Poster: Games Without Frontiers: Investigating Video Games as a Covert Channel

Bridger Hahn Rishab Nithyanand Phillipa Gill Rob Johnson
Department of Computer Science
Stony Brook University

Censorship circumvention tools face an arms race as they work to evade increasingly motivated censors. Tools which have distinctive features can be detected and blocked by censors (*e.g.,* Tor is actively targeted by censors around the world). As a result, there is increasing interest in disguising censorship circumvention traffic as benign protocols. SkypeMorph [1] and StegoTorus [2] are two pluggable transports [3] for Tor which aim to mask Tor traffic as Skype traffic, and a combination of Skype, HTTP and Ventrilo, respectively.

While these pluggable transports are able to capture features of the traffic they aim to imitate (*e.g.,* inter-packet timings, packet size distributions), Houmansadr *et al.* [4] point out that imitating a protocol is not enough to evade detection – i.e., if circumvention endpoints do not run the application they aim to hide within, they are vulnerable to censors who may perform active probing to observe that the endpoint does not actually implement the full functionality of the mimicked application. As such, Houmansadr *et al.*advocate that circumvention schemes should actually run the application they are using as cover, rather than mimicking properties of the cover traffic.

Towards this goal, FreeWave uses the Skype application as a modem to transmit IP data between two endpoints [5]. However, Geddes *et al.*demonstrate that even running the cover application is not enough to avoid detection by censors [6] – i.e., approaches like FreeWave may be detected via discrepancies between application behavior when it is acting as a covert channel, vs. regular application operation. They classify these discrepancies into three categories: (1) architectural mismatches between communication patterns of the application when it is acting as a covert channel vs. regular operation, (2) channel mismatches between reliability requirements of the application and the covert traffic and (3) content mismatches where the packet contents of the application differ because of the covert traffic being sent in place of regular application traffic.

Thus, circumventors are in a race to design new cover channels that are difficult to detect, while censors search for low-cost mechanisms to distinguish legitimate traffic from covert channels. In this poster, we argue that multi-player games as a covert channel presents a solution that can tip the scales heavily in the favor of circumvention tool developers. The large number of games and common features of games within a given genre facilitate adaptation of existing game-based covert channels to new games. Further, security features of games and their ability to leverage either a central server or peer-to-peer architecture significantly raise the bar for censors that aim to detect and block covert channels. Finally, as we will demonstrate with our prototype (named Castle), game-based covert channels can be designed such that they match the three properties highlighted by Geddes *et al.*.

The Castle approach: In order to create a covert channel mechanism that is general to the majority of games in the real-time strategy genre, Castle exploits two key properties.

- Most real-time strategy games share a common set of actions. Specifically, the ability to select buildings and assign a location where units created/trained in a building should go. This location is called a "rally point", and we denote the command of setting the rally point for units created in a given building by `SET-RALLY-POINT`. Games also provide the ability to move a selected unit to a given location (denoted by the `MOVE` command). Thus, any encoding that translates data into a combination of unit/building selections and these primitives will be general across most games in this class.
- Most real-time strategy games provide a replay option which saves every players' moves to disk (for later playback). Therefore, all in-game commands are written to disk where they can be read and decoded in real-time, with little effort.

Castle consists of two main components to send and receive data. These are illustrated in Figure 1. Sending is done by encoding data into game commands and then executing them within the game using desktop automation. The receiving process monitors the log of game commands and decodes this list to retrieve data sent via the system. Figure 2 overviews how the Castle system could be used to relay data from outside of a censored region to a client within the region. The client first installs Castle (*e.g.,* as a browser extension). The Castle client then initiates a game through a game lobby (or directly with the client outside of the censoring region). The client in the censoring region can then encode and send data (*e.g.,* Web requests) as game

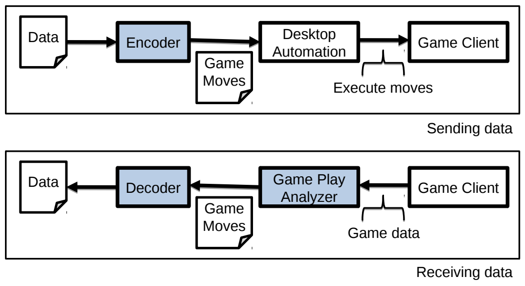

Fig. 1: Overview of data flow for sending an receiving in Castle. Shaded components are implemented as part of Castle while the others use existing off-the-shelf software.

