Backup File Artifacts
The Underrated Web Danger

TESTING AND EXPLOITING BFA WITH BFAC

By:
Mazin Ahmed
@mazen160
mazin AT mazinahmed DOT net

WHO AM I?

- **Mazin Ahmed**
- Freelancing Security Researcher at Bugcrowd, Inc.
- Security Contributor at ProtonMail
- Freelancing Information Security Specialist

- You can read more at https://mazinahmed.net/

WHO AM I?

- And I have contributed to the security of the following:

Table of Contents

- Introduction
- How BFA Occurs?
- Background
- Previous Researches
- Wrapping Ideas Together
- The Problem
- BFA Findings in Real-World.
- Introducing BFAC
- Using BFAC
- Mitigations
- Questions

What is Backup-File Artifacts?

Background
Code Editors

- Code editors makes a backup copy of the files that being edited.
- Mostly, it's being done in the same directory of the file.
- It's made using the same filename.
- It uses predictable patterns.

Background
Code Editors

Product Name	Backup Pattern (Assuming the filename is config.php)
Emacs	#config.php#
Nano	config.php.save
Vim (swap file)	config.php.swp
Vim (swap file)	config.php.swo
Vim, Gedit	config.php~

Background
Code Editors
The Problem

- Developers mostly forgets to check for Backup File Artifacts caused by code editors, and underestimate the impact of leaving similar artifacts.

- This would lead to the **disclosure** of the source code of the script that has been edited.

Background
Version Control Systems

- Version Control Systems (VCS) are systems made to manage changes on files, documents, computer programs, code, websites, etc…
- All Version Control Systems has a directory by design that contains data and/or source-code.
- The directory is made to have a backup of files, and to control the changes on the files within the project.
- By design, we already know directories names, the important files, and the directories' structure of source-code version controller systems.

Background
Version Control Systems

Version Control System	Pre-known Files and Directories
GIT	/.git *(Default Directory)*
GIT	/.gitignore
GIT	/.git/index
GIT	/.git/HEAD
GIT	/.git/refs
Mercurial	/.hg *(Default Directory)*
Mercurial	/.hg/requires
Bazaar	/.bzr/ *(Default Directory)*
Bazaar	/.bzr/README
Subversion	/.svn *(Default Directory)*
Subversion	/.svn/entries
Subversion	/.svnignore

Background
Version Control Systems
The Problem

- Companies and developers in many occasions clone or download the repository into production, and **FORGETS TO DELETE THE VCS DIRECTORIES.**
- This would lead to the **disclosure** of the source code of the company's application.

Background
Version Control Systems
The Problem

- The worst part: many websites don't disable directory listing, which makes exploiting that like a piece of cake.

Profit?

- downloading the VCS directory would be as simple as:

```
$ wget --mirror -np -I /[path_to_vcs_dir] http://example.com/[path_to_vcs_dir]
```

Background
Version Control Systems
The Problem

- In case the website disables directory listing, exploitation is still possible.

Background
Version Control Systems
The Problem

- **Exploitation of Missed VCS Folders on Web-Apps When Directory Listing is Disabled:-**

- Simply: Brute-Forcing the VCS structure.

Background
Version Control Systems
The Problem

- **Exploitation of Missed VCS Folders on Web-Apps When Directory Listing is Disabled:-**

- **There are tools that can do that too:**
  - GIT-Tools by Internetwache: A collection of scripts that can be used to retrieved GIT repositories from the web-app, even if directory listing is disabled. Written in Bash and Python.
  - DVCS-Ripper by @kost: A tool that can perform recover different repositories even if directory listing is disabled. Written in Perl.
  - DVCS-Pillage by Adam Baldwin: A tool that extracts and retrieves different DVCS repositories.

Background

Human-Based Missed Backup File Artifacts

- Developers usually do a backup before attempting to edit files.

- Example:-

Editing index.php

- Before editing, many developers do the following:

```
$cp index.php index.php.bak
```

Background

Human-Based Missed Backup File Artifacts

The Problem

- Developers in many cases forgets to remove the backup file.
- It may even be left on production.

Background

Human-Based Missed Backup File Artifacts

The Problem

Impact

- An unauthorized attacker can brute-forces through different patterns of human-based backing up, and check if the file exists and accessible to the public.
- If succeeded, the unauthorized attacker would be able to view the source-code of the script (Source-Code Disclosure).

Previous Researches

Pillaging DVCS Repos For Fun And Profit - Defcon 19 - Adam Baldwin – 2011

- The talk discusses different leakage of Distributed Version Control Systems, and how it can leak the website's source-code and data if it's incorrectly accessible by unauthorized users.
- A test on Alexa top 1M was done and showed that roughly 2,000 websites were vulnerable.
- A tool called **DVCS-Pillage** was released that helps retrieving repositories from websites that exposes them.

