Through Glass Transfer

Welcome to the Through Glass Transfer information site.

Through Glass Transfer is the label that has been given to a suite of tools that demonstrate the unauthorised infiltration and exfiltration of code and data through secure networks, particularly to highlight the vulnerability exposed through *offshoring* (human resources off-shore, systems and data on-shore) using commonly deployed enterprise security architectures. In that regard, the information presented on this site does not have a CVE-ID, because it does not reveal a new vulnerability in any specific application or technology, but rather a flaw in the end to end architecture.

What are the tools?

ThruGlassXfer (TGXf) is a protocol and application designed to let people download files through a screen. It is a proof-of-concept created to demonstrate a long standing and poorly understood vulnerability in enterprise security.

The TGXf protocol is a transport protocol that allows one way transfer of data, between two peers, typically in the form of binary data bundles (files), though streams are possible. The protocol supports high latency, interrupted transfers and error detection. The TGXf transport protocol utilises QR codes as frames in an optical packet network, though it will work on any packet protocol. Quick Response Codes were chosen for the proof-of-concept because they were designed to be recognised and decoded rapidly by machine.

ThruGlassXfer (TGXf) is also an application. It has been built as a proof-of-concept to demonstrate the effectiveness of the TGXf protocol. The ThruGlassXfer (TGXf) app has been made available in both the Apple and Google app stores and Linux binaries for the transmit (server) side have been made available below.

ThruKeyboardXfer (TKXf) is a protocol and application designed to let people upload files through a keyboard. It is a proof-of-concept created primarily to support TGXf (by showing that it is trivial to upload the encoder to sensitive compute/server platforms), but TKXf also serves to demonstrate the same poorly understood vulnerability in enterprise security (though inbound, rather than outbound, from the enterprise perspective).

Like TGXF, the TKXf protocol is a transport protocol that allows one way transfer of data between two peers. Unlike TGXf however, the TKXf protocol only supports a basic file transfer capability (a binary file encoded in a shell script for the target system) with a simple control structure - i.e. no recoverability from interrupted transfers and no error detection. The TKXf protocol utilises the USB HID keyboard packet as frames in its packet network, though it will also work on any packet protocol. TKXf requires a hardware device to interface as the USB HID keyboard, the Arduino hobby platform was selected for its ease of use.

ThruKeyboardXfer (TKXf) is also an application. It has been built as a proof-of-concept to demonstrate the effectiveness of the TKXf protocol. The ThruKeyboardXfer (TKXf) software has been made available for the Arduino and Linux platforms - please see below.

ThruConsoleXfer (TCXf) is an application that combines both TKXf and TGXf to create an asynchronous network socket that passes from the most sensitive compute/server platforms through the screen and keyboard of end user computing devices owned by the enterprise, and terminates on an attacker controlled device.

By the nature of the component protocols, TCXf remains undetected and unmitigated by existing enterprise security architectures.

clientlessTGXf (clTGXf)

clientlessTGXf is an alternate implementation of TGXf. It is designed to demonstrate a screen based transfer that doesn't require a binary client on server targets (the DCE), where the stream can be captured and processed without a camera, and also shows that the QR code can be replaced with even ASCII text.

highspeedTGXf (hsTGXf)

Like clientlessTGXf, highspeedTGXf is an alternate implementation of TGXf. However, hsTGXf is designed to demonstrate a high performance (>1Mbps) screen based transfer that doesn't require a binary client on the target (the DCE), where the stream can be captured and processed without a camera, and also shows that the QR code can be replaced with raw pixel data.

Want to see the tools run?

Watch TGXf demonstrations:

- TGXf Demo - Open Letter PDF, ANSI (Terminal) Version 1 at 8 FPS (http://youtu.be/ZrMN54Rooec) for an example of a TGXf upload from an ANSI terminal/shell such that a camera on the other side of the screen could download it, and;
- ThruGlassXfer Open Letter (PDF) - TGXf VER8 FPS5 GD (http://youtu.be/IXlYDYjqFLU) for an example of a TGXf upload from a graphical interface, also able to be downloaded from the other side of the screen.

Watch TKXf demonstrations:

- TKXf Demo - Keyboard upload of virus to hardened Windows platform (http://youtu.be/2Szza7dQZsY) for an example of a TKXf-like upload of the EICAR test virus to a Windows Thin Client, and;
- TKXf Demo - Keyboard upload of payload via Windows to Linux (http://youtu.be/QmROf-Tx92E) for an example of a TKXf upload to a Linux server via a Windows Thin Client.

Watch a TCXf demonstration:

- Attacker exfiltration from Linux via socket over PuTTY/XPe/HP Thin Client (http://youtu.be/sMHx5VDpFjQ) for an example of TCXf being used to download data to an attacker device, using netcat.

Watch clientlessTGXf demonstrations:

- clientlessTGXf upload from BASH (http://youtu.be/5ZhBwX4o4E0) for an example of a clientlessTGXf upload using a text-based datagram protocol (ASCII rendered ones and zeros) from a terminal/shell such that a HDMI capture device can record it to MP4, and;
- clientlessTGXf download (decode) in Linux (http://youtu.be/D8rGguS2a-s) for an example of an HDMI captured clientlessTGXf upload being restored from an MP4 capture, frame by frame.

Watch highspeedTGXf demonstrations:

- highspeedTGXf 12.1Mbps upload from browser (http://youtu.be/ejN4haM6Iww) for a demonstration of a high speed TGXf upload using a HTML5 canvas and HTML5 file upload in a single combined javascript and HTML page, run from FireFox in full-screen mode.
- 12.1Mbps highspeedTGXf download decode in Linux

(http://youtu.be/ckpjbU7GRcg) for an example of an HDMI captured highspeedTGXf upload being restored from a MOV converted AVI capture, frame by frame.

How is this a risk, technically?

ThruGlassXfer (TGXf)

Anywhere that data can be visually rendered, technical risk will exist.

TGXf encodes binary data into packets that can be displayed on the screen of one computer and then captured (via camera) on another, where they are decoded and the data is stored on disk. By doing this, TGXf turns any display surface into an egress (outbound) binary data transfer interface.

ThruKeyboardXfer (TKXf)

Anywhere that data can be entered, technical risk will exist.

As a part of the development process for TKXf, a TKXf-like transfer was created by encoding a binary payload and storing it on a *programmable keyboard* (the Arduino platform with USB HID keyboard capability) which sends this payload as a script (currently BASH or PERL) that is typed into the target system. When run, this script exports the original binary payload. See the image: *ThruKeyboardXfer (TKXf) programmable keyboard*.

TKXf encodes binary data into packets that can be sent out of the serial port of one computer and entered via keyboard on another, where they are decoded and the data is stored on disk. By doing this, TKXf turns any keyboard interface into an ingress (inbound) binary data transfer interface. The TKXf-like *programmable keyboard* device is transformed into a TKXf *keyboard stuffer* by adding a USB serial interface for the attacker's computer. This change allows the attacker to send a continuous steam of data (rather than a single pre-programmed file). See the image: *ThruKeyboardXfer (TKXf) keyboard stuffer*.

