Identifying Risk Factors for Webserver Compromise

Marie Vasek and Tyler Moore

Computer Science and Engineering Department
Southern Methodist University, Dallas, TX
Email: {mvasek,tylerm}@smu.edu

Abstract. We describe a case-control study to identify risk factors that are associated with higher rates of webserver compromise. We inspect a random sample of around 200 000 webservers and automatically identify attributes hypothesized to affect the susceptibility to compromise, notably content management system (CMS) and webserver type. We then cross-list this information with data on webservers hacked to serve phishing pages or redirect to unlicensed online pharmacies. We find that webservers running WordPress and Joomla are more likely to be hacked than those not running any CMS, and that servers running Apache and Nginx are more likely to be hacked than those running Microsoft IIS. Furthermore, using a series of logistic regressions, we find that a CMS's market share is positively correlated with website compromise. Finally, we examine the link between webservers running outdated software and being compromised. Contrary to conventional wisdom, we find that servers running outdated versions of WordPress (the most popular CMS platform) are less likely to be hacked than those running more recent versions. We present evidence that this may be explained by the low install base of outdated software.

Keywords: Content-management systems, webserver security, case-control study, cybercrime, security economics

1 Introduction

Each month many thousands of websites are compromised by criminals and repurposed to host phishing websites, distribute malware, and peddle counterfeit goods. Despite the substantial harm imposed, the number of infected websites has remained stubbornly high. While many agree that the current level of Internet security is unacceptably low, there is no consensus on what countermeasures should be adopted to improve security or where limited resources should be focused. One key reason we are in such a sorry state is that measuring security outcomes (and what factors drive them) is hard. In part, this is because those who fall victim to cybercrime often prefer not to speak out. But it is also because security mechanisms are deployed in the wild, where it can be impossible to design a randomized controlled experiment isolating the effect of a particular countermeasure to evaluate effectiveness.

However, even when controlled experiments are not feasible, other techniques may still be usefully applied. In this paper, we apply a widely-used method from epidemiology, called a *case-control study*, in order to better understand the factors driving webserver insecurity. Working backwards from data on security incidents and a control sample, we can identify risk factors associated with compromise. This in turn can help defenders better allocate scarce defensive resources to do the most good.

We investigate many observable characteristics of webservers that may affect the likelihood of compromise. Chief among them is whether or not they run a content management system (CMS), an application that simplifies the creation of web content. Some of the more popular CMSes, such as Joomla and WordPress, are consistently exploited to give a miscreant control over the webserver. Additional characteristics include the server type (e.g., Apache), the hosting country, and whether or not the webserver has demonstrated savviness in secure administration practices.

We identify these characteristics in two compromised populations (webservers used to host phishing pages and to engage in search-redirection attacks), as well as a control sample of non-infected webservers. Using the case-control method, we identify risk factors by calculating odds ratios and constructing a series of logistic regressions. Key findings include identifying which CMSes are at greater risk of compromise, demonstrating that CMS popularity is correlated with available exploits and higher rates of compromise, and presenting evidence that outdated WordPress installations are at lower risk of compromise than more recent ones because outdated versions are less popular.

Notably, our analysis focuses on *security outcomes*, not security levels. For instance, we do not claim that running outdated software makes a webserver less "hackable". Rather, by studying compromise data, we can report on what factors affect the likelihood of actually being hacked. We hope that our results demonstrate to others measuring cybercrime the value in employing case-control studies to evaluate outcomes.

The rest of the paper is organized as follows. Section 2 articulates our research questions and describes the data collection methodology. Section 3 presents our empirical results, which we sum up in Section 4. We present related work in Section 5 and discuss limitations, conclusions and opportunities for future work in Section 6.

2 Methodology

We begin by setting out the key research questions in Section 2.1, then outline the case-control study design in Section 2.2. We discuss the data collection and classification approach in Section 2.3. The collected data and analysis scripts are publicly available for replication purposes at `doi:10.7910/DVN/25608`. The methodology is a key contribution of the paper, since applying case-control studies to cybersecurity is new, and, we believe, a promising way to measure security in many other contexts.

2.1 Research Questions

We investigate three categories of research questions about factors that may influence webserver compromise: software type, software market share, and webserver hygiene.

Most generally, we hypothesize that there are measurable differences in compromise rates according to the type of software run on webservers.

H0: Running a CMS is a positive risk factor[1] for compromise.

[1] In this paper, a *positive* risk factor is actually a *bad* thing, as it indicates greater odds of compromise. By contrast, a negative risk factor indicates lower odds of compromise.

H0b: *(corollary)* Some CMS types are risk factors for compromise.
H1: Some server types are risk factors for compromise.

There are several reasons why servers running CMSes may be compromised more often. First, CMSes simplify configuration by reducing technical barriers, which means that they are often administered by non-experts. This could lead to a greater chance for server misconfiguration. Second, CMS platforms are a form of software monoculture, exhibiting common vulnerabilities in both the underlying code and the default configurations. We also expect some CMS platforms to be more secure than others.

We also anticipate that there will be differences in compromise rates based on the type of server software used. This is because there are different amounts of exploitable vulnerabilities present in the underlying code bases. Additionally, some applications (including CMSes) run only or primarily on particular server types, and each application has its own susceptibility to compromise.

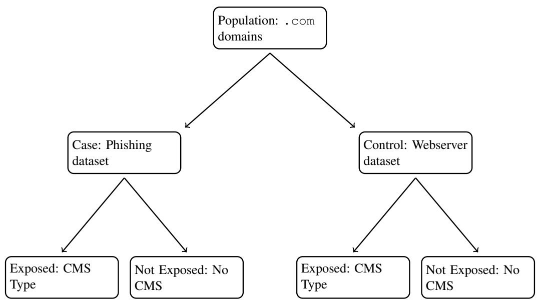
Furthermore, we suspect that a key driving force behind the variation in compromise rates across software types is the software's market share. When more webservers run a particular type of software, they collectively become a more attractive target for miscreants. The cost of crafting new exploits can be amortized over many more infections for more popular software. While many would agree with such logic on software types, we hypothesize that the same logic also applies to different versions of the same software: more popular software versions tend to be targeted more often than less popular ones. We suspect this is true even when the less popular version is more outdated and has more vulnerabilities.

