Bug or Feature? Covert Impairments to Human Computer Interaction

John V. Monaco
Naval Postgraduate School, Monterey, CA

ABSTRACT
Computer users commonly experience interaction anomalies, such as the text cursor jumping to another location in a document, perturbed mouse pointer motion, or a disagreement between tactile input and touch screen location. These anomalies impair interaction and require the user to take corrective measures, such as resetting the text cursor or correcting the trajectory of the pointer to reach a desired target. Impairments can result from software bugs, physical hardware defects, and extraneous input. However, some designs alter the course of interaction through covert impairments, anomalies introduced intentionally and without the user's knowledge. There are various motivations for doing so rooted in disparate fields including biometrics, electronic voting, and entertainment. We examine this kind of deception by systematizing four different ways computer interaction may become impaired and three different goals of the designer, providing insight to the design of systems that implement covert impairments.

Author Keywords
deception; influence; behavior change; cybersecurity; interaction

CCS Concepts
•**Human-centered computing → Human computer interaction (HCI); Interaction design;**

INTRODUCTION
This work is concerned with the subtle anomalies that occur during human computer interaction. Examples include the text cursor suddenly jumping to another location in a document, perturbations in the motion of the pointer diverting its trajectory, or a disagreement between tactile input and touch screen location. These anomalies impair interaction in the sense that they generally contradict the user's intentions and influence the user's behavior. Such anomalies can go undetected by the user and may even be corrected unconsciously. Their cause may be legitimately attributed to a software bug, physical hardware defect, or extraneous input to a peripheral device. As a result, users have acclimated to their presence [73].

Recent research, commercial products, and cyber attacks have leveraged this phenomenon by intentionally creating interaction anomalies to influence users. We refer to this as a *covert impairment*. They are covert in the sense that the anomalies either go undetected by the user or may be erroneously attributed to another source without the user's awareness of the designer's intentionality. They are impairments in the sense that the anomalies elicit a reaction from the user that would not have been performed otherwise and alter course of interaction. Unlike dark patterns [40], covert impairments implement deceit through interaction itself rather than the user interface, i.e., there's no deceit without interaction. We formally define covert impairments in the next section.

The motivations for this practice vary across fields, where tactics employed range from subtle to obvious. Covert impairments are leveraged as both a defense and an attack in cybersecurity. Reactions tend to be characteristic of the individual or a demographic attribute common to a group of users, which lends this technique to aid in user identification and profiling [78]. Such manipulation goes beyond traditional behavioral biometrics [2], which passively observe an individual's behavior for the purpose of identification or authentication, to that of making active measurements. They are additionally used to detect automated software, or bots [84]. In this way, the impairments elicit a reaction that suggests the presence of a human user, assuming it would be difficult for a machine to replicate. As an attack, covert impairments are a coercion technique that alter the user's perception of interaction in a way that leads to an outcome chosen by the designer.

In cognitive science, this technique is used to induce chosen affective states, such as stress or anxiety. Artificially delaying keyboard events during gameplay has been shown to cause stress [72]. Likewise, randomly dropping mouse motion events, which causes the pointer to stop moving, induces frustration [76]. These kinds of impairments are well studied in human computer interaction when occurring "naturally", such as from network lag or high system load [70]. Both large amounts of lag and dropout can seriously degrade task performance [53].

Little is known about the practice of covert impairments and how users react to them. Most efforts have considered impairments from legitimate sources, i.e., unintentional impairments such as network lag [53, 30, 70] or poor user interface design [57]. However, the use of covert impairments outside of the HCI community is widespread. The idea has been patented several times over [78, 83], is currently implemented in com-

Deceive	A system action not revealed to the user
Modify	that modifies input from a peripheral device
Influence	to elicit a reaction or force a change in behavior.

Figure 1. Definition of covert impairment.

mercial products [84], and remains a technique routinely used by cyber adversaries to perform coercion [47, 43, 98].

The goals of this paper are to provide a coherent framework within which covert impairments may be discussed and to highlight their usage in the wild. To better understand this practice, we systematize knowledge across several fields and describe four different ways interaction may become impaired, intentional or not. We identify legitimate sources of impairments and characterize their prevalence. Applications that implement covert impairments are then identified and characterized within a framework that captures impairment mechanics, designer's goal and intent, and the user's reaction. As a form of deception, the use of covert impairments carries risks, such as tricking users into performing undesirable actions or degrading task performance. We examine these risks and identify other potential use cases for covert impairments.

DEFINITIONS AND BACKGROUND

We define a covert impairment as an intentional design consisting of three elements. These include: 1) an act of deception; 2) the modification of user input to a computer; and 3) influence over the user's behavior. Our definition is shown in Figure 1. We describe each element in detail and motivate this definition by describing designs that exhibit some, but not all, of these elements.

Designs that deceive users

Deception occurs when a user's belief about a system is "demonstrably false" due to a system action or claim [1]. The use of deception is widespread in HCI and ranges from malevolent, which aims to benefit the designer at the expense of the user [26, 40], to benevolent, which aims to benefit both the user and the designer [1].

Deception in HCI has long been considered taboo despite there being legitimate use cases for deceptive design. While malevolent deception encompasses designs considered to be malicious or harmful to the user, such as dark patterns [40], deceptive techniques considered to be benevolent have grown to cover a wide range of legitimate use cases. Many of these practices aim to bring the user's mental model and system image in synchrony. This includes anthropomorphic designs, such as a game reducing the skill of a computer opponent to make it seem more human like [93], and the concealment of uncertainty, such as using a heuristic to estimate the amount of time a long running processing will take.

Computer security has a long history of using deception [41]. Design choices are often motivated by the requirement to balance the needs of a single user (e.g., accessing a protected resource) with that of all users (e.g., keeping resources compartmentalized). Deception is generally accepted and regarded as benevolent when it is intended primarily for malicious actors. For example, honeypots are illegitimate computer resources used to lure attackers, enticing them to waste time and expose tactics. Likewise, concealing the results of a failed login attempt by not informing the user whether their username or password was incorrect can also act as a deterrence to attackers by making it more difficult to perform account enumeration. This practice benefits the group (all users with accounts on the website) at the expense of the individual (a single user who mistypes their credentials must determine which field needs to be corrected).

In the framework of Adar et al. [1], this work is concerned with behavioral deception, which includes deceits that modify the way a system responds to user input. Behavioral deception generally aims to overcome a user's physical limitations, such as dynamically adjusting the pointer transfer function or the use of semantic pointing [16] to increase pointing accuracy. However, unlike the behavioral deception described in Adar et al. [1], which aims to synchronize user mental model and system image, the deception described in this work does exactly the opposite. Covert impairments are deceptive because the user is given a false impression of how the system interpreted their input. The act of modifying input or injecting an input error violates the user's mental model, for example if pressing a keyboard key has no effect.

Designs that modify input

On modern computers, there are up to a dozen individual components working together to map a physical action (pressing a key on a keyboard) to system state (printing a character on screen). The act of typing on a keyboard includes the: physical key, button and actuator, sensing mechanism (e.g., keyboard matrix circuit), microcontroller, protocol to communicate with the host (USB or PS/2), interrupt mechanism on the host, operating system scheduler, keyboard driver, keymap for translating key codes to locale-specific characters, and application itself (e.g., web browser). And if the application is remote, the network stack.

Each component in this pipeline determines what effects the user's action will have. The effect may be intentional, such mapping keys to uppercase characters in a "Caps Lock" state, or unintentional, such as the latency between pressing a key and seeing the result printed on the screen [95]. In this sense, modifications to peripheral input can be introduced anywhere along the pipeline.