ThruGlassXfer (TGXf) and ThruKeyboardXfer (TKXf)

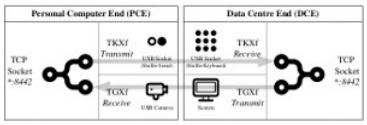
Given that in each case (TGXf and TKXf) these are simply a new type of file transfer protocol, the proof-of-concept software does not create new risk when being used to transfer files between consenting parties. However, there are common use cases where one party would not want to consent, and one of those cases is the remote worker (particularly an *offshore* partner).

When a remote worker (VPN user) with read access to files on a company's computers, uses TGXf to download those files, or TKXf to upload malicious software, that user circumvents the company's security controls. Controls such as packet filtering (firewalls), harmful code detection/containment (AV), intrusion detection or prevention systems (IDS/IPS), unauthorised change detection and data leakage prevention (DLP) all fail to mitigate these protocols. Defence in depth strategies that layer multiple session termination points (such as jump hosts, virtual desktops and hosted application environments) fail to mitigate both TKXf and TGXf, as in both cases the screen and keyboard is abstracted from zone to zone (at the application layer), and in the case of TGXf the protocol is capable of leveraging both terminal and graphical interfaces.

The included screen shot is of a graphical desktop with terminal software displaying a TGXf packet rendered via ANSI terminal sequences (which would also work through your browser while viewing this page). See the image: *ThruGlassXfer (TGXf) via xterm using ANSI codes*.

ThruConsoleXfer (TCXf)

By removing the file-based transfer aspects of each protocol (TGXf and TKXf) and then combining both protocols into a single application, ThruConsoleXfer, TCXf is now able to create an asynchronous TCP socket that routes the egress data flow via the screen and camera (TGXf) and the ingress data flow via the keyboard (TKXf). See the image: *ThruConsoleXfer (TCXf) application architecture*.



(TCXf-application-architecture.png)

ThruConsoleXfer (TCXf) application architecture

When TCXf is deployed, the Data Center End (DCE) is run on the application server, routed via the screen and keyboard of the end user computing platform that the enterprise gave to the attacker (in this case, the *offshore* partner), and the Personal Computer End (PCE) is run on the attacker's own computer (outside of enterprise control). In this way, TCXf transparently traverses the entire enterprise security architecture unchecked and unmitigated. See the image: *ThruConsoleXfer (TCXf) enterprise impact*.

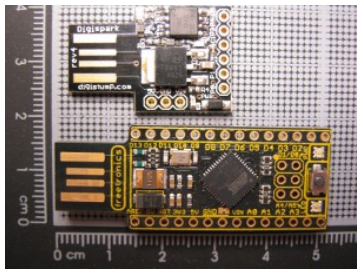
For more technical detail, please see the TCXf White Paper (TCXf%20White%20Paper%20-%20Midnight%20Code%20-%20v1.1.pdf).

clientlessTGXf (clTGXf)

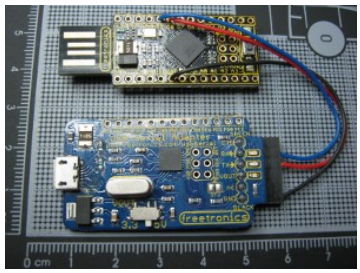
clientlessTGXf provides the same file transfer functionality as TGXf but via an alternate implementation.

The receiving device for clientlessTGXf is a HDMI capture device that stores the video stream as a video file (MP4). The video file is then processed frame by frame for TGXf packets that are decoded and written to disk.

Technical Diagrams


(TKXf-programmable-keyboard.png)

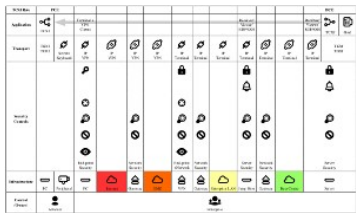
ThruKeyboardXfer (TKXf) programmable keyboard


(TKXf-keyboard-stuffer.png)

ThruKeyboardXfer (TKXf) keyboard stuffer


(xterm.png)

ThruGlassXfer (TGXf) via xterm using ANSI codes


(TCXf-enterprise-impact.png)

ThruConsoleXfer (TCXf) enterprise impact

The risk to heed from clientlessTGXf is that regardless of the proof-of-concept implementations on this site, in practice, data could be encoded in the display in any format - graphically as pixel data or images and logos, or in text as letters, words or phrases. I.e. don't dwell exclusively on application signatures or particular data structures (like QR codes).

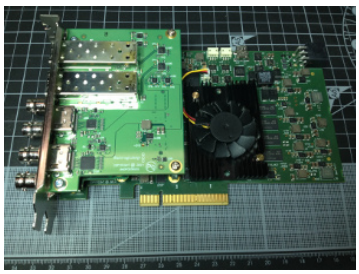
highspeedTGXf (hsTGXf)

highspeedTGXf provides the same file transfer functionality as TGXf but via an alternate implementation.

Like clientlessTGXf, the receiving device for highspeedTGXf is a HDMI capture device that stores the video stream as a video file (MP4/AVI/MOV). The video file is then processed frame by frame for TGXf packets that are decoded and written to disk.

The risk to heed from highspeedTGXf is that regardless of the proof-of-concept implementations on this site, in practice, data could be encoded in the display in any format - in this case, graphically as pixel data or images and logos. I.e. don't dwell exclusively on application signatures or particular data structures (like QR codes). Furthermore, the screen represents a significant data theft opportunity with hsTGXf capable of exfiltrating gigabytes of data per user session per day.

(clientlessTGXf-HDMI-capture.jpg)
clientlessTGXf (clTGXf) HDMI capture device



(highspeedTGXf-HDMI-capture2.jpg)
highspeedTGXf (hsTGXf) HDMI capture device

What is the business risk?

Your legal liability may have changed with the release of this software, and it stems from an artificial distinction drawn between being able to *read* data from a screen and being able to *download* that data.

Some people regard a file displayed to a screen as being within control of the computer system that governs that file, which then differs from a file that is downloaded to your computer such that the source computer system no longer has control over that file. In information security terms, once information has been displayed on a screen the source computer system has lost control of it (it has effectively been *uploaded* to the room). All that remains to *download* that data is acquiring it from the screen and writing it to disk on another computer. A human can perform this task manually today - remembering a handful of numbers and letters and then typing that information in to a computer at home. TGXf automates that download capability, such that information displayed in a certain structure on one computer can be readily acquired by another.

Unfortunately this distinction has been codified into our legal and regulatory frameworks.

For an Australian example, please refer to the Australian Privacy Principles guidelines (Privacy Act 1988) - version 1.0, February 2014 (http://www.oaic.gov.au/images/documents/privacy/applying-privacy-law/app-guidelines/APP-guidelines-combined-set-v1.pdf): An Australian organisation (APP entity) that *holds* personal information [6.6], and makes that personal information available for *use* [6.8] but not *disclosure* [6.11(b)] to an *overseas recipient* [8.5] is accountable [8.1] for ensuring that the overseas recipient does not breach the Australian Privacy Principles [8.2]. Personal information is disclosed when the Australian organisation *releases the subsequent handling of the information from its effective control* [8.8], which includes *disclosure* through *unauthorised access*, *if it did not take reasonable steps to protect the personal information from unauthorised access* [8.10]. The *reasonable steps* include *implementing strategies to manage*, amongst other things, *ICT Security*, *data breaches* and *regular monitoring and review* [11.8].