H2: CMS market share is a positive risk factor for webserver compromise.
H2b: *(corollary)* Outdated software with limited market penetration is a negative risk factor for compromise.
H2c: *(corollary)* The number of exploits available for a type of software is a positive risk factor for compromise.

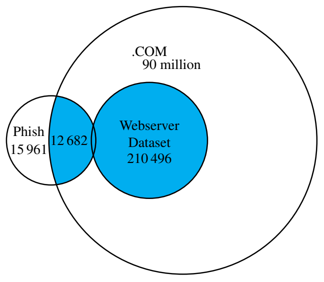
Our final group of hypotheses involve the individual security practices of webserver administrators. We believe that, independent of the software running on a webserver, adopting security best practices that improve server "hygiene" can influence the likelihood of compromise.

H3: Actively hiding detailed software version information is a negative risk factor for compromise.
H4: Running a webserver on a shared hosting platform is a positive risk factor for compromise.
H5: Setting the `HTTPONLY` cookie, which protects against cross-site scripting attacks, is a negative risk factor for compromise.

We note that there are other reasons why a webserver could be put at greater risk of being hacked than just the factors discussed above. For example, administrator competence (not captured by the hygiene indicators) certainly plays a role. Security policies also matter: lax password policies or practices could lead to compromise. Finally, the value of the target influences what gets hacked: high-reputation websites, for instance, are targeted for compromise more frequently in search-redirection attacks [1].



(a) Case-control study design, demonstrated for phishing dataset and CMS type as risk factor.



(b) Venn diagram demonstrates how we join webserver and phishing datasets.

Fig. 1: We join the webserver and compromise datasets to compare risk factors with outcomes.

We have chosen not to examine the impact of these additional factors in the present study. We decided to focus on CMSes, server software, and webserver hygiene indicators for three reasons. First, as explained above, there is substantial evidence that these factors strongly affect compromise rates (e.g., the large number of exploits available that target CMSes). Second, we have restricted ourselves to factors that could manageably be observed directly and in an automated fashion. By contrast, many of the factors that we chose not to study are not not directly observable, such as a company's password policy. Factors that require extensively crawling or fuzzing a domain to observe, such as inferring firewall policies, are also excluded because they cannot be carried out at sufficient scale. Third, we have restricted ourselves to factors that appear in our sample population with sufficient frequency. In particular, we investigated many of the risk factors from [2] and found the vast majority of them to occur too infrequently to include in our study. It is our view that the methods of analysis presented here could in fact be applied to additional factors, but we defer the task to future work.

2.2 Case-Control Study Design

In a case-control study typically used in epidemiology, data on those afflicted with a disease are compared against as similar a population as possible of those not afflicted [3]. For example, in the seminal case-control study that uncovered the link between smoking and lung cancer, Doll and Arthur surveyed British doctors about their smoking habits, then compared it against data collected subsequently on doctors' mortality rates [4]. They found that doctors who smoked were much more likely to die than doctors who did not. In general, case-control studies work by comparing two populations, one with a condition (the 'case') to one without who are otherwise similar (the 'control'). Researchers can then work backwards to identify important risk factors by comparing the relative incidence of different characteristics in the case and control populations.

Similarly, we sample a population of webservers and compare them to other populations of webservers that have been compromised. Figure 1a demonstrates the design for the phishing dataset. We start with a comparable webserver population – domains registered in `.com`. We then assign the `.com` domains from the phishing dataset as the case and the domains from the webserver dataset as the control. We can then treat characteristics such as CMS type, server type and hosting country as potential risk factors. (We explain how each of these datasets and risk factors are collected in the next subsection below.) Figure 1b shows a Venn diagram that explains how the phishing and webserver datasets are joined. A similar approach is used for the search-redirection attacks dataset and the webserver dataset.

Note that with case-control data, we do not make any claims about the overall incidence of compromise in the population. This is because we compare two different samples (the compromised and broader samples). Instead, we analyze the prevalence of compromise relative to the occurrence of risk factors such as CMS type.

2.3 Data Collection Overview

Control Population: Webserver Sample To answer our research questions, we need a random sample of webservers; however, obtaining a perfectly representative sample of all webservers is not possible since there is no global list available from which to sample. According to Verisign, there are over 252 million registered domains [5], but most zone files listing domains are not made public. Instead, we take a random sample of domains listed in the `.com` zone file. While limited to a single TLD, it is worth noting that `.com` comprises nearly half of all registered domains, and it is used by websites in many countries. Furthermore, `.com` domains include websites from a wide range of popularities. Thus, we feel that sampling from `.com` is broad enough to be representative of all webservers online.

We sampled webservers over a period of 9 days, obtaining information on 210 496 domains selected at random from the `.com` zone file downloaded January 15, 2013. We chose this sample size to ensure that it would likely include enough websites running CMSes with at least 1% market share. This, in turn, improves the chances of obtaining statistically significant results.

We remove all free hosting and URL shortening services (where the URLs are likely set up purposely by the criminals) from our collection. Finally, we refer to the trimmed sample of `.com` domains as the webserver dataset.

Case Populations: Compromised Webservers We consider two sources of data on webserver compromise. First, we examine an amalgamated “feed” of phishing URLs, comprising real-time reports from two firms that remove phishing websites on behalf of banks, a large brand owner, the crowdsourced list from PhishTank [6], and the Anti-Phishing Working Group’s community feed [7]. We examined 97 788 distinct URLs from 29 682 domains impersonating 1 098 different brands reported between November 20, 2012 and January 7, 2013 in the phishing dataset. According to [8], 94% of domains used for phishing during this period were compromised websites. Nearly all of the remainder are highly-ranked sites that we excluded as described below.

The second dataset on webserver compromise came from websites observed to be engaging in search-redirection attacks. Here, websites with high reputation are hacked and reconfigured to surreptitiously channel traffic from search engines to unlicensed pharmacies. We obtained the dataset gathered by the authors of [1], who updated their system to detect advanced forms of cookie-based redirection as described in [9]. The dataset includes web search results from 218 pharmaceutical-related search terms. Webservers are included in the list if they are observed to redirect to a third-party website and subsequently found to engage in cloaking. The search-redirection attacks dataset includes 58 516 distinct URLs gathered between October 20, 2011 and December 27, 2012. These correspond to 10 677 unique domains, 6 226 of which have a `.com` TLD.