This work is concerned with designs that intentionally modify peripheral input. Many such designs exist, most of which aim to increase task performance by predicting what action the user intended to perform. For example, key debouncing applied on the keyboard eliminates the spurious keystrokes that would occur as the circuit settles into a closed state [55]; the autorepeat feature enables keys held down for a long duration to be repeated with a single action rather than a series of separate keystrokes; and an application can make modifications to the key mappings, such as by making the "Enter" key behave like "Tab" when filling out a web form [4].

Modifying input from a pointing device, such as a computer mouse, is ubiquitous. Pointer acceleration is the method by which the displacement of the pointer on screen depends on the speed of the device, enabling longer distances to be covered by short, quick movements. Other modifications, such as pointer smoothing, adjustment of target dimensions, and context sensitivity, aim to increase pointing task performance [71, 12, 16]. The transfer function, which defines how physical motion is translated to pointer displacement, has remained elusive and varies across different operating systems with respect to parametric model and parameter choice [22]. There also exist a range of assistive technologies that aim to increase accessibility for disabled users, such as pointer motion smoothing to reduce spatial jitter, and bounce keys to avoid spurious keystrokes [92, 82].

The main difference between covert impairments and the techniques described above is that of the designer's goal: while the techniques above aim to facilitate interaction, often by bringing the system image closer to the user's mental model and increase task performance, covert impairments do exactly the opposite. The modifications introduced through covert impairments diverge the user's mental model from system image with the goal of influencing user behavior.

Designs that influence users

Designers may attempt to guide the user toward a particular course of action by influencing their behavior. It is useful to characterize methods of influence within Newell's time scale of human action [63], which defines a hierarchy of human actions based on their respective temporal dynamics: biological (milliseconds), cognitive (seconds), rational (minutes to hours), and social (days to weeks).

At the rational level, aesthetic manipulation, such as the use of color or placement of buttons in a prompt, can influence choice outcome [26, 40]. At the social level, users on a professional networking website might be rewarded with virtual insignia for volunteering their information [19]. Advertisers frequently utilize interface design to influence spending behavior and drive users to click on links. These designs are referred to as *dark patterns* [56]. The use of such techniques through established psychological principles of influence is also ubiquitous among phishing attacks, the goal of which is to convince a user to reveal credentials or other secret information [89].

Not all designs that influence user behavior are malicious. The same kinds of techniques can be leveraged to influence users in neutral or positive way. In training and simulation environments, the gender of an avatar or human conversant can play a role in whether users are receptive to changes in attitude [97]. Likewise, subliminal cues in a virtual learning environment can increase knowledge retention [23] and influence the outcome of choice tasks [14].

This work is concerned with the designs that influence users within the cognitive level of Newell's time scale. Covert impairments do not attempt to, e.g., persuade the user to explicitly divulge information, but instead invoke a psychomotor response that may lead to such an outcome. This is accomplished by enticing the user's mental model to depart from the system image, which can happen by: 1) inducing a wrong belief in the system; 2) presenting an observation that conflicts with the user's mental model; or 3) causing the user's predictions about how the system will behave to diverge [65, 81]. The result of this departure is an action taken by the user that would not have occurred otherwise.

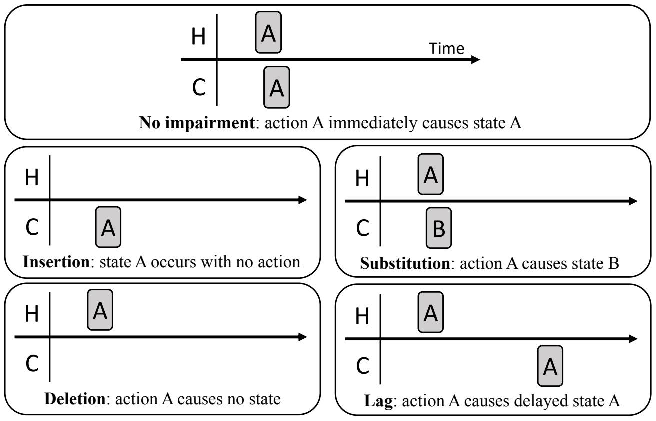
H
C
A
Time
No impairment: action A immediately causes state A
Insertion: state A occurs with no action
Substitution: action A causes state B
B
Deletion: action A causes no state
Lag: action A causes delayed state A

Figure 2. Four kinds of impairments to human computer interaction. In a normal scenario (top), motor action *A* performed by the human causes system state *A*. An impairment is one of: spurious system state (insertion), action with no effect (deletion), perturbed system state (substitution), or action with delayed effect (lag). H=human, C=computer.

Relation to dark patterns

As a design practice that may be used to inflict harm upon users, covert impairments are closely related to dark patterns [40]. However, there are several key differences. Notably, not all covert impairments are malicious whereas all dark patterns are. Some impairments that elicit behavior are better described by benevolent deception [1] and have valid use cases, such as an implicit CAPTCHA for bot detection [35]. Likewise, not all dark patterns are deceptive, e.g., nagging, whereas all covert impairments are. Although many dark patterns are deceptive, deception itself is not a defining feature.

Another key difference is that covert impairments and dark patterns manifest at different layers on the computer. To draw an analogy, consider the Open Systems Interconnection (OSI) model [99]. Dark patterns manifest at the "application layer": they are dominated by user interface design choices such as shape, color, and placement. Covert impairments manifest at the "transport layer": they modify input events before they reach the application, such as pointer acceleration through a transfer function.

KINDS OF IMPAIRMENTS

We start with a general discussion of interaction impairments. An *interaction impairment* is a breakdown that occurs somewhere along the processing pipeline between the user's physical action sensed by a peripheral device and the invoked system state. While impairments may be unintentionally caused by one of several legitimate sources, described in the next section, covert impairments are those intentionally introduced and hidden from the user. We characterize four different ways that interaction may become impaired, intentional or not.

For the purpose of our definition, we view interaction as that of mechanical determination [42]: a human action input to the system determines the system state. This view is implicit within the keystroke-level model (KLM), which defines a set of physical motor operators (actions) the user performs to accomplish a task [20]. Given this view, two discrete sequences are of interest: the sequence of physical actions performed by the user, such as pointing to a target, and the sequence of system states that occur as a result of each action, such as rendering the pointer at a different location on the screen. When the system acts reasonably according to the user's mental model [21], such as the displacement of the pointer being proportional to the physical movement, then no impairment has occurred. This scenario is depicted in Figure 2 (top) where motor action *A* causes system state *A*.

We borrow terminology from string metrics to describe the different kinds of impairments to HCI. Levenshtein edit distance measures the distance between two discrete sequences by the minimum number of edit operations needed to transform one to the other: insertions, deletions, and substitutions [51]. We characterize four different ways in which interaction may become impaired, which include the three primitive string edit operations and an additional operation that permits reasoning about time. These are each depicted in Figure 2 and described below.

Insertions are spurious system states for which the user never performed a corresponding action. These often occur in or around other actions the user did perform, making their presence more difficult to detect. For example, the mouse pointer jumping to another location on screen during a pointing task, or the text cursor moving to different position in a document while typing, both common as described in the next section. Spurious keystrokes have been around almost as long as the keyboard itself. On older keyboards, the layout of the matrix circuit resulted in "ghosting" (the sensing of a spurious keystroke), a phenomenon that occurs when three adjacent keys are pressed and a fourth is inadvertently sensed [29].

Deletions are user actions that are ignored by the system such that there is no perceived effect or change in system state. This may occur when the system is overloaded to the point that peripheral hardware interrupts are unacknowledged and the system appears "frozen". They may also result from physical defects, such as a loose cable, a missing or broken keyboard switch, or dust on an optical mouse sensor. Deletions may also be a limitation of the communication protocol, such as the "masking" that occurs when more than 6 keys are simultaneously pressed on a USB keyboard [37]. In distributed applications where user actions are transmitted over a network, packet dropout has a similar effect.

