ,

inurl: Spaces via Regex Exploit

As soon to be seen in the Stringed Dorks Chapter, there is a Regex Exploit relating the spaces possibility in inurl:.

e.g.

inurl:? id|game|tracking

inurl:? asp|aspx|php|pdf
inurl:? store.php|game.php|/game/|/store/

inurl:? store.php

Notice the space.

Chapter 12, Dorking for Plugins and Drivers

A lot of site developers run the rather cheap scape approach towards their sites. This is especially useful for us because a lot of the frameworks in circulation that are open source are also public and outdated. Providing the unique opportunity to exploit them in multiple fashionable ways.

Examples of this is like the Driver used in our Parameter Method. This is using an older Driver that can and has been proven to be susceptible to SQLi attacks and other vectors. This allows us the availability to find other resources through sites like GitHub, Database Managers, Older Coding Site Suggestion Articles. These resources we have access to allow us unique pivot points for our Dorking that other people do not consider when creating Dorks.

But how do we find these resources? What do we search for and how to we interact with this information?

1. Find a target, whether it be DBMS's (Database Management System Drivers or Site Plugins (like for Wordpress, or other systems). **IMPORTANT:** to get SQLi you need to find plugins or drivers that interact with the DBMS. To find these, it's suggested to use keywords like "Portal", "Plugin", "Cart" or "Login" to find these resources. Another way is to think like a developer and try out what they would have to search to find these systems.

2. Understand **PUBLIC** indicators that this site is operating this. (Parameters, Directories, Trademarks, Banners, Footers), also understand if it's a protected part of the site you're targeting or something accessible by everyone, (this is key for the next step)

3. **DORK!**

It may seem basic, but that's because it is. A good example is one that a student discovered.

inurl:d_blog_module

This is a management system for open cart payment gateways released in 2014. Yet this student was able to get outstanding results. Simply find systems like this through the resources above and utilize them to find unique Plugins and Drivers to find some results probably no one else has. It's an extremely versatile method of Dorking and can be used to expand the results of your Dorks and extend your knowledge as well. It's highly suggested to play around with this method and find something that suits you and to enjoy it.

Chapter 13, Email Access Dorks |Exposure | Psychographic

Email Access Dorks. The bane of Dorking because it's excessively unpredictable and un-targetable, to an extent. It can be found through previously exploited Exposure concerns, like default file dumps amongst other things, but can also be found through descriptions of services asking for emails and the psychological response of users on the internet.

Exposure Dorks are nothing special and are being countered more and more regularly, so therefore I find no need nor point to explaining those Dorks. So rather I will be explaining the Psychographic responses of users on the internet to help you find and determine targets susceptible to exposing their Primary Password with the rest of the world.

Psychographic Understanding

The understanding of the mind is built up of behavioural patterns. The one most important for this chapter being the level of trust that people have for large organisations and how they can stretch that trust to sites like mail outlets and shopping sites.

The primary understanding to keep it extremely simple. People trust big brands, if they see another company using the similar interface (Amazon to another smaller retail shop) they will seem to trust that small shop as well. And those can be highly susceptible to our attack vectors, giving us the unique opportunity to pinpoint this over trust. Same for smaller newspaper articles and sometimes some bigger ones. You can also experiment this in many other places where you find yourself reusing passwords for different fields.

"Why do people use the same password for shopping as email?"
Most people that do this are addicted to shopping or browsing the internet, this leads to late night signups and intense amounts of laziness on their behalf. Most people also have a go to password and well, shopping is a need in every society, unlike games and pornography.

This understanding of how people partition, and clump information together can allow us the unique opportunity to exploit the laziness of these people.

So, getting the goal of email access Dorks is pretty simple, target people, not the result. People are the weakest link in security and leave availability for mistakes to be capitalized on. Think where you may reuse your passwords and target those, what's big in the media, what finds its way from the bottom to the top of the internet and target those categories. Use that knowledge from those categories to expand your email access Dorks to a level where you actually get decently consistent hit rates. It's been seen like in the dump of Zynga, (the shopping company) that people reuse those same passwords from shopping to email. And it leaves that avid opportunity for people to exploit it.