Extracting Webserver Risk Factors The head of an HTML webpage often contains metadata about the webpage in so-called meta tags. One piece of information that many content management system (CMS) authors (and text editors) include is a "generator" tag. This optional tag generally contains the text editor type, content management system, version number and/or any special CMS themes used. For example, a website running WordPress version 3.2.1 might contain the tag `<meta name=''generator'' content=''WordPress 3.2.1''>`. We downloaded a copy of the HTML for the top-level webpage on a given domain, and then parsed the HTML to extract the tag.

We then attempted to identify the CMS, if any, along with the version information if included. We used manually crafted regular expressions to complete the task. We focused on the top 13 CMSes with at least 1.0% of CMS market share as of January 2013 according to W^{3}Techs [10]. These 13 CMSes collectively comprise 88.4% of all websites using CMSes. We could identify CMS type for 9 of the top 13 (84.6% of all CMSes). We also included 3 more CMSes, each with less than 1.0% of market share.

However, we cannot solely rely on generator tags to classify websites by CMS. For instance, most websites running Drupal, one of the most popular CMSes, do not display generator information in their metadata. Consequently, in addition to gathering generator information, we ran a number of regular expressions corresponding to 3 of the 4 most popular CMSes against the dataset. Appendix A compares our custom approach to several off-the shelf tools for CMS identification.

To identify server software, we collected the packet headers along with the HTML code. In each header was a line specifying the server such as `Server: Microsoft-IIS/7.5`. From this we extracted the server type and version number. We also fetched the IP address of the server and mapped this to the country of origin using MaxMind[11].

Reducing False Positives in the Infection Datasets Not all of the URLs in the compromise datasets are from hacked webpages. For the phishing dataset, we deem any URL to be a *false positive* if the URL does anything other than impersonate another website. For the search-redirection attacks dataset, we classify any URL as a false positive if the destination website following redirection appears related to the source website (e.g., `ilike.com` redirects to `myspace.com`, which bought the company).

Since the false positive rates for phishing are consistently higher than for search-redirection attacks, we developed automated techniques to discard websites that were errantly placed on these lists. We removed all FQDNs that redirected to legitimate

US-based banks[2] and other known non-banks frequently targeted by phishing, such as `paypal.com`, `amazon.com` and `facebook.com`. We also generated a sequence of regular expressions that detected Microsoft Outlook Web Applications and coupon websites and checked them against the HTML we downloaded previously. These initial steps reduced our overall false positive rate for the phishing dataset from 9.4% to 5.0%. To further improve, we manually inspected all URLs in the Alexa top million sites and excluded any false positives from further consideration, yielding final false positive rates of 2.3% for phishing and 4.3% for search-redirection attacks. These false positive rates were calculated by inspecting a stratified random sample by Alexa rank.

3 Empirical Results

Having detailed our methodological approach, we now turn to the results. In Section 3.1 we use odds ratios and in Section 3.2 we use logistic regression to identify which server characteristics are associated with higher and lower rates of compromise. Then in Section 3.3 we focus on how outdated software affects compromise in WordPress installs.

3.1 Finding Risk Factors for Compromise

Odds are defined by the ratio of the probability that an event will occur to the probability it will not occur. For example, if $p = 0.2$, then the odds are $\frac{p}{1-p} = \frac{0.2}{0.8} = 0.25$. Odds express relative probabilities. *Odds ratios* compare the odds of two events, each occurring with different probabilities.

In case-control studies, odds ratios compare the odds of a subject in the case population exhibiting a risk factor to the odds of a subject in the control population exhibiting a risk factor. Consider the four cases:

	Case (afflicted)	Control (not afflicted)
Has risk factor	p_{CaseRF}	p_{CtlRF}
No risk factor	$p_{\text{Case}\overline{\text{RF}}}$	$p_{\text{Ctl}\overline{\text{RF}}}$

The odds ratio, then, is the following product of probabilities:

$$\text{odds ratio (OR)} = \frac{p_{\text{CaseRF}}/p_{\text{Case}\overline{\text{RF}}}}{p_{\text{CtlRF}}/p_{\text{Ctl}\overline{\text{RF}}}} = \frac{p_{\text{CaseRF}} * p_{\text{Ctl}\overline{\text{RF}}}}{p_{\text{Case}\overline{\text{RF}}} * p_{\text{CtlRF}}}$$

An odds ratio of 1 means that there is no difference in proportions of the risk factor among the case and control groups. An odds ratio greater than 1 indicates that those in the case group are more likely to exhibit the risk factor (so-called *positive* risk factors). By contrast, an odds ratio less than 1 indicates that those in the case group are less likely to exhibit the risk factor (indicating a *negative* risk factor).

Odds Ratio Results Table 1 reports odds ratios for different CMS and server types for both compromise datasets. We computed odds ratios for webservers running each of the major CMSes compared to webservers not running any CMS. For the phishing

[2] Found on the FDIC website [12].

	Content Management System (CMS) Type									
	Phishing dataset					Search-redirection attacks dataset				
	Risk factor	Odds ratio	95% CI	# Phish	# Not phish	Risk factor	Odds ratio	95% CI	# Redir.	# Not redir.
No CMS		1.00		8 747	190 305		1.00		2 260	190 314
WordPress	+	**4.44**	(4.24, 4.65)	2 673	13 101	+	**17.18**	(16.20, 18.22)	2 674	13 106
Joomla	+	**7.11**	(6.62, 7.63)	1 106	3 384	+	**23.96**	(22.05, 26.04)	963	3 385
Drupal		0.79	(0.58, 1.04)	46	1 279	+	**6.59**	(5.33, 8.07)	100	1 279
Zen Cart	+	**4.84**	(3.26, 6.96)	33	149		2.35	(0.71, 5.56)	4	149
Blogger	–	**0.28**	(0.13, 0.52)	8	637		1.08	(0.49, 2.02)	8	637
TYPO3	–	**0.14**	(0.03, 0.37)	3	481	+	**4.23**	(2.72, 6.24)	24	481
Homestead	–	**0.04**	(0.00, 0.18)	1	607	–	**0.16**	(0.01, 0.69)	1	607