Substitutions qualitatively encompass the most diverse set of breakdowns that may occur. A substitution occurs when the user's action causes an unintended system state. With textual input, this can result from an incorrect mapping of keyboard scan codes. Modifications to the keyboard state can also result in substitutions, such as inadvertently toggling the Caps Lock state of the keyboard. Some authentication protocols even anticipate this impairment by accepting the user's password with case reversed [24]. Substitutions for pointing actions are prevalent. The transfer function used to translate physical motion to pointer motion is often nonlinear (dependent on pointer velocity) and context sensitive (attracting to nearby UI elements or boundaries). While the transfer function aims to bring the system image in synchrony with the user's mental model, it can disrupt this process when effects the user did not intend are encountered, such "pointer snapping" when finer resolution is preferred [15].

Lag occurs when the user's actions cause a delayed system state. There is an inherent lag associated with any electronic sensing device, where the lower bound is formed by the speed of light to transmit information from the sensor to the computer. However, other components greatly contribute to lag, including sensor mechanics, USB polling rate, and operating system scheduling [95]. Lag is typically not perceived if it remains within tolerable values, but as lag increases it can alter the course of interaction through a degradation in human performance [53].

UNINTENTIONAL IMPAIRMENTS

Interaction impairments that are not intentionally introduced may be attributed to a legitimate source. For some applications, it is this very notion that masks the use of covert impairments as a tool to elicit and measure human behavior, as the impairments introduced are intended to be subtle and remain undetected by the user. There are four main sources of unintentional impairments.

Hardware defects

Faulty hardware and malfunctioning physical components can impair interaction. This may occur when a sensor stops working correctly or another component interferes with the sensor. For example, a broken keyboard switch will result in deletions if the circuit for that particular key cannot be completed. Likewise, a sticky key can result in spurious input. A swollen battery on a laptop or mobile device can cause unexpected behavior, such as extraneous input to the keyboard (insertions) and displaced touchscreen events (substitutions).

Software bugs

User interfaces are inherently asynchronous and input/output bound. As a result, software bugs that result in interaction impairments are common. Insertions and substitutions can result from bad event handling, especially in distributed applications [73]. Stale updates to a client interface from a server or out-of-order event processing in a text editor may result in updates that do not reflect the actions the user performed. For example, the "jumping cursor" bug is prevalent among web applications [61]. When this occurs, the text cursor jumps to another position in a document, after which characters are inserted at the wrong location. Because the input events are processed asynchronously, these kinds of bugs may be difficult to replicate, identify the cause, and fix.

Deletions and substitutions can occur when a peripheral driver is misconfigured or mismatched to the device. Typing on a keyboard with the wrong driver installed or an incorrect keymap can result in substituted characters. Many touchscreens consist of a transparent touch-sensitive material placed on top of the screen. To accurately sense the location of touch, the device must be calibrated either during the manufacturing process or before use [28]. As a result, miscalibrated devices can report locations that differ from what users experience. These kinds of calibration bugs can have implications for electronic voting [47].

Resource constraints

Impairments can result from resource contention and other bottlenecks to processing input events. On time-sharing systems, applications must compete for CPU, system memory, network bandwidth, and power consumption. Energy constraints can result in either deletions or insertions, caused by, e.g., a low battery in a wireless mouse or keyboard [7]. The effects of lag on task performance and user perception are well-studied [53], which can result from multiple processes competing for CPU or high utilization of system memory. For example, keystrokes are buffered during periods of high CPU contention, and on touchscreen devices, even small amounts of lag may become perceptible [46]. It is interesting to note that modern devices generally exhibit greater lag than their vintage counterparts, partly attributed to a more complex input pipeline [27]. Distributed applications must additionally deal with the effects of network latency (packet transit time), jitter (variations in latency), and dropout (lost packets) [70]. Network latency is the sum of propagation (the physical distance separating two hosts), transmission, processing, and queuing times [30].

User errors

The user is sometimes the source of interaction impairments. Typing errors have been extensively characterized [75]. User errors at the execution phase of typing can cause substitutions due to misplaced fingers, deletions by not applying enough force to actuate a key, and insertions by pressing two adjacent keys together. The cost to correct these kinds of errors is also well studied [8]. Extraneous input to either a keyboard, mouse, touchscreen, or touchpad can result in spurious events, although there exist methods to prevent this to some extent, such as disabling the touchpad while the user is typing [50]. Impairments can also arise when the user has limited or inaccurate knowledge about an interface [81]. The inability to predict how a system will behave may result in the user performing an action that has no effect (deletion) or the wrong effect (substitution).

COVERT IMPAIRMENTS

The interaction impairments described in the previous section may be legitimately caused by hardware defects, software bugs, resource constraints, or user errors. While these are unintentional, the idea of intentionally introducing an impairment to influence user behavior has been applied in a diverse range of applications across academia and industry. In this section, we describe this phenomenon and then identify and examine specific applications where it can be found.

To analyze covert impairments, we followed an inductive coding procedure similar to [40]. We started with a set of exemplars encountered by ourselves and colleagues working in related areas. From these exemplars, we extracted four canonical impairments and noted the equivalence to string edit operations. We then searched for other applications that: 1) modify user input; 2) don't reveal to the user those modifications; and 3) decrease task performance. The last requirement excludes assistive technologies that meet the first two criteria.

Finding additional applications that fit these criteria was nontrivial due to differences in terminology across different fields. For example: dropout=deletion, perturbation=substitution, and lag=latency, to name a few. To gather a list of candidate applications, we used all combinations of these terms with {user input, input event, keyboard, mouse, pointer, click, touchscreen} on general, academic, and source code search engines. This resulted in a collection of about 100 candidates: scholarly articles, patents, blog posts, and device manuals. Each item was reviewed for inclusion as either an unintentional impairment, covert impairment, or neither. The covert impairments identified are summarized in Table 1 with an overview of the framework provided below.

Overview

We characterize each covert impairment along five dimensions, summarized in Figure 3: the *modality* and *kind of impairment*, which capture the mechanics of the impairment; the designer's *goal* and *intent*; and the user's *reaction*.

Impairment mechanics

The *modality* is the peripheral sensor through which the impairment occurs. This could be a keyboard, pointing device, or other sensor capable of modifying the system state. Note that because the impairments we describe occur only within the context of interaction, there must also be some feedback to the user, such as a display screen.

The impairment itself is a modification to the input event provided by the sensor. The *kind of impairment* can be one of the four canonical impairments: *insertion*, *deletion*, *substitution*, *lag*, or some combination of these. Because the user actions and system states each form a discrete sequence, any difference between these two sequences can be stated in terms of the four impairments.

Designer goal and intent

Covert impairments differ greatly with respect to the designer's *goal* for some specific outcome in the course of interaction, which can be to *elicit* information from the user, *coerce* the user to perform an action, or *induce* a chosen affective state.

Elicit: Because a covert impairment departs a user's mental model from the system image, they often need to be corrected. This involves the user first realizing that an error has occurred and then taking the necessary actions to bring the two models back in synchrony. In this view, an impairment is a kind of stimulus to the user, with the response being the actions the user performs to correct it. An application may use a covert impairment to elicit user behavior, which might reveal information such as the individual's identity or age [77], or could simply indicate that the presence of a human user as opposed to automated software.