Of the *six security considerations* [11.10] set out in the Australian National Privacy Principles, TGXf would appear to directly enable:

- *interference* when an overseas recipient instigates change to a computer system that *leads to exposure of personal information* [11.13], and;
- *unauthorised disclosure* when an overseas recipient *releases the subsequent handling of that personal information from [the Australian organisation's] effective control* [11.18].

This issue is not limited to Australia. Any modern legal or regulatory frameworks that rely upon the same distinction for *use* and *disclosure* to govern personal or private information (PI), personally identifiable information (PII) or personal health information (PHI) will have some business impact from the release of this software. For example HIPAA (see HIPAA Administrative Simplification, Regulation Text, 45 CFR Parts 160, 162, and 164 (http://www.hhs.gov/ocr/privacy/hipaa/administrative/combined/hipaa-simplification-201303.pdf) for the definitions of *disclosure* on page 12 and *use* on page 17).

The following Risk Calculator has been designed to help you calculate your relative exposure based on cost case studies from a number of jurisdictions, the method of the theft (from remembering one word per day to high speed transfer through the screen), the number of attackers and the average size of a PI/PII/PHI record in your organisation.

Risk Calculator

Cost per record : US$2/record

Exfiltration method : Remember (1 word/day)

Attackers : 1 person

Record size : 2KB/record

Impact Forecast

1
Records per Quarter

2.00
US$ per Quarter

Elapsed time:	0 business days, 0 hours
Exfiltrated data:	0.00B (bytes)
Exfiltrated records:	0
Cost:	US$0.00

Cost Case Studies

US$319/record (US$15m for 47k records, 2015)
How much do data breaches cost big companies? (http://fortune.com/2015/03/27/how-much-do-data-breaches-actually companies-shockingly-little/)

US$89/record (US$25m for 280k records, 2015)
Feds wallop with $25M fine over stolen data (http://www.cnet.com/n fcc-25-million-for-customer-data-breach/)

UK£50/record (UK£2.3m for 46k records, 2010)
UK insurer hit with biggest ever data loss fine (http://www.theregister.co.uk/2010/08/24/data_loss_fine/)

Cost Case Studies

UK£18/record (UK£3.2m for 182k records, 2009)
Fined £3.2m for losing customers details
(http://www.telegraph.co.uk/finance/newsbysector/banksandfinance
fined-3.2m-for-losing-customers-details.html)

US$2/record (US$252m for 110m records, 2015)
How much do data breaches cost big companies?
(http://fortune.com/2015/03/27/how-much-do-data-breaches-actuall
companies-shockingly-little/)

AU$1/record (AU$10.2k for 15k records, 2014)
Fined after breaching privacy of 15,775 customers
(http://www.abc.net.au/news/2014-03-11/telstra-breaches-privacy-o
customers/5312256)

If you are looking to contextualise or validate the above risk modeling then please also see the Data Breach Calculator (http://databreachcalculator.com/).

What does it look like?

A number of demonstrations can be found as links in the *About TGXf* (top) section of this site. However, if you only see three demonstrations, then the following are highly recommended.

ThruGlassXfer (TGXf)

The following video demonstrates a Through Glass Transfer with the TGXf protocol only. Shown is a common laptop with Windows 7 and Firefox, playing the below Open Letter PDF from Youtube (at version 8 with 5 FPS), in HD mode (http://www.labnol.org/internet/youtube-video-hdplayback-quality/13873/). An Android smart phone (HTC One XL) with the ThruGlassXfer app on it is held facing the laptop screen so that the phone's primary camera can clearly read the displayed QR codes. For this demonstration the phone has been put in *flight* mode, ensuring that it has no data network access.

Video: Android smart-phone in *flight* mode, downloading a PDF from Youtube via a Laptop screen

Observe the yellow progress bar updating on the phone as data is received and stored. At the end of the video the popup advises that the file has been completed successfully. The background noise is from a rack of computers next to me. The video is dark because it was filmed at night in an unlit room to make both screens clear in the final video. And, yes, there is a giant crack running through the face of that phone.

ThruConsoleXfer (TCXf)

The following video is a demonstration of a TCXf infiltration/exfiltration in multi-hop security architectures.

On the left of the scene, with the red tape markers, is the attacker PC. The attacker PC is running a 64bit Linux Mint live CD and is not connected to any network. It has a USB Camera (Microsoft LifeCam, seen here sitting on two small Pringles cans) connected to it via USB, as well as a Midnight Code "Keyboard Stuffer" - a USB serial device (Freetronics USB Serial Adapter), that uses TTL to output to a USB HID keyboard device (Freetronics Leostick).

The defender (enterprise) devices are denoted with the green/yellow tape markers. In the middle of the scene is an enterprise end user device; a HP Thin Client device with a hardened Windows XPe (embedded) Operating System on it. Two instances of the PuTTY (SSH client) have been launched from the Windows XPe system to a 64bit Linux PC (scene right). The Linux PC on the right represents the enterprise application server, where the user has access to sensitive assets for use, but not disclosure (can work with files, but not download them).

The attacker's camera can see the end user device screen (TGXf), while the attacker's keyboard stuffer appears on the end user device as a USB HID keyboard (TKXf). In this configuration, the attacker can establish a TCXf (ThruConsoleXfer) where the PCE (Personal Computer End) is on the attacker's device (outside of enterprise control) and the DCE (Data Center End) is on the application server (at the heart of the enterprise). The result is a full-duplex TCP socket (8442/TCP) on the attacker's PC that connects to matching full-duplex TCP socket on the application server (8442/TCP) via the screen and keybard (which is abstracted within the enterprise via the SSH session).

Video: TCXf Demo - IP networking over Screen and Keyboard!