Chapter 14, Default Directory Dorks | Common Directory.

When Dorking you can target more than just a parameter and some keywords. In fact, there is a benefit from sometimes avoiding what Google aims to avoid and just target something different.

For this chapter on Dorking, there will be a coverage of Default Directories, Common Directories and the advantages of targeting them in multiple instances. Directories on a webserver are the location of files that are being called to the site front end (client end, what we see) which indicate where to call the information from and mirror the compiled code. This is an important piece of information because without this we cannot comprehend this topic. A lot of premade sites or other services are too either mangled together or complicated for developers to bother rewriting to work in a different format. This is an advantage to us because once we find something that's popularly vulnerable, we can target it regularly.

A good example of this is if we were to be targeting the default webapp visitor page of php sites. index.php

Chapter 15, Bing vs Google

Google vs Bing. This is honestly such a requested topic over something so simple. Google has far more sites in their directory, more capabilities, SEO and other features. Bing is operating on a much smaller user base, much less sites and is made to avoid Dorking because they're liable to vulnerabilities. So, this cause of events results in Bing being hard and near damn impossible to Dork for because it will just restrict results. Google although, has a myriad of open-source syntax usage examples and libraries, has lots of search functions and operators under regex or otherwise. Making targeting easier for several reasons: it's got a lot more Search Engine Optimization (SEO) which improves accuracy of results, it's got search functions and operators that help in targeting efficiently and it's got more sites indexed than other providers.

Google is expensive to parse but with recent api's in usage that are cheap like the one used by BingO. There are advantages to parsing Google. Simply because there is just more URL's to be scraped. Bing is cheap to parse but it's got very few rewarding URL's and is just really difficult to target on.

Google > Bing.

Chapter 16, Generating vs Handwriting

Generating vs Handwriting, the most bullshit argument in Dorking.

Dorking is built around the construct of simplicity, success and efficiency. The most ideal way to meet most if not all of those criteria is to generate Dorks. BUT, will not always sustain the same quality of Dorks and therefore can cause consequences when adding up expenses of parsing and paying for proxies.

What are Generated Dorks?

Generating Dorks is using a combination of pre-sets with information like Keywords, Parameters, Dork Types and other variables used when making Dorks, and generating every possible combination in relation to that information provided. It's faster than doing it manually, certainly. But sometimes can cause concerns of quality of individual Dorks due to certain keyword or parameter combinations, which isn't ideal when producing Dorks. Fortunately, these problems do have work arounds and are pretty easy. The demand for mass Dorks by big providers is excessive but sometimes cannot match the quality of handmade Dorks.

Problem 1 | Mismatch Combinations

The main concern by a lot of Dork Providers is that there is a mismatch with keywords and parameters. This is a concern because it can invalidate a Dork or completely remove it's results completely. This is not ideal because we're looking to sustain a level of quality when producing Dorks and we want to obtain that quality when we're reaping the rewards of good Dorking Skills.

Solution:

Manually organise your Dork Types to use a singular Keyword Variable and manually sort your keywords with their operators and put into that singular file. The advantages are that you can control which keywords are together, allowing you the customization that regular Dork generation doesn't. Unfortunately, it's very time consuming an takes commitment, but it's avidly rewarding so you can pick and choose your battles on the topic.

Problem 2 | Invalid Dorks (Killing Proxies)

Dorks can be a pain in the arse to work with sometimes. Mainly because we cannot control which queries will succeed and which will fail, this can cause some of those mistakes to chew up, IP ban and other concerns which are unideal for our proxies. This is because when an invalid query occurs, Google can interpret this as an automated search and therefore may up the "risk" rating on your IP before blacklisting it from accessing Google without completing a Recaptcha.

Solution:

Dork Analysers! There is a feature in a few tools called a Dork Analyser, its primary purpose is to parse the Dorks and scrape how many URL's are in the results for each Dork and then if they're below a certain threshold they will be removed from the Dorks in the results file. It's a useful tool and other Dorker's use this and sell those Dorks as "Filtered Dorks". It's certainly appealing a lot of buyers as it meets their needs of high success rates that justify and almost guarantee their expenses. There are risks to this though, some Dorks will return false positive URL rates so that Google can't redirect Hackers to honeypot sites to track their usage on those sites. It's an avid threat but if you follow the syntax requirements in this book there shouldn't be a great concern for this topic.