Coerce: With a goal to coerce, the result of the user's reaction is of interest rather than the reaction itself. The designer may coerce the user to interact with a particular UI element, reveal sensitive information, e.g., a password, or perform some other

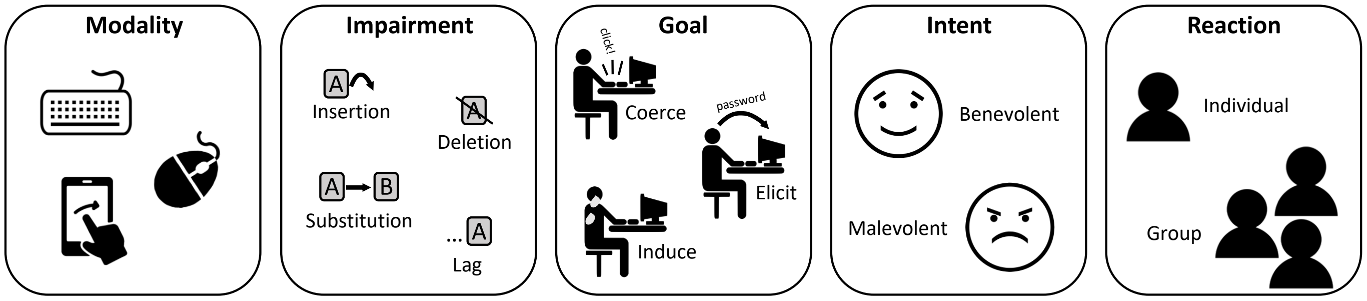

Figure 3. Covert impairments are characterized along five dimensions: *modality*, *kind of impairment*, *goal* and *intent* of the designer, and user *reaction*.

(presumably) undesirable action. It may never become apparent to the user that an impairment occurred, as applications that use coercion go to great lengths to hide the presence of the impairment, as discussed below. Within the dark patterns framework, coercion is similar to forced action with the key difference being that a forced action typically enables access to a new feature; coercion occurs unknowingly.

Induce: Covert impairments influence the interaction process by forcing the user to make errors, perform repetitious actions, or wait for the system to respond. As a result, they can alter the user's affective state, degrade task performance, or even provide entertainment value. The presence of covert impairments might invoke a chosen affective state, such as stress or anxiety, especially when a reward is at stake. They may also engage the user and invoke anticipation or excitement by making a task more challenging.

Closely related to the designer's goal is their *intent*. The designer's intent largely characterizes which party (user or designer) will benefit from the impairment. Although intent is more of a continuum [1], we discretize intent into two classes. With *malevolent* intent, the designer prioritizes their own needs over the needs of the user. In this case, there is risk of harm to the user without any perceived benefit. With *benevolent* intent, the needs of the user are either prioritized or balanced with the needs of another party. There still may be risk of harm to the user, but this risk is minimized or appropriately balanced with other needs, such as balancing the security of a user's account with difficulty to gain access.

User reaction

To achieve one of the goals described in the previous section, the designer expects the user to react in a particular way. This *reaction* can be broadly categorized according to whether the designer is relying on *individual* or *group* behavior.

Individual: Users exhibit unique and identifiable behavior when interacting with a computer interface, resulting from idiosyncratic factors such as motor skill, cognitive ability, and physiology [34]. Although these differences pose a challenge for HCI design [32], they have enabled other applications. Interaction with keyboard, mouse, touchscreen, and other HCI modalities enables user identification and profiling to be performed [2]. Related work has leveraged between-subject differences in HCI behavior to diagnose medical conditions, including Parkinson's disease [39], and to detect users who are being deceptive [13]. These applications are possible because of the individual differences in behavior that persist over time.

Group: Humans are largely bound by the same set of physiological and cognitive constraints. It is this principle that gives rise to the general laws that characterize the way members of a group will behave when interacting with a computer. Different users experience about the same amount of difficulty when performing pointing tasks [36] and operating a keyboard [75]. They are also bound to the same perceptual limitations, and as a result, vulnerable to the same kind of manipulation through interface design [40]. Covert impairments may leverage these between-subject similarities, for example, to differentiate between a human and a bot [85] or coerce a user to click on a malicious element by leveraging a common psychomotor reaction [3].

Impairments that elicit

There exist several commercial products that use covert impairments to elicit behavior. These applications influence the user's behavior to reveal their identity, demographic attributes, and the presence of a human user.

User identification

Keystroke and mouse biometrics are techniques by which users are identified based on between-subject differences in typing and pointing behavior [2]. These are largely passive techniques; mouse events and keystrokes are monitored while the user performs a routine activity, such as typing login credentials or filling in a form. Covert impairments take this technique a step further by making active measurements during this process and gauging the user's reaction, which can further reveal idiosyncratic tendencies.

Cognitive biometrics is a technique in which pointer motion events are perturbed such that the user must perform a corrective action to reach their anticipated target [78]. Motion events are inserted while the user performs a pointing task, resulting in a trajectory that diverges from the user's physical action. The way in which the user corrects the deviated path is relatively unique, considering features such as reaction time, velocity, acceleration, and angular variations. Although this technique was originally intended for benevolent purposes, it could also be used to violate user privacy over anonymizing networks such as Tor.

Application	Modality	Impairment				Goal	Intent	Reaction	Reference
		I	D	S	L				
User identification	Pointer	●				Elicit	Ben. or Mal.	Individual	[78]
Fraud detection	Pointer	●	●	●		Elicit	Benevolent	Individual	[84, 85]
Bot detection	Keystroke	●				Elicit	Benevolent	Group	[35]
Electronic voting	Touchscreen			●		Coerce	Malevolent	Group	[47, 10]
Clickjacking	Button			●		Coerce	Malevolent	Group	[43, 3, 98]
Cursorjacking	Pointer	●		●		Coerce	Malevolent	Group	[54, 33, 49, 68]
Strokejacking	Keystroke			●		Coerce	Malevolent	Group	[58, 52]
Affective computing	Keystroke		●		●	Induce	Benevolent	Group	[48, 76, 31, 72]
Performance modeling	Pointer		●	●		Induce	Benevolent	Group	[66, 17, 94]
Gaming/entertainment	Joystick		●	●		Induce	Benevolent	Group	[67, 44]

Table 1. Summary of applications that use covert impairments.

Fraud and bot detection

User interface automation is frequently employed for fraudulent and malicious purposes, such as to create fake accounts, distribute spam, and perform denial of service attacks. This requires automated software to perform navigation, text entry, and pointing tasks. To prevent this kind of abuse, users are required to pass a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA), a kind of challenge thought to be difficult for machines but easy for humans [90]. As machine learning is able to solve perceptual challenges with increasingly better accuracy [96, 18, 38], CAPTCHAs have become more difficult for human users. It is therefore desirable to design a CAPTCHA that is implicit, or transparent to the user [11]. Some recent techniques rely on user reactions to covert impairments.

Malboard is an attack that automates text entry, injecting malicious commands to a computer through a keyboard interface after it has learned the victim's behavior [35]. Because it mimics a human user, this enables it to bypass existing detection mechanisms based on keystroke dynamics. As a defense, the computer may generate spurious keystrokes while the user types. The user's reaction to these insertions is an indication that the user is human since correcting the impairment, e.g., pointing to the inserted character and then deleting it, requires a closed-loop feedback mechanism. When the spurious keystrokes go uncorrected, it is assumed that the text entry was performed by an automated device.

The trademark *Invisible Challenges* encompasses several patented techniques that implement covert impairments to elicit behavior [85]. These are used in a commercial product designed to detect fraud and automation by analyzing how users react to impairments applied to the mouse pointer. Impairments include: inserting motion to elicit a correction to the pointer trajectory; substituting motion events with smaller deltas so the user must exaggerate or repeat the same action; and deleting motion events by making the pointer disappear altogether, after which the user must move the mouse to reveal the location of the pointer on screen. Like the Malboard defense, these techniques provide an implicit CAPTCHA through the user's reaction.