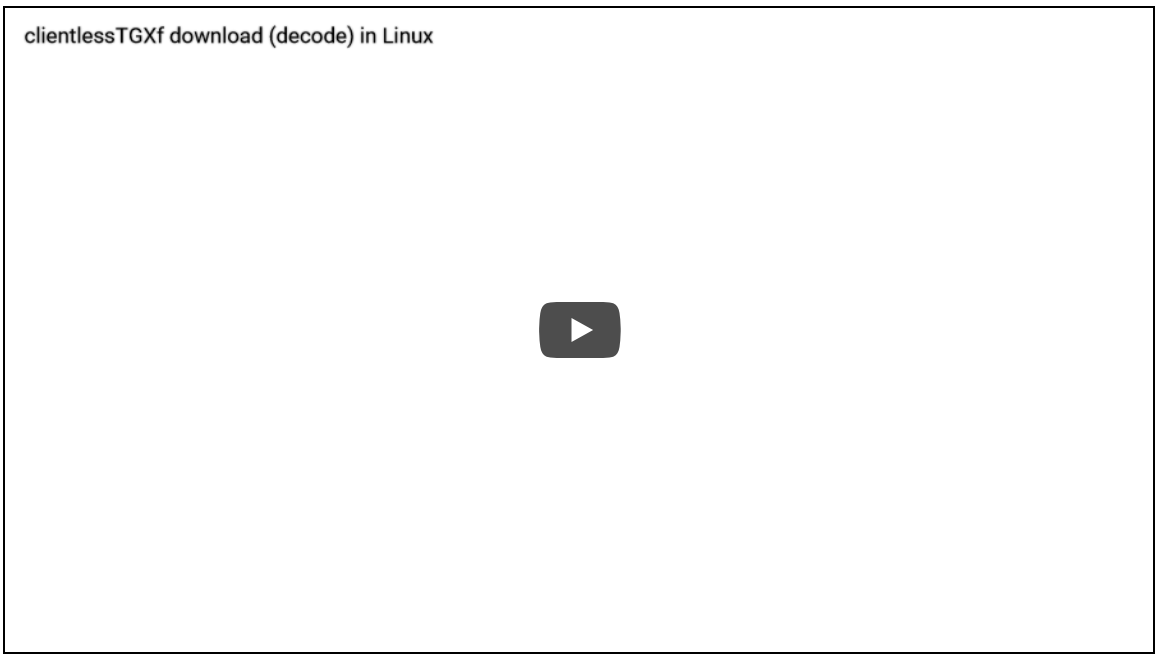
Apologies for the auto-focus on this video. If you want to know what is happening at each change in the scene, then please see the complete time-line that can be found in the description of the video on Youtube.

This demonstration is designed to show the full power of the TCXf solution. Rather than infiltrating or exfiltrating individual files, this TCXf connection has been used to connect a PPP peer on the attackers PC with a PPP peer on the application server at the heart of the enterprise. It is important to note that the user requires privilege on the application server in order to create a network interface (i.e. run the ppp daemon), but that no other aspect of this demonstration requires privileged access. Once the attacker has established the PPP link he runs an SSH session directly from the attacker PC to the application server on top of the IP network that now runs over the screen and keyboard on the Windows XPe and HP Thin Client end user device.

This access will circumvent all of the controls deployed in typical enterprise security architectures.

clientlessTGXf (clTGXf)

The following video demonstrates a Through Glass Transfer via the clientlessTGXf protocol.



Video: clientlessTGXf download (decode) in Linux

A file has been encoded in the clientless version of the Through Glass Transfer protocol (TGXf). Here the HDMI intercepted video (saved as an MP4 file) is being decoded by the receiving tool. The video is being played back frame by frame, with analysis of the screen for characters of a certain size. Once characters are located a randomly coloured box is drawn around it and it is handed off for character recognition. The recogniser has been trained on ones (1) and zeros (0), which are stored in a linked list by their location. After the frame has been fully recognised it is handed off to the TGXf protocol code which looks for 16 bits (1's and 0's) in the same row on the screen. Each of these TGXf rows is then processed as a sequence (datagram) in the layer 4 transport protocol. The counter helps the protocol to detect and reject errors (such as tearing from mismatched screen draws).

This is a proof of concept that shows data can be downloaded in binary formats through terminal/shell sessions via the screen, without the need for a client on the server or QR codes. The upload software is 300 bytes of BASH script, while the decoder is C++ source released under the GPL, which should port well to other platforms, including Windows.

highspeedTGXf (hsTGXf)

The following video demonstrates a Through Glass Transfer via the highspeedTGXf protocol.

Video: 12.1Mbps highspeedTGXf transfer downloaded (decoded) in Linux

The Through Glass Transfer White Paper (a 5.5Mbyte PDF) has been encoded in the high-speed version of the Through Glass Transfer protocol (TGXf). The file has been uploaded in 3.6 seconds from a Web browser in full-screen mode, on a PC with a 720p display, using hsTGXf at a resolution of 1240 pixels by 650 pixels with data encoded at 3 bit-per-pixel (BPP) and 10 frames-per-second (FPS).

Here, the HDMI intercepted video (RGB data saved in a lossless format) is being decoded by the receiving tool. The video is being played back frame by frame, with analysis of the screen for pixels in a certain location (the bounded region in the video). At three bits-per-pixel (BPP), the colour channels (Red, Green, Blue) in each pixel within the bounded region are evaluated. A colour channel that is "substantially light" is regarded as a binary 1, while "substantially dark" colour channel is regarded as binary 0. The bounded region therefore creates an effective datagram/packet size of 300,250 bytes (1240x650x3/8) and at 10 frames-per-second (FPS) operates with an effective throughput of 12.1Mbps.

This is a proof of concept that shows gigabytes of data could be rapidly exfiltrated (downloaded) in binary formats through remote enterprise desktop sessions via the screen, without the need for a binary client or the use of QR codes. Further, it should be clear from all of these video demonstrations, that targeting a specific implementation will not protect an organisation from data exfiltration via the screen.

Where can it be downloaded from?

Please find the proof-of-concept application that you are looking for, under the appropriate label. Note that the MD5 checksums (md5sums.txt) are available for the locally hosted files. Also note that build and test instructions can be found in the TCXf White Paper (TCXf%20White%20Paper%20-%20Midnight%20Code%20-%20v1.1.pdf).

Presentations

TGXf has been presented at various security conferences globally. If you're not looking for the presentation from a specific conference, then note the topics presented as each presentation is flavoured to that particular conference:

- COSAC (SABSA), Dublin/Ireland, 30 September 2014
  - Piano Thieving for Experts: That Bathroom Window IS Big Enough [PPT] (Piano%20Thieving%20for%20Experts%20-%20SABSA%202014%20-%20v1.5.ppt)
    Duration: 60 minutes
    Topics: principles, historical context, engineering (solution 1), architectural analysis, strategies chosen by historical figures and the US DoD.
- Ruxcon, Melbourne/Australia, 11 October 2014
  - Through Glass Transfer - Ted says this can't end well [PPT] (Through%20Glass%20Transfer%20-%20Ruxcon%202014%20-%20v1.0.ppt)
    Duration: 5 minutes
    Topics: principles, engineering (solution 1) overview, architectural overview.
- Kiwicon, Wellington/New Zealand, 12 December 2014
  - The TV people? Do you see them? [ODP] (The%20TV%20people%3F%20Do%20you%20see%20them%3F%20-%20Kiwicon%202014%20-%20v1.0.odp) / [PDF] (The%20TV%20people%3F%20Do%20you%20see%20them%3F%20-%20Kiwicon%202014%20-%20v1.0.pdf)
    Duration: 30 minutes
    Topics: principles, historical context, engineering (solution 1+2), architectural overview.
- BSidesLV, Las Vegas/USA, 5 August 2015
  - Remote Access, The APT [ODP] (Remote%20Access%20the%20APT%20-%20BSidesLV%202015%20-%20v1.1.odp) / [PDF] (Remote%20Access%20the%20APT%20-%20BSidesLV%202015%20-%20v1.1.pdf) Video [YouTube] (http://youtu.be/LAWj2KZXWKY) / [Web] (https://archive.org/details/BSidesLV2015) / [MP4] (BG09-Remote-Access-the-APT-Ian-Latter.mp4)
    Duration: 45 minutes
    Topics: principles, historical context, engineering (solution 1+2+3), architectural overview, strategies chosen by historical figures and the US DoD, Business Impact (US).
- Def Con 23, Las Vegas/USA, 7 August 2015
  - Remote Access, The APT [ODP] (Remote%20Access%20the%20APT%20-%20DefCon%202015%20-%20v1.1.odp) / [PDF] (Remote%20Access%20the%20APT%20-%20DefCon%202015%20-%20v1.1.pdf) Video [YouTube] (http://youtu.be/lfiBu8-RCdc)
    Duration: 45 minutes

Topics: principles, historical context, engineering (solution 1+2+3), architectural overview, strategies chosen by historical figures and the US DoD, Business Impact (US).

