FlashPatch: Spreading Software Updates over Flash Drives in Under-connected Regions

Henry Corrigan-Gibbs
Stanford University
henrycg@stanford.edu

Jay Chen
NYU Abu Dhabi
jay.chen@nyu.edu

ABSTRACT

We present the design and implementation of FlashPatch, a system which *exploits* the ubiquity of USB drives—normally a vector for virus transmission—to facilitate the distribution of software updates. Our system spreads software updates to offline machines by having the updates "piggy-back" on the existing circulation of USB flash drives. We discuss the techncial and ethical issues involved in the deployment of such a system and we describe the results of preliminary field testing in rural Ghana.

Categories and Subject Descriptors

D.4.6 [**Software**]: Operating Systems—*Security and Protection*

General Terms

Human factors, Security

Keywords

anti-virus; development; security; software updates

1. INTRODUCTION

The proliferation of computer viruses in developing regions has had a variety of detrimental effects: viruses cause Internet café owners to lose business, they render users' data unrecoverable, and they inconvenience computer owners by consuming scarce system and network resources [2]. In areas with slow or expensive Internet connectivity, the common usage of USB flash drives for file sharing facilitates the spread of viruses [1]. At the same time, bandwidth limitations make it time-consuming and costly for users to keep their operating system, anti-virus software, and other applications up to date, so viruses can take advantage of known security vulnerabilities to infect unpatched machines.

In this work, we present the design and implementation of FlashPatch, a system which *exploits* the ubiquity of USB drives—normally a vector for virus transmission—to facilitate the distribution of software updates. Our system spreads software updates to offline machines by having the updates "piggy-back" on the existing circulation of USB flash drives. When a user plugs a flash drive into a computer running our software, the computer will (with the user's consent) copy a number of software updates onto the drive. When the user later connects that same drive to a different machine, which lacks those particular updates, the machine will download and install the updates from the USB drive. A crucial element of our system design is that we require minimal user intervention in the update process.

ACM DEV 4 December 6–7, 2013 Cape Town, South Africa
ACM 978-1-4503-2558-5/13/12.
http://dx.doi.org/10.1145/2537052.2537067.

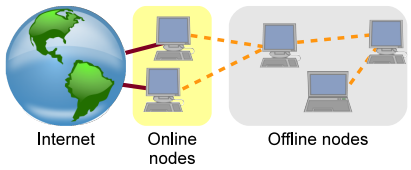

Figure 1: Updates flow over network links (solid lines) to online nodes, and over USB drives (dashed orange lines) to offline nodes.

To test the feasibility of such a system, we have implemented a prototype virus scanner based on ClamAV which uses FlashPatch to update the scanner's virus definition database. We have deployed our prototype antivirus at six Internet cafés, a graphic design shop, and a high school in Hohoe, Ghana, and we are expanding the experimental deployment to 100 machines in the upcoming weeks.

Although FlashPatch bears some resemblance to an "altruistic virus," we emphasize that our software *does not spread itself* and *is not a virus*: our system copies only data (not executables) to the USB drives, and only machines running our prototype software receive the updates. Even so, we had to address a number of ethical issues in deploying the software.

2. SYSTEM ARCHITECTURE

The goal of FlashPatch is to provide timely software updates to computers with expensive and unreliable Internet connectivity. We have also implemented a system for recovering log files from patched offline machines to help gauge the effectiveness of our trial deployment of FlashPatch.

Distributing Updates. As depicted in Figure 1, a deployment of FlashPatch involves three classes of nodes:

- a single **vendor node**, which publishes updates for the piece of software in question,

- **online nodes**, which are able to download software updates from the vendor, and
- **offline nodes**, which have no network connectivity and can communicate with the online nodes only via USB drives.

When using FlashPatch to upgrade many different pieces of software, there can be many different vendors operating at once (one per application), but for simplicity we consider the case of a single vendor updating a single software application.

Whenever the vendor publishes (e.g., to a public website or FTP server) an updated version of its software, the vendor assigns a version number to the new version and digitally signs the entire update package with its private signing key. Online nodes periodically check the vendor's servers for updates, download new versions of the software when available, verify the digital signature on the newly downloaded package, and then install it locally.