Impairments that coerce

Covert impairments that influence the user to perform a chosen action are coercive, and applications that utilize this technique are generally malevolent: they place the needs of the designer over the needs of the user.

Click-, cursor-, and, stroke-jacking

UI redressing is a technique in which an interface is modified to coerce a user to perform an undesirable action [43]. Such an attack is feasible when different applications share the same screen canvas, such as a webpage that includes both primary and third-party content. Covert impairments comprise a subset of UI redressing attacks that manipulate user input. These attacks are used to spread malware, manipulate online reputation metrics, and cause a victim to divulge sensitive information.

Clickjacking attacks have been around for nearly two decades [45] but remain a widespread issue [98]. In a clickjacking attack, the victim is coerced to interact with a malicious element in a webpage using the mouse button. This is accomplished through event substitution, whereby the mouse button event on the intended element is substituted for an event passed to a malicious element. A transparent element, not perceptible to the user, placed on top of the malicious element will register any mousedown and mouseup events performed at that location. Alternatively, JavaScript can be used to quickly display a malicious element or remove a benign element somewhere on the webpage before the button event occurs. This same kind of attack has been leveraged for touchscreen and keyboard input [64, 52]. In a strokejacking attack, the victim's keystrokes are diverted from a benign element to a malicious element in a similar manner [58].

Cursorjacking is arguably more sophisticated and makes use of several kinds of impairments [43]. In one variant, the mouse pointer is rendered at a different location on screen, either through spoofing or the use of an elongated cursor with transparent region. The click point set by the designer is located at a constant offset to the rendered cursor such that when the victim points to and clicks on a benign element, the click event is registered by a malicious element located at that same offset [33, 54, 49, 68]. A canonical example of this

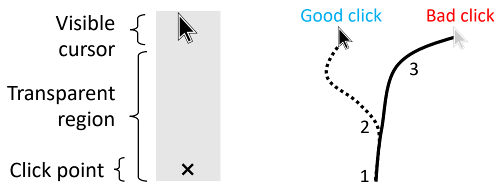

Figure 4. Coercive impairments to the pointer. Left: pointer motion is substituted by rendering the pointer at on offset from the click point (denoted by the ×). Right: inserted pointer motion causes the user to navigate to a malicious element. The dotted line represents the rendered path and solid line represents the actual path. The pointer starts at (1); leftward motion events are inserted at (2); the user compensates with rightward motion at (3) and reaches the malicious target.

attack is shown in Figure 4 where the benign pointer location is substituted for a malicious one.

Some of the attacks described above can be detected with simple rules that ensure the integrity of the display and pointer [86, 43]. However, pointer destabilization, a cursorjacking variant, is particularly difficult to detect [3]. In this attack, mouse motion events are gradually inserted while the user performs a pointing task. This causes the path of the pointer to deviate from the user's physical action. Upon noticing this error, the user reacts by correcting the pointer trajectory in such a way that the actual pointer location ends up over a malicious element. An example of this attack is shown in Figure 4. For the attack to work, users must correct the pointer trajectory in a predictable way, a kind of group behavior that has been the focus of prior work [17, 94].

Electronic voting

Similar to clickjacking, coercive attacks on electronic voting machines have emerged. Electronic voting machines offer reduced operating costs over manual counting and reduced time to aggregate and disseminate results. However, the integrity of using these machines continues to be a concern, as a variety of ways in which they may be compromised have been demonstrated [10]. In particular, electronic voting machines that utilize a touchscreen interface are susceptible to coercion through covert impairments.

The touchscreen display on many electronic voting machines is composed of a transparent touch-sensitive layer placed on top of a display screen. The touch-sensitive layer is typically a resistive sensor, as opposed to the capacitive sensor found on most smartphones (see [91] for a survey of touchscreen technologies). Resistive touchscreens must be calibrated to ensure that the user's tactile input is registered to the correct location on the display. In a calibration attack, an attacker adjusts the calibration settings to either prevent votes from being registered or coerce a user to vote for another candidate. In this way, a miscalibrated device will perform substitutions by sensing the user's tactile input in a different location on the screen, or deletions by disabling parts of the screen altogether [10]. This defect on electronic voting machines has been observed in practice [47], and when performed at scale, this kind of attack has the potential to compromise the election process. Even when properly calibrated, users at various heights voting for the same candidate may touch different parts of the screen due to differences in viewing angle [28].

Impairments that induce

Covert impairments can both frustrate and entertain. They remain an important research tool for better understanding how humans interact with and react to an interface, especially under degraded conditions. As a research tool, the use of covert impairments can induce a particular affective state by making a task more difficult and reveal the limits of the human psychomotor system.

Affective computing

Keystroke and mouse dynamics have been recognized as unobtrusive modalities to measure affect [100]. However, manipulating keyboard and mouse input also has the ability to induce affect. Stress or anxiety may be induced by covert impairments, especially during tasks that involve a performance-based reward. This is useful for research because it enables the measurement of physiological responses in different affective states. Protocols have involved frustrating the user by introducing lag to pointing tasks during gameplay with monetary reward [76]. Likewise, a sense of loss of control is simulated by ignoring some keystrokes [31, 72].

Cognitive science

A general understanding of the psychomotor system has been a major driving force in HCI research, culminating in seminal works such as Fitts' Law [36] and Salthouse's characterization of typing phenomena [75]. Extensions to these efforts have involved testing the limits of the human psychomotor system by intentionally degrading interaction. The presence of lag, between the user's physical action and perceived system state, places a fundamental constraint on the human-computer feedback loop by making interaction tasks more difficult. Variations in lag, or jitter, can further increase this difficulty. The primary mechanism for studying performance degradation is through a covert impairment, such as intentionally introducing lag to the input processing pipeline.

MacKenzie and Ware first characterized Fitts' index of difficulty as a function of lag [53], followed by [69] and [70] wherein the effects of *changes* in lag (jitter) and inserted motion events (spatial jitter) were examined. When subjected to the relatively higher cost to perform an action, [66] observed that users spent more time planning and less time acting to cope with the impairments.

The work of [17] and [94] considered how users make corrections when exposed to several kinds of impairments during a pointing task. In [17], the cursor changed direction of motion (substitution) or suddenly jumped to a new location (insertion) to test the similarity of corrections made through a pointing device to those made with the hand. Reactions to the impairments were surprisingly consistent between subjects, and this work may inform future cursorjacking attacks that exploit the way users react to pointer manipulation [3].

Entertainment

Modifications to peripheral input are prevalent in gaming and entertainment industries. Input events are typically modified so that the system state better matches user intent, for example by making the keyboard seem like an analog device with the use of an attack, decay, sustain, release (ADSR) curve [6].

This technique is ubiquitous in electronic musical instruments wherein an envelope generator creates sound [87]. While these manipulations provide entertainment value, some games impair input to induce a state of excitement and encourage users to part with their money.

The claw crane, a kind of "skill with prize" game with several variants, allows the user to aim a claw that reaches down into a pile of prizes in an attempt to pick up and retrieve a prize. To ensure that not everyone is a winner, impairments are introduced to the button or joystick input. Typically, the claw will apply little force making it impossible to retrieve a prize, except for a small percentage of users for which the claw operates with enough force. Only 1 in 18 users are destined to retrieve a prize, depending on the game and adjustable settings programmed by the vendor [67]. This is a form of substitution, where the user's intended action ("strong claw") is replaced with something else ("weak claw"). Claw crane games have been criticized, and in many places regulated, as there is little difference between these kinds of games and gambling. In response, the American Amusement Machine Association enacted the Fair Play Pledge, part of which specifies that "the player's input controls the outcome of the game" [5].