Previous Researches

1% of CMS-Powered Sites Expose Their Database Passwords - Feross Aboukhadijeh – 2011

- A research that shows how can human-based backup artifacts and backup artifacts made by code-editors leaks sensitive data on popular CMSs.
- Also, tested Alexa top 200,000 websites for basic BFAs on CMSs configuration files.
- Results showed a large number of high-profile websites had BFA left on production.
- Wrote a tool that can detect basic BFAs for CMSs configuration files.

Previous Researches

Don't publicly expose .git or how we downloaded your website's sourcecode – Internetwache - 2015

- A research done by Internetwache on GIT VCS, and how it can leak web-applications' source-code and data.
- Demonstrated different techniques for real-world exploitation of missed GIT repositories on production.
- Tested Alexa top 1M websites for web-applications that left GIT repositories on production.
- Published a set of scripts to automatically tries to retrieve files from GIT repositories.

Wrapping Ideas Together

- Backup-File Artifacts are caused by different ways, mostly, it's the admin/developer mistake to allow it accessible.
- Exploitation of Human-Based BFA and BFAs that is caused by code-editors can be done as the following:

Example:-

Saying that index.php.bak is discovered.

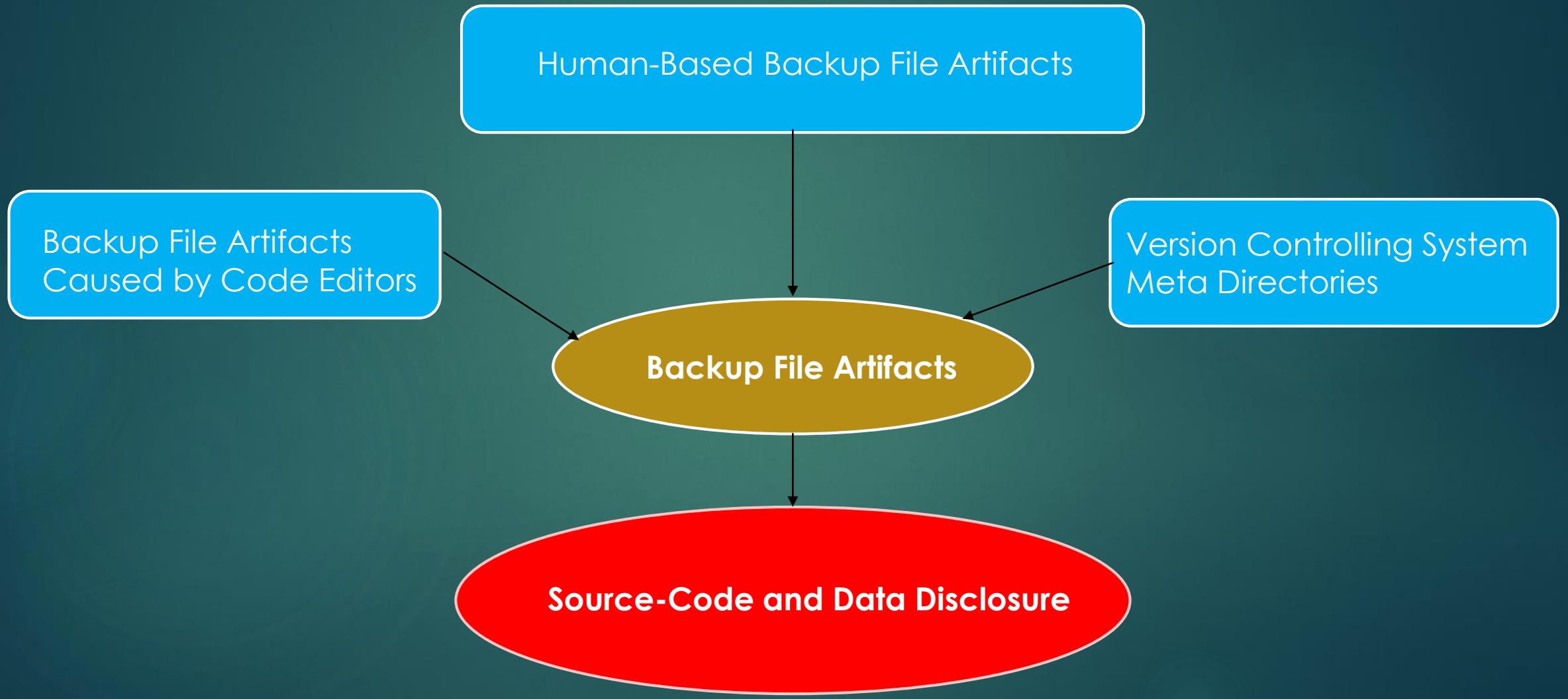
```
$ curl -s http://example.com/index.php
Normal Data
$
$ curl -s http://example.com/index.php.bak
<?php
echo "Normal Data\n";

////Credentials
// admin:password123
?>
$
```

Wrapping Ideas Together

- Having the meta directory of Control Version Systems accessible also leads to source-code and data disclosure.