	Server Type									
	Phishing dataset					Search-redirection attacks dataset				
	Risk factor	Odds ratio	95% CI	# Phish	# Not phish	Risk factor	Odds ratio	95% CI	# Redir.	# Not redir.
Microsoft IIS		1.00		1 002	60 495		1.00		193	60 497
Apache	+	**5.44**	(5.10, 5.81)	10 549	117 017	+	**14.12**	(12.26, 16.36)	5 276	117 031
Nginx	+	**2.24**	(2.01, 2.50)	507	13 649	+	**8.63**	(7.26, 10.30)	376	13 649
Yahoo	–	**0.62**	(0.41, 0.89)	27	2 634		1.57	(0.85, 2.64)	13	2 634
Google		0.63	(0.35, 1.03)	14	1 359		1.88	(0.84, 3.57)	8	1 359

Table 1: Odds ratios for varying CMS and server types.

dataset, some less popular CMSes fare better than not using a CMS, but the more popular CMSes are positive risk factors. WordPress, Joomla and Zen Cart had increased odds of compromise, while Blogger, TYPO3 and Homestead reduced risk. This supports hypothesis **H0b**, but partially refutes hypothesis **H0** that using *any* CMS increases the odds of compromise. For search-redirection attacks, CMSes are either as bad or worse than not using a CMS, supporting **H0**. Notably, the odds ratios for Joomla and WordPress are even higher than for phishing. The WordPress odds ratio jumps from 4.4 phishing to 17 for search-redirection attacks; for Joomla, the jump is from 7 to nearly 24!

For some smaller CMSes, the evidence for phishing and search-redirection attacks is mixed. Homestead has a negative risk factor for phishing and search-redirection attacks dataset. TYPO3 and Blogger are negative for phishing, but TYPO3 has a positive risk factor for search-redirection attacks, whereas Blogger is not statistically significant.

We note that the larger CMSes tend to be the strongest positive risk factors for compromise, according to both datasets. This supports hypothesis **H2** that CMS market share is positively correlated with compromise, but more analysis is needed.

For server software type, we compute risk factors relative to Microsoft IIS, the second-most popular server software. Apache and Nginx are positive for both phishing and search-redirection attacks. Note that we are not making any claims about the relative security levels of the different software classes. All software contains vulnerabilities, and we are not taking sides on the debate over whether open- or closed-source software has fewer unpatched holes [13]. Instead, our results simply show that, relative to software popularity, criminals tend to use Apache and Nginx more for perpetrating their crimes than Microsoft IIS.

3.2 Explaining Why Compromise Rates Vary

We now present logistic regressions to study why websites are compromised. We run four regressions in all: two for webservers running a CMS (one each for the phishing and search-redirection attacks datasets) and two for webservers not running any CMS (one for each compromise dataset). We run the additional regressions because some explanatory variables only apply to CMSes, but many of the variables measuring security signals apply regardless of whether or not a webserver uses a CMS.

We group the following explanatory variables into three categories: CMS market share, webserver hygiene and server attributes.

CMS Market Share

# Servers: We took market share for each CMS from [10] as of January 1, 2013 and multiplied it by population of registered `.com` domains (106.2 million) and estimated server response rate (85%) [5]. This variable was omitted for non-CMS regressions.

Webserver Hygiene

`HTTPONLY` cookie: We checked the header for an `HTTPONLY` cookie used to protect against cross-site-scripting attacks. We interpret setting this cookie as a positive signal of overall server hygiene. Checking for this cookie was one measure of server hygiene also used in [2].
Server Version Visible: We analyzed the server headers for any version information regarding the server, whether it be Apache 2 or Apache 2.2.22. This is a Boolean variable which is true if the server gave any potentially valid version information.
Shared Hosting: We counted the number of times we observed an IP address in the combined webserver and compromised datasets. We deem a domain to be part of a shared host if 10 domains resolve to the same IP address. A recent Anti-Phishing Working Group report presents evidence that some attackers target shared hosting in order to simultaneously infect many domains [8].

Server Attributes

Country: We took the top ten countries from the combined dataset and compared each of them the domains hosted in all the other countries in the dataset.
Server Type: This categorical variable looks at the type of server software a webserver is running. We only consider the 5 most popular types: Apache, Microsoft IIS, Nginx, Google, and Yahoo.

The model takes the following form:

$$\log \frac{p_{comp}}{1 - p_{comp}} = c_0 + c_1 \text{ lg (\# Servers)} + c_2 \text{ HTTPONLY} + c_3 \text{ Server Vsn?}$$
$$+ c_4 \text{ Shared Hosting?} + c_5 \text{ Country} + c_6 \text{ Server type} + \varepsilon$$

Table 2 shows the results from these four regressions. CMS popularity is positively correlated with compromise in the phishing dataset. Each doubling of the number of webservers running the CMS increases the odds of compromise by 9%, supporting hypothesis **H2**. The result is inconclusive for search-redirection attacks, but the trend is similar. Also, Appendix B studies the link between market share and exploitability. The analysis in Appendix B shows that the number of exploits is also a positive risk factor for being hacked to serve phishing pages, which supports **H2c**.