As a way of providing entertainment value, some games utilize covert impairments to dynamically adjust game difficulty. Rather than an absolute difficulty setting, dynamic difficulty adjustment enables a game to better match the player's skill level as measured by, e.g., player health or accumulated points [44]. The rationale for dynamic difficulty adjustment is that the game should be not too easy nor too difficult; there is a "zone" in which satisfaction and comfort are maximized. Adjustments to game difficulty may be achieved by introducing covert impairments. The primary mechanism for this is substitution, whereby the strength, accuracy, and impact of a player's actions are decreased for players of greater skill, and increased for players of lesser skill [44].

DISCUSSION

Ethical and practical considerations

Covert impairments have legitimate use cases, in both cybersecurity and as a research tool. But because they are intended to influence user behavior, and are used widely for illegitimate purposes, some considerations toward users' well-being must be made.

Involuntary actions: Covert impairments are already widely used with malevolent intent [10, 3]. Beyond web-based attacks, covert impairments have the potential to alter the outcome of critical HCI tasks, such as an election conducted through electronic voting. The core issue is that impairments provide a means to coerce the user to perform an action chosen by the designer. This abuse degrades HCI, and the detection and prevention of such impairments should remain an active area of research [86, 98].

Resentment: Too many impairments can have the effect of inflicting frustration or stress upon the user [76]. Combined with the act of deception, users may feel resentment if they become aware that, e.g., their actions do not have the intended consequences. This is almost surely the case for clickjacking and related attacks, but impairments used in other contexts should be cognizant of potentially inducing negative affect.

Privacy: Unlike other behavioral biometrics which are passive in nature, the use of covert impairments forces a user to interact with the computer in order to correct the impairment. They may serve as a form of access control, but could also be used to track or deanonymize users based upon their behavior. Some recent tools have been developed that aim to limit the effectiveness of such identification and profiling tools by introducing imperceptible perturbations to the input, such as by randomly delaying keystrokes [59]. It is not clear if such tools remain effective when covert impairments are used.

Cost vs return: For applications that use impairments to elicit behavior for a legitimate purpose, such as fraud or bot detection, a careful balance must be struck between the impairment rate and the detection rate. Covert impairments have an implicit cost, which is the decrease in task performance because they require the user to perform some additional action that would not have been performed otherwise. Whether this is acceptable or not may depend on the sensitivity of the application.

Timing: As a practical consideration to using covert impairments as a CAPTCHA, it is necessary to think about when is the right time to introduce an impairment. For example, a designer could choose to insert motion immediately before a pointing task, at the midpoint between the start and the target, or when the pointer nears the target. The work of [94] may inform this choice, which investigated how users react to perturbations introduced immediately before or during various pointing tasks.

Sustainability: Some work has demonstrated that users develop coping mechanisms when input is impaired [66]. For applications that perform user identification, it is not clear whether repeated exposure to the same impairment would result in similar adaptations that increase within-subject differences over time. As users acclimate to the presence of a particular impairment, they may adjust their reaction strategy, or choose to ignore it altogether [9].

In balancing these considerations and determining the extent to which a covert impairment alters the course of interaction, we propose measuring impact at three different levels:

Task impact: Covert impairments consist of three primitive string operations and lag. Therefore, the generalized minimum edit distance provides a metric with which we may compare discrete event sequences before and after the impairment as a way of quantifying the magnitude of the impairment.

User impact: On top of increased task difficulty, impairments may carry costs to the user, such as increased cognitive load or negative affect. These may be measured objectively, e.g., through a user's keystroke dynamics, or subjectively through self reporting. For example, [48] reported significantly higher levels of self-reported frustration in the presence of lag.

Population impact: Because some users might be more or less affected by an impairment than others, the proportion of users vulnerable to impairment is of interest. For example in

[3], 99% of users were vulnerable to a pointer destabilization attack, a kind of substitution impairment. Note that the term "vulnerable" is a bit a misnomer here since one may likewise consider how many users react to a benevolent impairment, such as a CAPTCHA [35].

Detection and countermeasures

Defenses against covert impairments could range from detection, in which the presence of a covert impairment is established and can be avoided if chosen, to prevention, where the mechanism that introduced the impairment is removed or neutralized. These defenses may be considered from two perspectives depending on the designer's intent.

Malevolent impairments: From the perspective of the user, it is desirable to both detect and prevent malevolent impairments. The motivation for doing so could be to mitigate information leaked through behavior for increased privacy [59] or to ensure the design of trustworthy systems. Detection can be performed through closed loop measurements, for example using techniques similar to EchoMouse [22] and LagBox [95] where ground truth events are physically induced on a sensor and then measured on the display. From this, the presence of insertions, deletions, substitutions, or lag can indicate an impairment. Toward prevention, concepts in the W3C UI safety specification [86], such as cursor integrity, can be leveraged. These ideas may be extended to other modalities, such as "caret integrity" for keyboard input, or implemented at other layers, such as within the OS instead of the browser.

Benevolent impairments: From the perspective of the designer, it is desirable to ensure that impairments remain effective and are not circumvented. Designing an effective CAPTCHA (high accuracy, low usability cost) is nontrivial, where successful methods invoke behaviors that have both low within-subject and low between-subject variability. It would also be necessary to consider automation attacks against the impairment, as methods to defeat traditional image-based CAPTCHAs have advanced significantly [38].

Related work

Tangential to covert impairments, several lines of research have emerged that either leverage or examine how changes to an interface influence users.

Concealed information tests: Modification to the UI itself, rather than user input, can be used to elicit behavior from users. When presented with an unexpected question, a user's keystroke dynamics can indicate whether that individual is being deceptive [60]. This could be implemented on a web form, for example, when applying for an insurance policy. Likewise, the concealed information test can reveal deception through mouse pointer motion relative to the strategic placement of UI elements [88].

Reflexive eye movements: Eye movement is an emerging biometric modality. The works of [79] and [80] take this a step further and consider reflexive eye movements in response to changes in the interface. They consider the user's eye movement reactions to a stimulus that jumps around on the screen. Like covert impairments, changes to the UI elicits behavior that can reveal between-subject differences. But these changes occur at the interface level, and not along the input processing pipeline.

Subliminal cueing: Subtle design choices can also be used to persuade users. Like coercion through impairments, subliminal cueing may be performed through the choice or design of UI elements [74, 23, 14, 62]. For example, a barely perceptible image has the potential to prime the outcome of choice tasks [14]. Similarly, the presentation of a stimulus for a duration under the perceptual visual threshold can increase learning performance when the stimulus provides a relevant cue [23]. Again, the main difference from impairments is that these design choices affect the UI itself and not the interaction that takes place.

Difficult interfaces: The design of interfaces that are difficult to use, either intentionally or because of a bug, provides insight on how users react and adapt to impairments. In a technique called "frost brushing", button labels remain hidden until the user hovers the pointer over the label after which the text is revealed [25]. Despite initial decreased task performance, this technique increased engagement and spatial learning. Likewise, in a gesture recognition system with errors intentionally introduced, users were found to adapt to the errors at a rate proportional to the misrecognition rate [9]. Unintentionally difficult interfaces have had other unexpected outcomes, such as increased communication in collaborative environments to foster workarounds for impaired input [73].

CONCLUSIONS AND FUTURE WORK

Impairments are breakdowns between the user's physical actions and the induced system state. In this paper, we systematized four kinds of impairments to HCI, characterized by the three primitive string edit operations: insertions, deletions, substitutions, and a fourth primitive, lag. Unintentional impairments arise from a variety of sources, and covert impairments are deliberate attempts to mimic these.