ThruGlassXfer (TGXf)

The TGXf proof-of-concept application is available via the following links:

- Ready-to-Run
  - Android
    - ThruGlassXfer in the Google Play Store (https://play.google.com/store/apps/details?id=com.midnightcode.thruglassxfer_android)
  - Apple
    - ThruGlassXfer in the App Store on iTunes (https://itunes.apple.com/us/app/thruglassxfer/id847236431?ls=1&mt=8)
  - Linux
    - tgxf-transmit-ansi-x86_64.bin (tgxf-transmit-ansi-x86_64.bin) a TGXf transmit binary for Linux 64bit platforms with ANSI output only.
    - tgxf-transmit-gd-x86_64.bin (tgxf-transmit-gd-x86_64.bin) a TGXf transmit binary for Linux 64bit platforms with ANSI and GD (graphic) output.
  - Any (PHP)
    - tgxf-transmit.php (tgxf-transmit.php.zip) a TGXf transmit script for PHP enabled platforms with ANSI and GD (graphic) output.
- Source
  - Android
    - tgxf-tgxf-android-ee36f0cf0ddc.zip (tgxf-tgxf-android-ee36f0cf0ddc.zip) the ThruGlassXfer Android source.
  - Apple
    - tgxf-tgxf-ios-dce3e1d10a40.zip (tgxf-tgxf-ios-dce3e1d10a40.zip) the ThruGlassXfer IOS source.
  - Source (ANSI C)
    - tgxf-transmit.c (tgxf-transmit.c) a TGXf transmit program.
    - tgxf-transmit-gd.c (tgxf-transmit-gd.c) a TGXf transmit program variation that outputs PNG or JPG files.
    - tgxf-transmit-pull.c (tgxf-transmit-pull.c) a TGXf transmit program variation can be timer/pull driven (used for the mobile apps).
    - tgxf-receive.c (tgxf-receive.c) a TGXf receive program (attacker computer).
    - zbar.patch (zbar.patch) a patch for the ZBar library to prevent binary data from being interpreted.

Make sure to set the *Options* so that the Max Errors is 50 and Timeout is 50 seconds on the receiving device, until you're comfortable with TGXf transfers.

The higher the performance of the receiving device, the more robust the transfer. For example, the receiving HTC One XL (1.5Ghz Qualcomm Snapdragon S4 MSM8960 SoC) out performed the receiving iPhone 5 (1.3Ghz Apple A6 SoC).

Note that these implementations have been limited to files of 65,535 bytes in size, or less. Please see the FAQ below regarding source code releases.

ThruKeyboardXfer (TKXf)

The TKXf proof-of-concept application is available via the following links:

- Any (Shell)
  - encodeBinKey.sh (encodeBinKey.sh.zip) a script to create a TKXf-like Arduino program to upload binary payloads.
  - encodeTextKey.sh (encodeTextKey.sh.zip) a script to create a TKXf-like Arduino program to upload text payloads.
- Source (ANSI C, Arduino INO)
  - tkxf-transmit.c (tkxf-transmit.c) a TKXf transmit program (attacker computer).
  - tkxf-transmit.ino (tkxf-transmit.ino) a TKXf transmit encoder for Arduino (keyboard stuffer).
  - tkxf-receive.c (tkxf-receive.c) a TKXf receive program.
  - arduino.patch (arduino.patch) a patch for the Arduino library to expose the USB HID packet.
- Demo Source (Arduino INO)
  - eicar.ino (eicar.ino) a TKXf-like Arduino program to upload the EICAR test virus.

ThruConsoleXfer (TCXf)

The TCXf proof-of-concept application is available via the following links:

- Linux
  - tcxf-dce-ansi-x86_64.bin (tcxf-dce-ansi-x86_64.bin) a TCXf Data Center End binary for Linux 64bit platforms with ANSI output only.
- Any (Shell)
  - mk.sh (mk.sh.zip) a script to help build TCXf in different configurations.
- Source (ANSI C, Arduino INO)
  - tcxf-pce.c (tcxf-pce.c) a TCXf program for the Personal Computer End (attacker computer).
  - tcxf-dce.c (tcxf-dce.c) a TCXf program for the Data Center End.
  - tcxf-usb.ino (tcxf-usb.ino) a TCXf transmit encoder for Arduino (keyboard stuffer).

clientlessTGXf (clTGXf)

The clientlessTGXf proof-of-concept application is available via the following links:

- Linux
  - clientlessTGXf-x86_64.bin (clientlessTGXf-x86_64.bin) a clientlessTGXf decoder binary for Linux 64bit platforms, dynamically linked.
- Any (Shell)
  - clientlessTGXf.sh (clientlessTGXf.sh.zip) a script to encode a file on descriptor 3 into a clientlessTGXf transfer.
- Source (ANSI C, Arduino INO)
  - clientlessTGXf-v1.0.tgz (clientlessTGXf-v1.0.tgz) a clientlessTGXf program for the Personal Computer End (attacker computer).
  - clientlessTGXf.ino (clientlessTGXf.ino) a clientlessTGXf transmit encoder for Arduino.

highspeedTGXf (hsTGXf)

The highspeedTGXf proof-of-concept application is available via the following links:

- Linux
  - highspeedTGXf-1240x650-1bpp-x86_64.bin (highspeedTGXf-1240x650-1bpp-x86_64.bin) a highspeedTGXf decoder binary for Linux 64bit platforms, dynamically linked.
  - highspeedTGXf-1240x650-3bpp-x86_64.bin (highspeedTGXf-1240x650-3bpp-x86_64.bin) a highspeedTGXf decoder binary for Linux 64bit platforms, dynamically linked.
- Any (Browser)
  - hstgxf-v1.0.html (hstgxf-v1.0.html) a script to encode a file via HTML5 canvas for highspeedTGXf transfer.
- Source (ANSI C, Arduino INO)

◦ highspeedTGXf-v1.0.tgz (highspeedTGXf-v1.0.tgz) a highspeedTGXf program for the Personal Computer End (attacker computer).

Can I try TGXf myself?

An open letter, sent to CERT Australia and the Australian Information Commissioner on 21 June 2014, has been published in both TEXT and PDF formats to Youtube. It is available in all TGXf encoding methods, so that you can get a feel for the nature of Through Glass Transfers.