We consider hygiene variables next. We do not observe any consistent evidence that hiding server information promotes or inhibits compromise, so we can neither refute

	CMS						No CMS					
	Phish			Search-redirection attacks			Phish			Search-redirection attacks		
	coef.	odds	p-value	coef.	odds	p-value	coef.	odds	p-value	coef.	odds	p-value
Intercept	-4.77	**0.01**	< 0.0001	-4.10	**0.02**	< 0.0001	-4.11	**0.02**	< 0.0001	-5.99	**0.00**	< 0.0001
lg # Svrs	0.09	**1.09**	< 0.0001	0.02	1.02	0.16						
HTTPONLY	0.22	1.25	0.06	-0.83	**0.44**	< 0.0001	-0.87	**0.42**	< 0.0001	0.15	1.17	0.12
No Svr Vsn	-0.15	**0.86**	0.0001	0.10	**1.11**	0.01	0.04	1.04	0.09	0.32	**1.38**	< 0.0001
Shared Host	0.95	**2.58**	< 0.0001	-1.58	**0.21**	< 0.0001	0.28	**1.32**	< 0.0001	-1.27	**0.28**	< 0.0001
Apache	1.49	**4.45**	< 0.0001	1.48	**4.38**	< 0.0001	1.80	**6.06**	< 0.0001	1.37	**3.94**	< 0.0001
Nginx	0.59	**1.80**	0.003	1.37	**3.93**	< 0.0001	0.70	**2.00**	< 0.0001	1.43	**4.19**	< 0.0001
Yahoo	-0.34	0.72	0.59	2.72	**15.12**	< 0.0001	-0.54	**0.58**	0.009	-0.02	0.98	0.97
Google	-1.50	**0.22**	0.0003	-0.81	0.44	0.10	-0.36	0.70	0.35	0.25	1.29	0.67
Other	1.92	**6.84**	< 0.0001	0.83	**2.30**	0.0009	0.81	**2.24**	< 0.0001	0.96	**2.62**	< 0.0001
Model fit:	$\chi^2 = 1\,353, p < 0.0001$			$\chi^2 = 1\,825, p < 0.0001$			$\chi^2 = 5\,937, p < 0.0001$			$\chi^2 = 2\,113, p < 0.0001$		

Table 2: Table of coefficients for logistic regressions comparing rate of compromise to many explanatory variables.

nor support **H3**. Setting an `HTTPONLY` cookie appears to be a negative risk factor for being compromised, but we need more data to support the associated hypothesis **H5**.

Running on a shared host is a positive risk factor for being hacked to serve phishing pages, which supports **H4** and findings from [8]. However, we note that it is a negative risk factor for being hacked for search-redirection attacks. It appears that cybercriminals engaged in phishing have adopted different techniques for infecting webservers than those carrying out search-redirection attacks. Further investigation shows that there is a correlation between being on a shared host and having a low or no Alexa rank: 13% of the top 10M, 26% of the next 10M, and 55% of websites without an Alexa rank are hosted on a shared host (from our combined webserver and search-redirection attacks dataset). This result could signal that search-redirection attacks attackers target higher ranked pages, which makes sense in light of [1], which showed that compromised websites with a higher PageRank stay in search results longer.

Previous results from webservers in Section 3.1 are similar to those in this regression – notably that Apache and Nginx webservers remain positive risk factors compared to Microsoft IIS in all cases.

Finally, we note that there is more consistency between the regressions examining CMSes and no CMSes than there is between regressions for phishing and search-redirection attacks. The results for the shared host variable are the same, regardless of whether a CMS is used, as are the results for server types and most countries. Only the practice of hiding detailed server version information was very inconsistent, being a negative risk factor for phishing on CMSes and a negative risk factor for search-redirection attacks when no CMS is used.

3.3 Does Outdated Software Get Hacked More?

A best practice for webserver security is to run the most recent version of software available, as updates tends to plug security holes as well as add new features. For instance, Google notifies webmasters via its Webmaster Tools when it detects outdated server software as a way to improve security[14]. However, updating server software can be a nuisance, due to cross-dependencies, poor interfaces and the demands of maintaining uptime. Consequently, many webservers run software that is many months, or even years, out of date. The security firm Sucuri Labs even runs a website[15] that names and shames websites running woefully outdated CMS or server software.

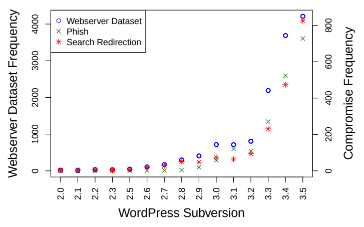
But we wondered whether or not servers running outdated software actually do get compromised more often than those that do not. We hypothesize that the opposite is usually true: that outdated webservers are compromised less often provided that most other webservers are already upgraded. To test this and related hypotheses, we restrict ourselves to the servers running WordPress. This is for two reasons: WordPress is the most popular content management system and, by default, WordPress installs provide detailed version information ordered straightforwardly.

Odds Ratios for Major Version Differences First, we investigated whether servers running WordPress that hid version information were at less risk of compromise (to test hypothesis **H3**). The results are shown in the first row of the table in Figure 2c. In fact, hiding WordPress version is a positive risk factor for being hacked for phishing pages. This contradicts the frequently held view that hiding detailed version information improves security, and it instead lends credence to the view that publishing information helps defenders more than attackers. For instance, WordPress and Google send out reminder emails to server administrators to update their software, but those who obscured their generator version for security reasons do not receive the reminders. We also note that even though we looked at version information through the generator tag, attackers oftentimes try their hack on any server running WordPress, regardless of what version it says it is. We see no statistically significant effect for search-redirection attacks, though the trend is similar.

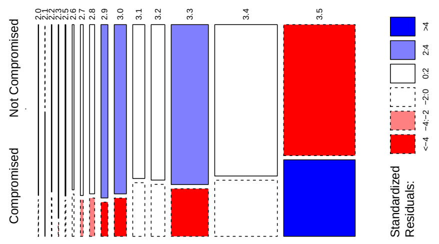
There are differing degrees of outdated software. For servers with version information, we first compared the risk facing servers at the most recent version (3.5.1 during our collection time) to running any other version of WordPress. Running the most up-to-date version is a positive risk factor for being hacked for search-redirection attacks. This too goes against conventional wisdom, and indirectly supports hypothesis **H2** since the most recent version is also the most popular one.

We also looked at the difference in major versions, ignoring version 1 since we only had 7 instances in our combined datasets. We compared all of WordPress 2.* and WordPress 3.* against WordPress installs with no version information. We see that WordPress 3.* installs face more risk of being hacked to serve phishing pages than WordPress 2.*. We observe similar but statistically insignificant results for search-redirection attacks.