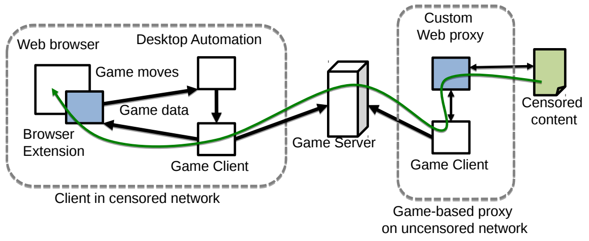

Fig. 2: Overview of how Castle can be used as a proxy for clients within censoring countries.

moves that can be decoded by the client outside of the censoring region. The game client outside of the censoring region can then act as a proxy to retrieve censored content and send it via Castle to the client in the censoring region.

We demonstrate the feasibility of our approach by prototyping on two different games (one open-source, and one extremely popular closed-source real-time strategy game) with minimal development overhead ($<$ 9 hours for development and deployment) and show its resilience to a network adversary. Specifically, our results show that Castle is:

- *Extensible:* Castle's strength comes from it's pluggable architecture which allows it to be easily ported to any number of games. As an example, it took a bright undergrad less than 6 hours to complete a basic port of Castle over a very popular closed-source real-time strategy game. Due to the availability of game specific hacks and reverse engineering guides in popular gaming forums, completing game specific enhancements in order to improve the data rate of Castle, required only an additional 3 hours.
- *Usable:* Even without any game specific modifications, Castle is able to provide throughput sufficient for transfer of textual data. Additional (game specific) enhancements make it suitable for use as a web proxy. Figure 3 demonstrates the throughput of Castle (under several configurations) over the closed-source RTS-game, with and without any game specific enhancements.

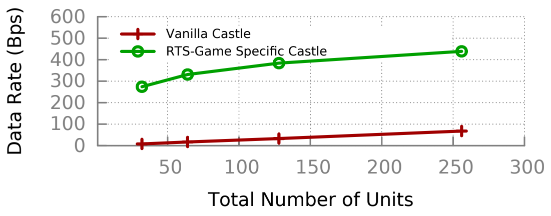

Fig. 3: Throughput of Castle (with and without enhancements) implemented over closed-source RTS-Game

- *Secure:* Castle is resistant to attacks such as IP/port filtering and deep-packet inspection since it actually executes the game application. Further, we find that Castle is also resistant to attacks that depend on analysis of flow level features such as packet sizes and inter-packet times. Since the traffic generated by a standard multi-player game is strongly dependent on many parameters (e.g., player personality, strategies employed, scenario type, map, number of players, etc.), flow level features may vary widely between game instances. Castle (under all configurations) generated traffic was well within the variance seen in real human vs. human game traffic, for both – inter-packet times and packet sizes.

Acknowledgements

This material is based in part upon work supported by the National Science Foundation under Grant Number CNS - 1350720. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 97–108, New York, NY, USA, 2012. ACM.

[2] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. StegoTorus: A Camouflage Proxy for the Tor Anonymity System. In *Proceedings of the 19th ACM conference on Computer and Communications Security (CCS 2012)*, October 2012.

[3] Tor Project: Pluggable Transport. https://www.torproject.org/docs/pluggable-transports.html.en. Accessed: 2015-02-22.

[4] A. Houmansadr, C. Brubaker, and V. Shmatikov. The Parrot Is Dead: Observing Unobservable Network Communications. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 65–79, May 2013.

[5] Amir Houmansadr, Thomas Riedl, Nikita Borisov, and Andrew Singer. I Want my Voice to be Heard: IP over Voice-over-IP for Unobservable Censorship Circumvention. In *Proceedings of the Network and Distributed System Security Symposium - NDSS'13*. Internet Society, February 2013.

[6] John Geddes, Max Schuchard, and Nicholas Hopper. Cover Your ACKs: Pitfalls of Covert Channel Censorship Circumvention. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &; Communications Security*, CCS '13, pages 361–372, New York, NY, USA, 2013. ACM.