Wrapping Ideas Together



The Problem

- Most Open-Source and Commercial Web Vulnerability Scanners **DO NOT PERFORM TESTING FOR BACKUP FILE ARTIFACTS.**

- **Examples of Tested Open-Source Scanners That Do Not Perform BFA Checks:-**
  - OWASP ZAP 2.4
  - Vega
  - W3AF – There are partial plugins that does basic testing, but it's disabled by default.
  - Nikto 2.4.6

The Problem

- There are very few current Scanners that performs Backup Artifacts. Testing. However, those rare scanners only test the very basic pattern of BFA.
- **Many Important patterns testing are MISSED!**

BFA Findings in Real-World

Paypal BFA That Lead to RCE - Ebraheem Hegazy

- Ebraheem found a script called upload.php on a Paypal-owned server.
- wrote script to test for BFA, and to check if the script exist on other subdomains.
- Found a BFA for upload.php, and transformed the Blackbox testing to whitebox testing.
- After source-code reviewing it, he found a bug that leads to RCE.

Profit: Got an RCE exploit on a Paypal server. Paypal fixed the issue, and he was awarded a nice bounty.

BFA Findings in Real-World

ISC.org BFA that Disclosed Database Credentials - Feross Aboukhadijeh

- During Feross's research, he tested ISC.org for BFA on Wordpress configuration files, wp-config.php.
- He found a BFA left on the main production site of ISC.org, containing database credentials.

Profit: The researcher had potentially valid credentials to ISC.org ("potentially" is said, as the researcher stated that he didn't use it gain unauthorized access). Feross reported it to ISC.org, and they have fixed it.

BFA Findings in Real-World

A Collection of BFAs - Mazin Ahmed

- Used BFAC to discover a number of BFA issues on web-applications, mostly for confidential clients.

- BFA Findings Leads To:
- Credentials leakage.
- Turning testing to whitebox.
- Finding additional security vulnerabilities.
- RCE.
- SQLI.
- XSS.
- Even more BFA issues.
- etc...

Introducing BFAC

Introducing BFAC

- **About the Project:**
- BFAC (Backup-File Artifacts Checker) is an automated tool that checks for backup artifacts that may discloses the web-application's source code.
- BFAC goal is to be an all-in-one tool for backup-file artifacts black-box testing.

Introducing BFAC

- **About the Project:**

- **Code Specifications:-**
  - Written in pure Python.
  - Compatible with both Python2.x and Python3.x.
  - Cross-Platform: Works on Linux, Windows, Mac, Android, and IOS.
  - Requires a slight amount of resources/modules.
  - API friendly.
  - Released under GNU GPLv3.0 License.

Introducing BFAC

- **About the Project:**

- **Features:-**

- Testing all common types of backup-file artifacts patterns, including human-based BFAs, and BFA that can occur via code-editors.
- Includes tests for common VCS artifacts, such as GIT, Subversion, Mercurial, and Bazaar.
- Smart detection techniques: Capable of detecting "Not Found", and valid pages using different tests.
- Stealthy Interaction with Web Servers.
- Easy to edit and customize based on needs; easy to add custom BFA patterns.
- Dynamic and generic; made to be not specified to test a specific environment or server.

Downloading BFAC

BFAC HomePage:

https://github.com/mazen160/bfac

Downloading and Installing BFAC

- Cloning **BFAC**

```
$ git clone https://github.com/mazen160/bfac.git
```

- Installing **BFAC** (for *NIX machines)

  As simple as:

```
$ python install.py confirm
```

Using BFAC

- Performs basic tests on the link, showing all attempts, and using random user-agents on each request.

```
$ bfac --url 'http://example.com/test.php' --verbose --random-agent
```

- Ignore 500 and 503 HTTP response codes when is detected as findings.

```
$ bfac --url 'http://example.com/test.php' --verbose  --exclude-status-codes 500,503
```

Using BFAC

- Default option to confirm the existence of the BFA is set to check both of the HTTP Status Codes and the Content Length (set to "both").

- Rely only on HTTP Status Codes for confirming the existence of the BFA.

```
$ bfac --url 'http://example.com/test.php' --verbose --verify-file-availability status_code
```

- Rely only on Content Length measures in order to confirm the existence of the BFA.

```
$ bfac --url 'http://example.com/test.php' --verbose --verify-file-availability content_length
```

Using BFAC

- Perform BFA testing for all current testing levels (Level 4 includes tests from all below levels).

```
$ bfac --url 'http://example.com/test.php' --verbose --level 4
```

- Print a clean output with only findings, separated by newlines. Suitable for APIs and integration with other tools.

```
$ bfac --url 'http://example.com/test.php' --no-text --level 4
```

Mitigation

Developer-Level:-

- **Awareness**
- Probably the only reliable solution.
- Developers' behavior regarding actions on production-level and publicly accessible applications are the main reason for this vulnerability type to occur.

Software-Level:-

- Access rules seems to be useful in blocking some of the patterns. However, it does not protect as needed.

Questions?

Mazin Ahmed
Twitter: @mazen160
Email: mazin AT mazinahmed DOT net
Website: https://mazinahmed.net
LinkedIn: https://linkedin.com/in/infosecmazinahmed