Chi-squared Test for Risk Across Subversions The odds ratios just discussed offer initial evidence that being out of date reduces the risk of infection for webservers running WordPress, at least when comparing major versions. We now drill down and investigate differences across WordPress subversions (e.g., WordPress 3.3.*). Figure 2a plots the relative frequency of servers in our webserver and compromise datasets running each WordPress subversion. Note the different scales to the vertical axes – the left axis tracks the frequency in the webserver dataset while the right axis is used for the two compromise datasets. We first observe that more outdated subversions are indeed less popular compared to the most recent subversions. We also see that the compromise



(a) Incidence of compromise by WordPress version, along with the popularity of WordPress version.



(b) Mosaic plot of WordPress version popularity and incidence of compromise (red cells indicate statistically significant underrepresentation, blue cells overrepresentation).

	Phishing dataset					Search-redirection attacks dataset				
	Risk factor	Odds ratio	95% CI	# Phish	# Not phish	Risk factor	Odds ratio	95% CI	# Redir.	# Not redir.
Version Found		1.00		1 834	9 676		1.00		1 936	9 680
No Version	+	**1.29**	(1.18, 1.41)	839	3425		1.08	(0.98, 1.18)	738	3 426
Other WordPress versions		1.00		1 606	8 599		1.00		1 440	8 601
WordPress 3.5.1		1.13	(0.97, 1.32)	228	1 077	+	**2.75**	(2.43, 3.09)	496	1 079
No Version		1.00		839	3 425		1.00		738	3 426
WordPress 2.*	–	**0.12**	(0.08, 0.17)	26	918		0.88	(0.73, 1.05)	173	918
WordPress 3.*	–	**0.84**	(0.77, 0.92)	1 808	8 751		0.93	(0.85, 1.03)	1 762	8 755

(c) Odds ratios by WordPress versioning.

Fig. 2: Exploring the relationship between WordPress version and the incidence of webserver compromise.

rate roughly follows the popularity of the subversion, but with substantial variation and lower compromise rates for more outdated versions.

But are the differences in compromise rates statistically significant? We can answer that using a χ^2 test, but first, we can inspect the differences visually using the mosaic plot in Figure 2b. The vertical axis shows for each version the proportion of compromised webservers (either phishing or search-redirection attacks) compared to the proportion of uncompromised webservers (from the webserver dataset). The horizontal axis is scaled so that the area of each cell matches the frequency of each category. For instance, the dark blue cell in the bottom right corner shows the proportion of webservers running WordPress Version 3.5.* that have been compromised. This plot shows that the fraction compromised falls steadily as the subversions grow more outdated. It also shows that the collective proportion of outdated servers is still quite substantial.

Finally, the cells are lightly shaded if the difference in proportion for being compromised is statistically significant at the 95% confidence interval according to the χ^2 test, and over 99% confidence interval if darkly shaded. Red cells are underrepresented and blue cells are overrepresented. We can see that most of the WordPress 2.* versions are statistically overrepresented in the webserver dataset and underrepresented in the compromise datasets. WordPress 3.0 and 3.3 are also overrepresented in the compromise datasets and underrepresented in the webserver dataset. The most recent, WordPress 3.5, is the only subversion overrepresented in the phish dataset and underrepresented in the webserver dataset. These findings support hypothesis **H2b** that unpopular outdated CMSes are negative risk factors for compromise. It is also consistent with our findings from the odds ratios that the most recent version is the most at risk of compromise.

Logistic Regressions The final check we make comparing compromise rates in WordPress versions is to run a simple logistic regression comparing the popularity of a version to the compromise rate in the phishing dataset.
# Servers: We took the market share for each WordPress subversion from [10] as of January 1, 2013 and multiplied it by population of registered .COM domains (106.2 million) and the estimated server response rate (85%) from [5].

$$\log \frac{p_{comp}}{1 - p_{comp}} = c_0 + c_1 \text{ lg (\# Servers)} + \varepsilon.$$

The logistic regression yields the following results:

	coef.	Odds Ratio	95% conf. int.	Significance
Intercept	-5.60	**0.00**	(0.00, 0.01)	$p < 0.0001$
lg(# Servers)	0.19	**1.20**	(1.17, 1.24)	$p < 0.0001$
Model fit:	$\chi^2 = 200.31, p < 0.0001$			

These results show that each time the number of servers running the same subversion of WordPress doubles, the risk of the server being hacked to serve phishing pages increases by 20%. This offers further evidence supporting **H2**.

4 Discussion

We now sum up the results of the prior sections by first revisiting the original hypotheses and second discussing how the results can be leveraged by security engineers.

Evaluating Research Questions We summarize the analysis of the previous section by returning to the original research questions.

H0 *(Running a CMS pos. RF)* Supported for search-redirection attacks, not uniformly for phishing
H0b *(Some CMS types are RFs)* Broadly supported
H1 *(Some server types are RFs)* Broadly supported
H2 *(CMS market share pos. RF)* Broadly supported, across all CMSes and across WordPress subversions
H2b *(Outdated unpopular software neg. RF)* Supported across WordPress subversions
H2c *(# exploits pos. RF)* Supported
H3 *(Hiding version info neg. RF)* Contradicted
H4 *(Shared hosting pos. RF)* Supported for phishing, contradicted for search-redirection attacks

H5 *(`HTTPONLY` cookie pos. RF)* Inconclusive

Many hypotheses are broadly supported, especially that server type and CMS market share are positive risk factors. We find less support for hypothesis **H0** that *all* CMSes exhibit higher rates of compromise; instead, *most* CMSes, especially the popular ones, are positive risk factors for compromise. Finally, it does not appear that hiding version information is a negative risk factor in most circumstances, but it is unclear how often it may be a positive risk factor.

Making the Results Actionable So what can be made of these results? At a high level, the findings can help reduce information asymmetries regarding security outcomes for different webserver configurations [16]. By making security outcomes such as compromise incidents more directly comparable across platforms, we can help others make more informed decisions about the relative risks posed. Publishing such data can also motivate software developers to improve the security of their code.

We have seen, however, that not all "name-and-shame" policies are consistent with empirical observation. Notably, efforts to call out websites running outdated software are misguided, since they obscure our finding that up-to-date servers tends to be hacked more often. Instead, relative metrics such as odds ratios can be used to identify the worst offenders and apply peer pressure to improve. They can also be used as positive reinforcement by encouraging everyone to improve compared to others.