The use of covert impairments is largely concentrated in cybersecurity, and it is ironic to note that the same technique is used as both an attack and defense. Perturbations to the mouse pointer, as a method to elicit behavior, provide a means to verify the presence of a human user or to reveal the user's identity. However, the same kind of perturbations applied in a malicious way are leveraged to coerce a user into performing an undesirable action, such as clicking on a link. This dichotomy arises based on the context in which the impairments are introduced: in a benign environment on the one hand, and in the presence of malicious content on the other [43].

Within the web security community, the mitigation and detection of covert impairments provide some means of ensuring the integrity of HCI during sensitive tasks, such as voting or logging into a bank website. This work may be informed by some of the research performed within the HCI community that examines user reactions to impairments [94].

ACKNOWLEDGMENTS

Many thanks to the anonymous reviewers for their time and suggestions to improve this paper.

REFERENCES

[1] Eytan Adar, Desney S. Tan, and Jaime Teevan. 2013. Benevolent deception in human computer interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI'13*. ACM Press. DOI:http://dx.doi.org/10.1145/2470654.2466246

[2] Ahmed Awad E. Ahmed and Issa Traore. 2007. A New Biometric Technology Based on Mouse Dynamics. *IEEE Transactions on Dependable and Secure Computing* 4, 3 (jul 2007), 165–179. DOI: http://dx.doi.org/10.1109/tdsc.2007.70207

[3] Devdatta Akhawe, Warren He, Zhiwei Li, Reza Moazzezi, and Dawn Song. 2014. Clickjacking Revisited: A Perceptual View of UI Security. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/woot14/workshop-program/presentation/akhawe

[4] Alan J. Flavell. 2008. FORM submission and the ENTER key? http://web.archive.org/web/20190202111851/http://www.alanflavell.org.uk/www/formquestion.html. (2008). Accessed: 2019-09-02.

[5] American Amusement Machine Association. 2017. Fair play pledge. http://web.archive.org/web/20181215215619/https://coin-op.org/fair-play-pledge/. (2017). Accessed: 2019-09-05.

[6] Andrew Dotsenko. 2017. Designing game controls. http://web.archive.org/web/20180621150152/https://gamedesignframework.net/designing-game-controls/. (2017). Accessed: 2019-09-05.

[7] Apple. 2016. If your pointer is jumpy when you use a trackpad, Magic Trackpad, or Magic Mouse. http://web.archive.org/web/20190828180557/https://support.apple.com/en-us/HT203171. (2016). Accessed: 2019-08-28.

[8] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2010. Predicting the cost of error correction in character-based text entry technologies. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI'10*. ACM Press. DOI: http://dx.doi.org/10.1145/1753326.1753329

[9] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2014. User Adaptation to a Faulty Unistroke-Based Text Entry Technique by Switching to an Alternative Gesture Set. In *Proceedings of Graphics Interface 2014 (GI'14)*. Canadian Information Processing Society, CAN, 183–192.

[10] Adam Aviv, Pavol Černy, Sandy Clark, Eric Cronin, Gaurav Shah, Micah Sherr, and Matt Blaze. 2008. Security evaluation of ES&S voting machines and election management system. In *Proceedings of the conference on Electronic voting technology*. USENIX Association, 11.

[11] Henry S. Baird and Jon L. Bentley. 2005. Implicit CAPTCHAs. In *Document Recognition and Retrieval XII*, Elisa H. Barney Smith and Kazem Taghva (Eds.). SPIE. DOI:http://dx.doi.org/10.1117/12.590944

[12] Ravin Balakrishnan. 2004. "Beating" Fitts' law: virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies* 61, 6 (dec 2004), 857–874. DOI: http://dx.doi.org/10.1016/j.ijhcs.2004.09.002

[13] Ritwik Banerjee, Song Feng, Jun Seok Kang, and Yejin Choi. 2014. Keystroke Patterns as Prosody in Digital Writings: A Case Study with Deceptive Reviews and Essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. DOI:http://dx.doi.org/10.3115/v1/d14-1155

[14] Oswald Barral, Gabor Aranyi, Sid Kouider, Alan Lindsay, Hielke Prins, Imtiaj Ahmed, Giulio Jacucci, Paolo Negri, Luciano Gamberini, David Pizzi, and Marc Cavazza. 2014. Covert Persuasive Technologies: Bringing Subliminal Cues to Human-Computer Interaction. In *Persuasive Technology*. Springer International Publishing, 1–12. DOI: http://dx.doi.org/10.1007/978-3-319-07127-5_1

[15] Eric A. Bier and Maureen C. Stone. 1986. Snap-dragging. *ACM SIGGRAPH Computer Graphics* 20, 4 (aug 1986), 233–240. DOI: http://dx.doi.org/10.1145/15886.15912

[16] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. 2004. Semantic pointing. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI'04*. ACM Press. DOI: http://dx.doi.org/10.1145/985692.985758

[17] Eli Brenner and Jeroen Smeets. 2003. Fast corrections of movements with a computer mouse. *Spatial Vision* 16, 3 (2003), 365–376. DOI: http://dx.doi.org/10.1163/156856803322467581

[18] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C. Mitchell. 2014. The End is Nigh: Generic Solving of Text-based CAPTCHAs. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/woot14/workshop-program/presentation/bursztein

[19] Ana Caraban, Evangelos Karapanos, Daniel Gonçalves, and Pedro Campos. 2019. 23 Ways to Nudge. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI'19*. ACM Press. DOI:http://dx.doi.org/10.1145/3290605.3300733

[20] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1980. The keystroke-level model for user performance time with interactive systems. *Commun. ACM* 23, 7 (jul 1980), 396–410. DOI: http://dx.doi.org/10.1145/358886.358895

[21] John M. Carroll and Judith Reitman Olson. 1988. Mental Models in Human-Computer Interaction. In *Handbook of Human-Computer Interaction*. Elsevier, 45–65. DOI:http://dx.doi.org/10.1016/b978-0-444-70536-5.50007-5

[22] Géry Casiez and Nicolas Roussel. 2011. No more bricolage!: methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST'11*. ACM Press. DOI: http://dx.doi.org/10.1145/2047196.2047276

[23] Pierre Chalfoun and Claude Frasson. 2011. Subliminal Cues While Teaching: HCI Technique for Enhanced Learning. *Advances in Human-Computer Interaction* 2011 (2011), 1–15. DOI: http://dx.doi.org/10.1155/2011/968753

[24] Rahul Chatterjee, Anish Athayle, Devdatta Akhawe, Ari Juels, and Thomas Ristenpart. 2016. pASSWORD tYPOS and How to Correct Them Securely. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE. DOI:http://dx.doi.org/10.1109/sp.2016.53

[25] Andy Cockburn, Per Ola Kristensson, Jason Alexander, and Shumin Zhai. 2007. Hard lessons. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI'07*. ACM Press. DOI: http://dx.doi.org/10.1145/1240624.1240863

[26] Gregory Conti and Edward Sobiesk. 2010. Malicious interface design. In *Proceedings of the 19th international conference on World wide web - WWW'10*. ACM Press. DOI: http://dx.doi.org/10.1145/1772690.1772719

[27] Dan Luu. 2017. Computer latency: 1977-2017. http://web.archive.org/web/20190828214546/ https://danluu.com/input-lag/. (2017). Accessed: 2019-08-28.

[28] Dan Wallach. 2006. Touchscreen Calibration Issues with Voting Machines. http://web.archive.org/web/20190828185257/ http://accurate-voting.org/2006/11/01/ touchscreen-calibration-issues-with-voting-machines/. (2006). Accessed: 2019-08-28.

[29] Dave Dribin. 2000. Keyboard matrix help. http://web.archive.org/web/20190729085531/ https://www.dribin.org/dave/keyboard/one_html/. (2000). Accessed: 2019-09-05.