Handwritten Dorks

Handwritten Dorks have a different advantage to Generated Dorks. Not ideal for time efficiency or the creation of mass lists of Dorks but can allow for the advantage of customization and specifications. With this method you can target certain directories or other regex features for specific parameters, keywords and other factors. This is in the manner that you can customize it uniquely for that Dork making it the highest quality of that Dork around and also providing unique results in relation to your changes.

What are Handwritten Dorks?

Handwriting Dorks is a process of writing Dorks from complete scratch to a level that obtains results. It's time consuming but rewarding if done correctly. You can target unique things unlike Generating Dorks. It's not using any pre-sets but the mind and the information you've obtained to specify your target.

Problems…

It's time consuming and not rewarding for mass targeting. Honestly, I'd rather integrate Handwritten into Generating any day. There are fixes to the problem of speed with Handwritten. If you're okay with time consumption and hand checking each Dork then do it, by all means do it, but it isn't as rewarding as it used to be. Generators are too fast, too strong and too customizable now. Handwriting Dorks cannot compare.

Chapter 17, Exploitation Targeting on Google

Google is an index of the most websites on the internet. That also means there is a lot of badly made sites or protected sites. Except sometimes it's not the way that's popularised in scenes around the internet. SQLi attacks take up over 50% of internet attacks and more severe and rewarding attacks not even 25% of the attacks on the internet. For that reason, for anyone who obtains some fresh 0Days or just something extremely rare or uncommonly known about, there is an opportunity that you need to knowledge to Dork for to find exploitable targets. Hence this chapter on Exploit Targeting.

Most commonly renowned of exploits is the vBulletin and AWS S3 ones that have stormed the internet over recent years. Known exploits like these are not the point of this chapter as they will have already been exploited by other people, instead this chapters is made for people who find exploits while they're 0-day and need assistance with targeting them; And fortunately it's actually easy (most the time).

A perfect Google Dork-able exploit is vBulletin, renowned for its footer, making it simplistic to target. Other exploits for more protected sites can be harder, but the same logic applies.

vBulletin versions before 5.x.x have a footer stating it's a vBulletin forum and also has what version of vBulletin.

These can be Dorked for,

intext:"Powered by vBulletin"

Unfortunately, over time they decided to make the branding of which version selective for payers of 5.x.x. This can be bypassed by the fact they' still have "All times are GMT" in the footer.

For more complicated targets though, other noticeable indicators like directories, footers, headers and branding can be used to target on Google. You can find these important factors to use when targeting the sites that could potentially be vulnerable to your 0Day. Google can only show you indexed data from the page and cannot show you information to target that isn't visible to the user, which is the biggest downfall to Google Dorking aside from IP Restrictions.

Chapter 18, Administrator Panel Targeting

Chapter 19, Bing A-Z

Bing is a rather cheap and simple Search Engine to use when Dorking. This is because of their poor level anti-botting. It's presumed this is because of the lack of traffic on Bing and it would probably be discontinued without us helping them along, unlike Google who gets too many requests and is trying to reduce them to improve advertiser results to increase their profit.

Bing and Google have different Operators and Search Functions. They also have different capabilities. This is rather boring because Bing is inaccurate and just painful to use. I honestly can't bother Dorking on it but due to the mass demand for me to learn it to help everyone. There are a few negative updates that have stopped using URL targeting search functions properly and other operators. This has caused for a decrease in success using Bing.

Chapter 20, Post SQLi Targeting

Chapter 21, LFI Targeting

Revision 3 and this had their name ***Kings of Dorking*** Please Don't Leak

Chapter 22, PublicWWW | Vulnerability & Exploit Targeting

I tried

Chapter 23, Google vs Google API (Custom Search Engine)

Chapter 24, ***Hub | GitHub Dorking.