Once an online node has downloaded the updated version of the software, it will make this new version of the software available to offline nodes via our USB file transfer mechanism. Whenever a user later inserts a USB drive into a node (either online or offline), the node will check to see if the USB drive has a newer signed version of the software than the node does by comparing the version numbers. If the USB drive's version is newer, the machine will install the updated software locally. If the machine's version of the software is newer, the machine will copy the software to the USB drive.

Whenever a user plugs a USB drive into an up-to-date online machine and then into an out-of-date offline machine, the out-of-date machine will be able to retrieve the latest version of the software from the USB drive. In this way, users who move USB drives amongst collections of online and offline machines (e.g., between online Internet cafés and an offline school computer lab) inadvertently spread software updates. As long as there is only a single vendor who publishes updates for a particular piece of software, nodes will never encounter inconsistent updates.

Collecting Logs. Measuring the effectiveness of FlashPatch required us to build a system for collecting log files from machines that do not have Internet connectivity. To collect log files from offline machines, we built a system for nodes to share their log files, in addition to software updates, over USB drives. Nodes use a simple distributed shortest-path algorithm to determine which USB drives are closer or farther (in terms of number of hops) from our our server. After encrypting their log files with our public key, nodes use USB drives to forward the encrypted log files to nodes that are closer to our server. Once the log files reach an online node, the online node uploads these logs to our server.

3. EVALUATION

To evaluate the feasibility and practicality of FlashPatch in a real system, we have implemented a virus scanner which uses FlashPatch to distribute virus database definition updates. Our virus scanner is written in C# and it uses the open-source ClamAV anti-virus as the underlying scanning engine. Our scanner attempts to download the latest ClamAV virus definition database every week over the network and it can also receive database updates via the FlashPatch network of USB drives.

For our first-round deployment, we installed our prototype virus scanner on 20 computers in six Internet cafés, one school, and one graphic design shop in Hohoe, Ghana. Hohoe is a small town in Ghana's Volta Region whose main source of Internet access is intermittent 3G service provided by Ghana's major mobile phone operators. Our prototype virus scanner automatically scans all files on any USB drive inserted into the machine on which the software is installed. To evaluate FlashPatch on both online and offline computers, we are deploying the software on 80 additional computers in Hohoe, many of which have no Internet connectivity.

4. ETHICAL CONSIDERATIONS

Consent. Since human subjects are involved in our field evaluation of FlashPatch, we sought and obtained IRB approval for our study methodology and we obtained written consent from the owner of each computer running the software. We limit the amount of space that our software can consume to 20% of the size on the USB drive, and we do not copy anything onto the drive if doing so would fill up the drive. In practice, the prototype uses around 35 MB of space on a flash drive. Before copying data files to the flash drive, we copy a `README` file to the drive, which explains how to remove the update files from the drive and also how to uninstall the prototype completely from the user's machine.

Privacy. Our prototype software does not record file names, IP address, MAC addresses, or other information which would make it easy to identify a computer owner. We take the extra precaution of having our prototype software encrypt its log files using our research group's public key before the software sends the logs over the Internet or over our network of USB drives.

5. RELATED WORK

Prior work has demonstrated that computer viruses are an economic and social problem in developing regions [2, 3]. Paik [4] responded to the virus problem by describing a system for gathering detailed information about computer virus prevalence in developing regions. Others have considered the idea of spreading updates or patches via high-latency links [5]. Our application of these techniques to the context of software updates in developing regions is novel, to the best of our knowledge.

6. REFERENCES

[1] Yahel Ben-David et al. Computing security in the developing world: A case for multidisciplinary research. In *5th NSDR*, pages 39–44, 2011.

[2] Prasanta Bhattacharya and William Thies. Computer viruses in urban Indian telecenters: Characterizing an unsolved problem. In *5th NSDR*, pages 45–50, 2011.

[3] David L. Johnson et al. Traffic characterization and internet usage in rural Africa. In *20th WWW Companion*, pages 493–502, 2011.

[4] Michael Paik. Gotta catch'em all! Innoculous: enabling epidemiology of computer viruses in the developing world. In *5th NSDR*, pages 51–56, 2011.

[5] Ian H. Witten et al. Liveware: A new approach to sharing data in social networks. *International journal of man-machine studies*, 34(3):337–348, 1991.