For the system administrator, our results can be applied in two ways. First, the results can be used to make better choices when choosing among available software types and configuration. Second, after systems have been deployed, the findings can be used to manage heterogeneous configurations (e.g., environments with multiple CMSes and server software types). Here, administrators can prioritize how defensive countermeasures such as attack detection should be deployed. Security policies could even be set in accordance with the observed relative risk.

More broadly, we have demonstrated a general method of studying how webserver characteristics affect the risk of compromise. The methods presented here can be applied to other characteristics if the the data can be collected. Furthermore, odds ratios help to identify relationships that should be tested further using experimental methods.

5 Related Work

While often challenging to carry out, substantial progress has been made over the past several years in conducting large-scale measurements of cybercrime. Some work is particularly relevant due to the results from studying the security of webservers. For instance, Doupe et al. describe a state-aware fuzzer in which they evaluate vulnerabilities in CMS platforms [17]. Scholte et al. study vulnerabilities in CMS platforms, though they do not relate vulnerabilities to exploits or observed compromise [18]. Nikiforakis et al. crawl many webpages on top webservers to measure the quality of third-party JavaScript libraries running on the webservers [2].

Another series of papers are relevant to the compromise datasets we study. For example, Wang et al. performed a large-scale study of cloaking, which is often caused by search-redirection attacks [19]. Notably, the authors dealt with false positives using clustering. While our data source on search-redirection attacks focuses exclusively on redirections to unlicensed pharmacies [1], the attack technique is general [20].

A number of studies deploy methods in common with our own. Notably, Lee describes the use of a small case-control study to identify characteristics that predispose academics to spear-phishing attempts [21]. We adopt one of the signals of security hygiene used by [2], while Pitsillidis et al. measure the purity of spam feeds in a manner consistent with how we detect false positives in our compromise datasets [22].

Many studies have been primarily descriptive in nature, though some have managed to tease out the factors affecting the prevalence and success of attacks. For instance, Ransbotham connected vulnerability information with intrusion detection system data to show that open-source software tends to be exploited faster than closed-source software following vulnerability disclosure [23].

Our work is distinguished from prior work in two ways. First, we focus extensively on the relationship between webserver characteristics, notably CMS type and market share, and compromise. Second, we use the case-control method to understand the characteristics of large cybercrime datasets.

6 Concluding Remarks

We have presented a case-control study identifying several webserver characteristics that are associated with higher and lower rates of compromise. We joined two datasets on phishing and search-redirection attacks with a large sample of webservers, then automatically extracted several characteristics of these webservers hypothesized to affect the likelihood the webserver will be compromised.

Supported by statistical methods of odds ratios and logistic regression models, we found that certain server types (notably Apache and Nginx) and content management systems (notably Joomla and WordPress) face higher odds of compromise, relative to their popularity. We also found that a key driving factor behind which CMSes are targeted most is the underlying popularity of the platform. We presented evidence that this was true across CMS types, as well as for less popular but outdated subversions of WordPress. In many respects, this finding can be thought of as a webserver-based corollary to the old truism for desktop operating systems that Macs are more secure than PCs because they have less market share.

There are a number of limitations to the present study that can be addressed in future work. First, the findings of case-control studies should be complemented by other forms of experimentation that directly isolate explanatory factors when possible. It is our hope that our findings may be further validated using different approaches.

Another limitation of the current study is that there is a delay between the time of reported compromise and the identification of risk factors. It is possible that some of the webservers may have changed their configurations before all indicators could be gathered. There is a trade-off between collecting large data samples and the speed at which the samples can be collected. In this paper, we emphasized size over speed. In future work, we aim to close the gap between compromise and inspection to improve the accuracy of our CMS and software classifications.

Other opportunities for further investigation include carrying out a longitudinal study of these risk factors over time. Incorporating additional sources of compromise data, notably servers infected with drive-by-downloads, could be worthwhile. We would like to construct a control sample for domains other than `.com`, since others have shown that different TLDs such as `.edu` are frequently targeted [1].

Finally, we are optimistic that the case-control method employed here may be applied to many other contexts of cybercrime measurement. It is our hope that doing so will lead to deeper understanding of the issues defenders should prioritize.

Acknowledgments

This work was partially funded by the Department of Homeland Security (DHS) Science and Technology Directorate, Cyber Security Division (DHS S&T/CSD) Broad Agency Announcement 11.02, the Government of Australia and SPAWAR Systems Center Pacific via contract number N66001-13-C-0131. This paper represents the position of the authors and not that of the aforementioned agencies.

References

1. N. Leontiadis, T. Moore, and N. Christin, "Measuring and analyzing search-redirection attacks in the illicit online prescription drug trade," in *Proceedings of USENIX Security 2011*, San Francisco, CA, Aug. 2011.
2. N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. V. Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "You are what you include: Large-scale evaluation of remote JavaScript inclusions," in *ACM Conference on Computer and Communications Security*, 2012, pp. 736–747.
3. J. Schlesselman, *Case-control studies: design, conduct, analysis*. Oxford University Press, USA, 1982, no. 2.
4. R. Doll and A. Hill, "Lung cancer and other cuases of death in relation to smoking; a second report on the mortality of british doctors," *British Medical Journal*, vol. 2, pp. 1071–1081, Nov. 1956.
5. Verisign, "The domain name industry brief," Apr. 2013, https://www.verisigninc.com/assets/domain-name-brief-april2013.pdf. Last accessed May 1, 2013.
6. "PhishTank," https://www.phishtank.com/.
7. "Anti-Phishing Working Group," http://www.antiphishing.org/.
8. APWG, "Global phishing survey: Trends and domain name use in 2H2012," 2013, http://docs.apwg.org/reports/APWG_GlobalPhishingSurvey_2H2012.pdf. Last accessed May 5, 2013.
9. N. Leontiadis, T. Moore, and N. Christin, "Pick your poison: pricing and inventories at unlicensed online pharmacies," in *ACM Conference on Electronic Commerce*, 2013.
10. W3techs, "Market share trends for content management systems," http://w3techs.com/technologies/history_overview/content_management/. Last accessed May 3, 2013.
11. "MaxMind GeoIP," https://www.maxmind.com/en/geolocation_landing.
12. "FDIC institutions," http://www2.fdic.gov/idasp/Institutions2.zip.
13. J.-H. Hoepman and B. Jacobs, "Increased security through open source," *Communications of the ACM*, vol. 50, no. 1, pp. 79–83, 2007.
14. P. Chapman, "'New software version' notifications for your site," http://googlewebmastercentral.blogspot.com/2009/11/new-software-version-notifications-for.html.