The higher the version number, the more data per image. The higher the FPS number, the more images displayed per second.

If you just want to download the open letter then the Version 8 and FPS 5 encodings (blue buttons) are recommended, or follow this link to the PDF (open-letter-140621.pdf).

Optimal: Good: Possible: Unlikely:

PDF	FPS1	*TEXT*	FPS1	FPS5	FPS2
VER1		VER1	(http://youtu.be/w0Yq4vuy0jE)		
VER2		VER2			
VER8	(http://youtu.be/HjV6_IdJTgk)	VER8			
VER15	(http://youtu.be/Eimk0O6msIM)	VER15			

PDF file (open.pdf) in Youtube
Download on all platforms except Apple

TEXT file (open.txt) in Youtube
Download on all platforms

Frequently Asked Questions

Where can I get more information?

At the COSAC/SABSA conference on Tuesday the 30th of September 2014 (http://www.cosac.net/about.html) in Dublin, the presentation Piano Thieving for Experts: That Bathroom Window IS Big Enough (Piano%20Thieving%20for%20Experts%20-%20SABSA%202014%20-%20v1.5.ppt) worked through the principles and methods used to understand the vulnerability, to articulate it architecturally and then consider the flaws in enterprise security architecture, and strategies chosen by historical figures and the US DoD.

Following that presentation the 200 page (approx.) TCXf White Paper (TCXf%20White%20Paper%20-%20Midnight%20Code%20-%20v1.1.pdf) was released, documenting:

- The problem space and the principle at the core of this issue, and;
- Technology examples that drove/supported this concept from the past 15 to 20 years, and;
- Protocol specifications for the TGXf and TKXf transport protocols, and;
- Source code to the GPL reference implementations for the TGXf client and server applications, as well as those for the TKXf and TCXf applications, and;
- A complete test framework for repeating these experiments in your own environment, against your own security controls, and;
- Architectural analysis of the change created by the above, and;
- A review of some of the historical considerations of this type of problem (from the 1970's) and the strategies chosen to address it, and;
- An example of the legal outcome via the Australian Privacy Principles.

There are also a number of articles that have been written addressing different aspects of ThruGlassXfer, which you may also find useful:

- Through Glass Transfer: A New Protocol Designed to Cause Headaches for the CISO (http://www.securitycurrent.com/en/writers/richard-stiennon/through-glass-transfer-a-new-protocol-designed-to-cause-headaches-for-the-ciso), Security Current / Richard Stiennon, 30 September 2014
- Data Leak Prevention Has A New Challenge: Introducing Through Glass Transfer (http://www.forbes.com/sites/richardstiennon/2014/10/01/data-leak-prevention-has-a-new-challange-introducing-through-glass-transfer/), Forbes / Richard Stiennon, 1 October 2014
- Through Glass Transfer (TGXf) (http://hakin9.org/through-glass-transfer-tgxf/), Hakin9, 7 November 2014
- Your data: Stolen through PIXELS (http://www.theregister.co.uk/2014/12/11/your_data_stolen_through_pixels/), The Register / Darren Pauli, 11 December 2014
- Downloading Data Through The Display (http://hackaday.com/2014/12/14/downloading-data-through-the-display/), Hackaday / Brian Benchoff, 14 December 2014
- TGXf Project Warns of File Transfer through Screen Pixels (http://www.linux-magazine.com/Online/News/TGXf-Project-Warns-of-File-Transfer-through-Screen-Pixels), Linux Magazine / Joe Casad, 16 December 2014
- 没有任何安全技术能防范的黑客手段--TGXF (http://xjh.k59k.cn/20141219104-there-is-no-security-technology-can-prevent-hacking-of-tgxf.html), Security Personnel Expression / Winter Yu, 19 December 2014
- Air gaps: Happy gas for infosec or a noble but inert idea? (http://www.theregister.co.uk/2015/02/11/air_gap_feature/), The Register / Darren Pauli, 14 February 2015
- Compte-rendu de la conférence BSides Las Vegas 2015 (http://www.securityinsider-solucom.fr/2015/08/compte-rendu-de-la-conference-bsides.html), Security Insider / Arnaud Soullie, 13 August 2015
- You Built a Better Mousetrap? They Built Better RATs (https://www.duosecurity.com/blog/you-built-a-better-mousetrap-they-built-better-rats), Duo Security / Kyle Lady, 20 August 2015

If you're looking for an insider view on the development of the project, then a good friend of mine at the Cranky Potato (http://crankypotato.com/?page_id=2) blog has been following the project since before it was public:

- ThruGlassXfer - exfiltration via QRCode (http://crankypotato.com/?p=10466), 10 June 2014
- ThruGlassXfer starts to make an impact (http://crankypotato.com/?p=11405), 2 October 2014
- Stealing your data while you watch (http://crankypotato.com/?p=12044), 20 December 2014

How do I shut-down Through Glass Transfer use in my environment?

If you want to defeat the proof-of-concept TGXf protocol implementation then you could filter QR codes at strategic points in your security architecture (though the tools to do this don't currently exist). This will not shut-down all screen based transfers, however.

If you want to defeat the proof-of-concept TKXf protocol implementation then you could start by looking for Arduino USB device identifiers on end user devices, and raise alarms on their connection. Note that this value can be changed at build time (to look like a HP Chicony keyboard at 0x03F0:0x0024 for example), that this will not identify all keyboard based transfer attempts, and that it may result in a substantial number of false positives. I would not deploy this solution, but as I have been asked for the details please find the Arduino VID:PID values (https://github.com/arduino/Arduino/blob/master/hardware/arduino/boards.txt) below:

- 0x1B4F:0x9208 LillyPad
- 0x2341:0x8036 Leonardo
- 0x2341:0x8037 Micro
- 0x2341:0x8038 RobotControl
- 0x2341:0x8039 RobotMotor
- 0x2341:0x803C Esplora

In both cases, a number of mitigation proposals have been made in the COSAC/SABSA presentation and the white paper. It needs to be made clear that there is no known effective technical mitigation strategy for this vulnerability that doesn't also severely constrain productivity.

Can I protect systems by banning "smart" mobile phones?

No. TGXf is a uni-directional protocol. This means that a video recording of sufficient quality will adequately transfer the file contents out of your work place to a location more convenient to the attacker. Any of the high-end spy pen (http://www.amazon.com/s/?field-keywords=spy+pen), spy watch (http://www.amazon.com/s/?field-keywords=spy+watch) or spy glasses (http://www.amazon.com/s/?field-keywords=spy+glasses) devices will do. And before you set out to ban those devices as well, you might want to do some research as high definition miniature cameras are in (http://www.amazon.com/s/?field-keywords=spy+water+bottle) almost (http://www.amazon.com/s/?field-keywords=spy+coat+hook) anything you can imagine (https://google.com/search?q=hidden+spy+cameras+miniature&source=lnms&tbm=isch), my favourite being the religious artefacts (http://www.ankaka.com/cross-spy-camera-ultimate-hidden-digital-camcorder_p46652.html) for the additional legal implications of search and confiscatiscation.

Further-more, even if you were successful in creating a camera free environment, high performance data exfiltration would still be possible via the same channel. The architecture issues have been discussed at COSAC/SABSA and in the TCXf White Paper (TCXf%20White%20Paper%20-%20Midnight%20Code%20-%20v1.1.pdf) as well as demonstrated via clientlessTGXf.

What type of attack is TGXf?

The cause is a class of vulnerability known as a *covert channel*. Of the two types of covert channel, timing and storage, TGXf is a *storage* based covert channel. The effect is data exfiltration (data extraction or data theft).

What is the maximum theoretical download rate of screen-based transfers?

A modern computer will use HDMI to display data on a 1080p screen. If you were to read pixel data directly from the HDMI interface then that equates to 1920 pixels wide x 1080 pixels high x 24 bit (True Color) x 24 frames per second which is 1,194,363,600 bits per second or approximately 150 Megabytes per second (faster than (http://en.wikipedia.org/wiki/Gigabit_per_second#Examples_of_bit_rates)Gigabit Ethernet (http://en.wikipedia.org/wiki/Gigabit_per_second#Examples_of_bit_rates)).

What data rates are possible with TCXf?

The two protocols that comprise TCXf have differing data rates.

ThruGlassXfer (TGXf)

For implementation simplicity the TGXf protocol has been defined as supporting the following QR code versions and their respective usable capacities;

- Version 1: 21x21 pixels, 15% ECC, 14 bytes per frame, 10 usable bytes
- Version 2: 25x25 pixels, 15% ECC, 26 bytes per frame, 22 usable bytes
- Version 8: 49x49 pixels, 15% ECC, 152 bytes per frame, 148 usable bytes
- Version 15: 77x77 pixels, 15% ECC, 412 bytes per frame, 408 usable bytes

The TGXf protocol also allows frame rates of 1, 2, 5, 8 and a maximum of 10 frames per second (to allow at least a 2x over-sample on 30fps cameras).

Applying the matrix you get a data transfer rate of between 10 bytes per second (80bps) and 4,080 bytes per second (32kbps). Which isn't shabby considering that in all cases (ECC, resolution and FPS) conservative values have been chosen, and that this is a binary clean communications channel.

ThruKeyboardXfer (TKXf)

TKXf, as inplemented in the proof-of-concept, has an upper bound of 3 bytes per two milliseconds (12kbps). With additional work, it may be possible to increase this two-fold by carefully using the blanking keyboard packet (to 24kbps), and increasing it by one third if base64 encoding were used instead of a straight hexadecimal encoding (32kbps).

ThruConsoleXfer (TCXf)

Per the above calculations, TCXf as implemented today has a limit of 32kbps/12kbps (egress/ingress). This is interesting when you consider that The Cuckoo's Egg (http://en.wikipedia.org/wiki/The_Cuckoo%27s_Egg) began with a 1200 baud (1.2kbps) connection.

Of the reviews I've found and read, one of my favourites is this very apt analysis from Rodger, describing TCXf as Satan's 1200/75 Modem (http://rodger.donaldson.gen.nz/archive/2014/12/kiwicon-8-day-2/).

clientlessTGXf (clTGXf)

The clientlessTGXf protocol as implemented today peaks at about 1kbps (egress). However due to the independent decoding cycle, the download side of the same transaction (pending available processing power) runs at about 100bps.

highspeedTGXf (hsTGXf)

The highspeedTGXf upload bandwidth is limited to the largest screen resolution and refresh rate (see above for the maximum theoretical transfer rates), but in practice the rate is limited by the receiver. The off-the-shelf gamer capture device tested (Game Capture HD II) seems to peak at about 1.5Mbps (1240 x 650 x 1bpp x 2fps, egress). Of the professional video capture cards tested, the DeckLink Mini Recorder was constrained to YUV capture and therefore 1 bit-per-pixel (black or white) data, peaking at 4.7Mbps (1240 x 650 x 1bpp x 8fps, egress). While the DeckLink 4K Extreme 12G supported 10bit-per-channel RGB capture, a lack of open source tools to process the resulting R210 RGB48LE AVI file correctly limited this card to 3 bits-per-pixel data, maxing out at 12.1Mbps (1240 x 650 x 3bpp x 10fps, egress).

I'm concerned about my privacy, how can I help?

This project has been designed to draw the attention of the information security industry to the problem. I work in the industry but like you the security of my government, health, financial and other private records are beyond my control. If you are unable to contribute to the improvement process directly then consider showing your support. Feel free to link to this site from your social site, blog or email signature, and you're welcome to link to or re-serve the logo at http://thruglassxfer.com/logo.png (http://thruglassxfer.com/logo.png).

Why can't I complete a TGXf transfer?

The most common cause of failed transfers for new users is timing out before the first packet is received, or too many errors during the transfer. Make sure to set the *Options* so that the Max Errors is 50 and Timeout is 50 seconds on the receiving device, until you're comfortable with TGXf transfers.

If you are still having difficulties then see the reliability question, below.

Why is the Apple TGXf app different from the others?

Without file-system access in the Apple platform we could only hand data off to a native handler, which includes images and text. It seems that everyone else gets around this by using a local sqlite database to store their files. If this proof-of-concept application is updated, it may use this method to bring it to parity with the other builds.

This is also why the text field is awkward to use on the iPhone (with the virtual keyboard riding over it). It was a feature that only existed on the Apple and all of the testing for text transfers was done with cut-n-paste.

How do I improve reliability of TGXf transfers to my phone?

During testing on the smart phone versions of TGXf, particularly, we learnt some interesting lessons:

- Avoid glare from/on the screen of the transmitting device
- Low (CPU) powered receiving devices will not complete transfers successfully
- Some devices don't do continuous auto-focus correctly, so their first focus is either hit or miss
    - If you are testing against the Youtube demos, then wait until the box is drawn and the count-down begins before starting the download. The fine lines being displayed by the video will provide a good auto-focus target.
- If your transmission source is a video, then ensure it is being played at a native resolution and frame rate
- If you're still having problems then try decreasing the transmission version and FPS. The sweet spot with off-the-shelf hardware as at mid-2014 seems to be Version 8 with FPS 5 (in graphics mode).
    - Our testing has resulted in the performance matrix in the demonstration, above (green = good, amber = possible, red = unlikely), for the mobile platforms
- Gorilla arm (http://en.wikipedia.org/wiki/Touchscreen#.22Gorilla_arm.22) can make it hard to hold the receiving device steady for the duration of a complete transfer. Consider propping the receiving device up using readily available items. In the spirit of the Cantenna (http://en.wikipedia.org/wiki/Cantenna), a Pringles can with an appropriate modification is recommended.

Also note that the receiving CRC check has been disabled in the Android and Apple app versions of TGXf for usability reasons. The upshot is that you can try and open a file from a transfer that completed with errors (it will have chunks missing), and if you re-transmit the same file you may correctly fill the omissions from the first failed transfer even if the second transfer also completes with errors.

I don't believe that I just downloaded a PDF from Youtube! What's the trick?

Turn flight-mode on and try again - it's really happening!

Other screen and/or keyboard projects?