15. “URLFind,” http://urlfind.org/.
16. R. Anderson and T. Moore, “The economics of information security,” *Science*, vol. 314, no. 5799, pp. 610–613, Oct. 2006.
17. A. Doupe, L. Cavedon, C. Kruegel, and G. Vigna, “Enemy of the State: A State-Aware Black-Box Vulnerability Scanner,” in *Proceedings of the USENIX Security Symposium*, Bellevue, WA, August 2012.
18. T.Scholte, D. Balzarotti, and E. Kirda, “Quo vadis? A study of the evolution of input validation vulnerabilities in web applications,” in *Financial Cryptography and Data Security*. Springer, 2012, pp. 284–298.
19. D. Wang, S. Savage, and G. Voelker, “Cloak and dagger: Dynamics of web search cloaking,” in *Proceedings of the 18th ACM Conference on Computer and Communications Security*. ACM, 2011, pp. 477–490.
20. Z. Li, S. Alrwais, Y. Xie, F. Yu, and X. Wang, “Finding the linchpins of the dark web: A study on topologically dedicated hosts on malicious web infrastructures,” in *34th IEEE Symposium on Security and Privacy*, 2013.
21. M. Lee, “Who’s next? identifying risks factors for subjects of targeted attacks,” in *Proceedings of the Virus Bulletin Conference*, 2012, pp. 301–306.
22. A. Pitsillidis, C. Kanich, G. Voelker, K. Levchenko, and S. Savage, “Taster’s choice: A comparative analysis of spam feeds,” in *ACM SIGCOMM Conference on Internet Measurement*, 2012, pp. 427–440.
23. S. Ransbotham, “An empirical analysis of exploitation attempts based on vulnerabilities in open source software,” in *Proceedings (online) of the 9th Workshop on Economics of Information Security*, Cambridge, MA, Jun. 2010.
24. “BlindElephant web application fingerprinter,” http://blindelephant.sourceforge.net/.
25. “WhatWeb,” http://whatweb.net/.
26. “Plecost,” https://code.google.com/p/plecost/.
27. “Exploit database,” http://www.exploit-db.com.

A Comparison of Methods to Identify CMS Type

While a number of tools provide CMS detection as part of more general-purpose web-service fingerprinters (e.g., BlindElephant[24], WhatWeb[25] and the WordPress-specific Plecost[26]), we opted to build the custom CMS detector described above to improve efficiency and accuracy over existing tools. Both BlindElephant and Plecost issue many HTTP requests to characterize each server. We ruled these tools out because we needed a lightweight solution that could quickly detect CMS type and version for hundreds of thousand webservers. Like our method, WhatWeb issues a single HTTP request per server (at its lowest “aggressiveness” level). Combined with its multi-threaded design, WhatWeb should offer fast identification of CMS versions. We therefore decided to evaluate its performance and accuracy compared to our own system.

We selected 2 000 random URLs from the webserver dataset and attempted to identify the CMS type using our system and WhatWeb’s. In terms of efficiency, we were surprised to find that WhatWeb took nearly twice as long to finish, despite being multithreaded. We speculate that the difference in speed can be attributed to its general-purpose nature. We also found that our system was substantially more accurate, identifying the correct CMS on more websites and having far fewer inaccurate classifications. We manually inspected all disagreements between WhatWeb and our tool in order to establish the following detection, false positive and false negative rates:

Method	FN Rate	FP Rate	TN Rate	TP Rate	# Results
WhatWeb	40.7%	6.1%	74.3%	59.3%	1 297
Our Method	5.4%	0.1%	99.0%	92.2%	1 674

Based on these findings, we conclude that our custom method is best-suited to the task of identifying CMS type.

B Does CMS Popularity Affect Exploitability?

Results from the Subsection 3.1 showed that the some of the most popular CMS platforms, notably WordPress and Joomla, are compromised disproportionately often. We now dig a bit deeper to see if there is a statistically robust connection between CMS popularity and compromise. Before inspecting the compromise rates directly, we first compare CMS popularity to the number of readily-available exploits targeting the CMS platform.

For this analysis, we considered many more CMSes than in other sections. We consider all 52 CMS platforms tracked in [10]. These additional CMSes all have very small market shares, and so not enough registered in our datasets to include in the other analysis. For each CMS we collected the following two indicators:
# Servers: We took the market share for each CMS from [10] as of January 1, 2013 and multiplied it by population of registered `.com` domains (106.2 million) and the estimated server response rate (85%) from [5].
# Exploits: The Exploit Database [27] is a search engine that curates working and proof-of-concept exploits from a variety of sources, including the popular penetration-testing tool Metasploit. We searched the Exploit Database for each CMS and recorded the number of hits as a measure of how "exploitable" each CMS is. We discarded any results not matching the searched-for CMS. We deem this to be a more accurate measure of attacker interest in and the "hackability" of a content management system than would be counting the vulnerabilities reported for a CMS. Unlikely many vulnerabilities, exploits provide directly actionable information to compromise machines.

We hypothesize that the number of exploits available for a CMS depends directly on the number of servers in use. Because both variables are highly skewed, we apply a log transformation to each. Here is the statement of the linear regression:

$$\lg(\text{\# Exploits}) = c_0 + c_1 \lg(\text{\# Servers}) + \varepsilon.$$

The regression yields the following results:

	coef.	95% conf. int.	Significance
Intercept	**-8.53**	(-3.37, -13.69)	$p = 0.002$
lg(# Servers)	**0.64**	(0.33, 0.95)	$p = 0.0001$
Model fit: $R^2 = 0.23$			

Indeed, this simple linear model has a reasonably good fit. While there is additional unexplained variation, this lends indirect support to **H2**. Due to the collinearity of these variables, we only use one of them (# Servers) in our regressions in this paper.