There are many projects that have developed either scripted command interfaces, infiltration or exfiltration via the keyboard, or a type of URL encoding or data exfiltration via the screen. Unfortunately neither my friends nor I had seen most of these projects before I developed TGXf. While none of these technologies completely overlaps with TGXf, TKXf or TCXf, Ben and Scott's QR code exfiltrator and VSzA and d3adOne's keyboard exfiltrators were so close to where I wanted to go that I probably wouldn't have started this project if I had have seen those first. For the sake of future researchers, and to credit these original thinkers, I am providing a comprehensive link repository here:

- Programmable emulated computer keyboard (USB HID scripting)
    - Programmable HID USB Keystroke Dongle: Using the Teensy as a pen testing device (http://www.irongeek.com/i.php?page=security/programmable-hid-usb-keystroke-dongle), IronGeek, 23 March 2010
    - Hak5 USB Rubber Ducky (http://hak5.org/episodes/episode-709) [device] (http://hakshop.myshopify.com/products/usb-rubber-ducky-deluxe) [toolkit] (http://www.ducktoolkit.com/) [forums] (https://forums.hak5.org/index.php?/forum/56-usb-rubber-ducky/), Hak5, 14 April 2010
    - Netragard's Hacker Interface Device (HID) (http://www.netragard.com/netragards-hacker-interface-device-hid), Netragard, 24 June 2011
- Data encoded in emulated computer keyboad (USB HID upload/download)
    - Data Leak Prevention Bypass (http://thomascannon.net/dlp-bypass/), Thomas Cannon, 08 Nov 2010
    - Leaking data using DIY USB HID device (http://techblog.vsza.hu/posts/Leaking_data_using_DIY_USB_HID_device.html), VSzA, 29 October 2012
    - Extracting data with USB HID (http://hackaday.com/2013/01/26/extracting-data-with-usb-hid/), d3adOne, 26 January 2013
- Data handle encoded in light (LED/environmental transmission)
    - Fujitsu embeds data in LED light for smartphones to detect (http://www.computerworld.com/article/2847270/fujitsu-embeds-data-in-led-light-for-smartphones-to-detect.html), Fujitsu, 17 November 2014
- Data encoded in light (LED/environmental transmission)
    - Datalink library for some Timex watches (http://datalink.fries.net/), dfries, 28 January 2006
    - Visible light communications: achieving high data rates (http://smartlighting.rpi.edu/resources/PDFs/Smart_Lighting_ERC_OBrien_11_02_08.pdf), Dominic O'Brien, 8 February 2011
    - Using Visible Light Frequencies for Wireless Data Transfer (http://www.techthefuture.com/technology/using-visible-light-frequencies-for-wireless-data-transfer/) [video] (https://www.ted.com/talks/harald_haas_wireless_data_from_every_light_bulb), Harald Haas, 31 July 2011
    - LightMessage by PureVLC (http://visiblelightcomm.com/lightmessage-by-purevlc/), PureVLC, 10 May 2012
    - Supermarket LED lights talk to smartphone app (http://www.bbc.co.uk/news/technology-32848763), Leo Kelion, 22 May 2015
    - LED-it-GO Leaking (a lot of) Data from Air-Gapped Computers via the (small) Hard Drive LED (http://cyber.bgu.ac.il/advanced-cyber/system/files/LED-it-GO_0.pdf) [video] (http://youtu.be/4vIu8ld68fc), Mordechai Guri, Boris Zadov, Eran Atias,Yuval Elovici, 22 February 2017
- Data handle encoded in computer display (retrieved visually)

    - Fujitsu figures out how to transfer files using a camera (http://www.geek.com/mobile/fujitsu-figures-out-how-to-transfer-files-using-a-camera-1536734/), Fujitsu, 22 January 2013
    - inTouch Enables Data Transfer Data between Mobile Screens using a Miniature Device (http://www.atelier.net/en/trends/articles/intouch-enables-data-transfer-data-between-mobile-screens-using-miniature-device_424958), inTouch, 31 October 2013
- Data encoded in computer display (retrieved electro-magnetically)
    - The Complete, Unofficial TEMPEST Information Page (http://www.jammed.com/~jwa/tempest.html), Joel McNamara, 17 December 1996 - 4 December 1999
    - AirHopper: Bridging the Air-Gap between Isolated Networks and Mobile Phones using Radio Frequencies (http://www.slideshare.net/mordechaiguri/air-hoppermalwarefinale), 29 October 2014
- Data encoded in computer display (retrieved visually)
    - QuickeR: Using video QR codes to transfer data (http://stephendnicholas.com/archives/310), Stephen Nicholas, 3 October 2011
    - DLP Circumvention: A Demonstration of Futility (http://labs.neohapsis.com/2012/07/20/dlp-circumvention-a-demonstration-of-futility/) [source] (https://github.com/Neohapsis/QRCode-Video-Data-Exfiltration), Ben Toews & Scott Behrens, 20 July 2012
    - Using a flashing LCD monitor to transfer data (http://hackaday.com/2013/02/25/using-a-flashing-lcd-monitor-to-transfer-data/), Mike Szczys, 25 February 2013
    - Eye-Fi - Notebook file transfer through QR codes, using only the screen and webcam (https://github.com/anderson-/Eye-Fi), Anderson de Oliveira Antunes, 14 November 2014
    - Transfer data via the screen (http://youtu.be/RbQw6dTKlI0) [video] (http://youtu.be/Zfm6PjBaAdA), foo52ru, 2 December 2014
    - VisiSploit: An Optical Covert-Channel to Leak Data through an Air-Gap (https://arxiv.org/pdf/1607.03946.pdf), Mordechai Guri, Ofer Hasson, Gabi Kedma, Yuval Elovici, 13 July 2016
- Computer manipulation of touch interfaces
    - Iphone clicker (http://youtu.be/sxLux91MB_Y), Mickey Coke, 21 September 2011
    - Arduino Plays Piano Tiles (Tutorial) (https://github.com/psurya1994/arduino-plays-piano-tiles) [video] (http://youtu.be/8hlQ0MiowN8), psurya1994, 6 December 2014
    - Arduino Device Play iOS Game "White Tiles" (http://hackaday.io/project/3658-arduino-device-play-ios-game-white-tiles) [video] (http://youtu.be/QLNiDpJHrC4), denmen7, 21 December 2014
    - Stupid Games call for Stupid Cheating (http://youtu.be/4ElZec033vQ), imkNick, 4 February 2016
    - Piano Tiles 2 Robot | Beginner 21.079 Record (http://youtu.be/fqOW84ZTL7k), DenverFinn, 26 April 2016

If you believe that your project should be included here, then please let me know.

Who created TGXf, TKXf and TCXf?

Through Glass Transfer is a Midnight Code (http://www.midnightcode.org/projects/TGXf/) project, created by Ian Latter (mailto:ian.latter@thruglassxfer.com). The reference TGXf C code was ported to both IOS and Android by James Hudson (http://www.disconnectionist.com/).

When was this information last updated?

This site was published on 8 June 2014. The last update was Sun, 05 Mar 2017 02:35:55 +0000