[30] Declan Delaney, Tomás Ward, and Seamus McLoone. 2006. On Consistency and Network Latency in Distributed Interactive Applications: A Survey—Part I. *Presence: Teleoperators and Virtual Environments* 15, 2 (apr 2006), 218–234. DOI: http://dx.doi.org/10.1162/pres.2006.15.2.218

[31] Holger Diener and Karina Oertel. 2006. Experimental Approach to Affective Interaction in Games. In *Technologies for E-Learning and Digital Entertainment*. Springer Berlin Heidelberg, 507–518. DOI:http://dx.doi.org/10.1007/11736639_62

[32] Andrew Dillon and Charles Watson. 1996. User analysis in HCI — the historical lessons from individual differences research. *International Journal of Human-Computer Studies* 45, 6 (dec 1996), 619–637. DOI:http://dx.doi.org/10.1006/ijhc.1996.0071

[33] Eddy Bordi. 2010. Proof of concept - cursorjacking (noscript). http://web.archive.org/web/20150704223642/ http://static.vulnerability.fr/noscript-cursorjacking.html. (2010). Accessed: 2019-09-02.

[34] Dennis E. Egan. 1988. Individual Differences In Human-Computer Interaction. In *Handbook of Human-Computer Interaction*. Elsevier, 543–568. DOI: http://dx.doi.org/10.1016/b978-0-444-70536-5.50029-4

[35] Nitzan Farhi, Nir Nissim, and Yuval Elovici. 2019. Malboard: A novel user keystroke impersonation attack and trusted detection framework based on side-channel analysis. *Computers & Security* 85 (aug 2019), 240–269. DOI: http://dx.doi.org/10.1016/j.cose.2019.05.008

[36] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6 (1954), 381–391. DOI: http://dx.doi.org/10.1037/h0055392

[37] USB Implementers' Forum. 2001. *Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID)*.

[38] Haichang Gao, Jeff Yan, Fang Cao, Zhengya Zhang, Lei Lei, Mengyun Tang, Ping Zhang, Xin Zhou, Xuqin Wang, and Jiawei Li. 2016. A Simple Generic Attack on Text Captchas. In *Proceedings 2016 Network and Distributed System Security Symposium*. Internet Society. DOI: http://dx.doi.org/10.14722/ndss.2016.23154

[39] L. Giancardo, A. Sánchez-Ferro, T. Arroyo-Gallego, I. Butterworth, C. S. Mendoza, P. Montero, M. Matarazzo, J. A. Obeso, M. L. Gray, and R. San José Estépar. 2016. Computer keyboard interaction as an indicator of early Parkinson's disease. *Scientific Reports* 6, 1 (oct 2016). DOI: http://dx.doi.org/10.1038/srep34468

[40] Colin M. Gray, Yubo Kou, Bryan Battles, Joseph Hoggatt, and Austin L. Toombs. 2018. The Dark (Patterns) Side of UX Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI'18*. ACM Press. DOI: http://dx.doi.org/10.1145/3173574.3174108

[41] Xiao Han, Nizar Kheir, and Davide Balzarotti. 2018. Deception Techniques in Computer Security. *Comput. Surveys* 51, 4 (jul 2018), 1–36. DOI: http://dx.doi.org/10.1145/3214305

[42] Kasper Hornbæk and Antti Oulasvirta. 2017. What Is Interaction?. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI'17*. ACM Press. DOI: http://dx.doi.org/10.1145/3025453.3025765

[43] Lin-Shung Huang, Alex Moshchuk, Helen J. Wang, Stuart Schecter, and Collin Jackson. 2012. Clickjacking: Attacks and Defenses. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. USENIX, Bellevue, WA, 413–428. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/huang

[44] Robin Hunicke. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology - ACE'05*. ACM Press. DOI: http://dx.doi.org/10.1145/1178477.1178573

[45] Jesse Ruderman. 2002. iframe content background defaults to transparent. http://web.archive.org/web/20190918203748/https://bugzilla.mozilla.org/show_bug.cgi?id=154957. (2002). Accessed: 2019-09-18.

[46] Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 2013. How fast is fast enough?. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI'13*. ACM Press. DOI:http://dx.doi.org/10.1145/2470654.2481317

[47] Kim Zetter. 2008. ES&S Voting Machines In Tennessee Flip Votes. http://web.archive.org/web/20190828185618/https://www.wired.com/2008/10/ess-voting-mach/. (2008). Accessed: 2019-08-28.

[48] Jonathan Klein, Youngme Moon, and Rosalind W Picard. 2002. This computer responds to user frustration: Theory, design, and results. *Interacting with computers* 14, 2 (2002), 119–140.

[49] Krzysztof Kotowicz. 2012. Cursorjacking again. http://web.archive.org/web/20190207012845/http://blog.kotowicz.net/2012/01/cursorjacking-again.html. (2012). Accessed: 2019-09-02.

[50] Shao-Tsu Kung and Cheng-Sung Lee. 2005. Cooperative keyboard and touchpad control method. (March 17 2005). US Patent App. 10/605,045.

[51] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[52] Tongbo Luo, Xing Jin, Ajai Ananthanarayanan, and Wenliang Du. 2013. Touchjacking Attacks on Web in Android, iOS, and Windows Phone. In *Foundations and Practice of Security*. Springer Berlin Heidelberg, 227–243. DOI: http://dx.doi.org/10.1007/978-3-642-37119-6_15

[53] I. Scott MacKenzie and Colin Ware. 1993. Lag as a determinant of human performance in interactive systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI'93*. ACM Press. DOI:http://dx.doi.org/10.1145/169059.169431

[54] Marcus Niemietz. 2011. Cursorjacking. http://web.archive.org/web/20171104074428/http://www.mniemietz.de/demo/cursorjacking/cursorjacking.html. (2011). Accessed: 2019-09-02.

[55] Robert E Marin. 1975. Debounce logic for keyboard. (May 27 1975). US Patent 3,886,543.

[56] Arunesh Mathur, Gunes Acar, Michael J. Friedman, Elena Lucherini, Jonathan Mayer, Marshini Chetty, and Arvind Narayanan. 2019. Dark Patterns at Scale. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (nov 2019), 1–32. DOI: http://dx.doi.org/10.1145/3359183

[57] Roy A. Maxion and Robert W. Reeder. 2005. Improving user-interface dependability through mitigation of human error. *International Journal of Human-Computer Studies* 63, 1-2 (jul 2005), 25–50. DOI:http://dx.doi.org/10.1016/j.ijhcs.2005.04.009

[58] Michal Zalewski. 2010. The curse of inverse strokejacking. https://web.archive.org/web/20181004042954/http://lcamtuf.blogspot.com/2010/06/curse-of-inverse-strokejacking.html. (2010). Accessed: 2019-09-05.

[59] John V Monaco and Charles C Tappert. 2016. Obfuscating keystroke time intervals to avoid identification and impersonation. *arXiv preprint arXiv:1609.07612* (2016).

[60] Merylin Monaro, Chiara Galante, Riccardo Spolaor, Qian Qian Li, Luciano Gamberini, Mauro Conti, and Giuseppe Sartori. 2018. Covert lie detection using keyboard dynamics. *Scientific Reports* 8, 1 (jan 2018). DOI:http://dx.doi.org/10.1038/s41598-018-20462-6

[61] Mutally Human. 2018. The Curious Case of Cursor Jumping. http://web.archive.org/web/20190828174408/https://www.mutuallyhuman.com/blog/the-curious-case-of-cursor-jumping/. (2018). Accessed: 2019-08-28.

[62] Paolo Negri, Luciano Gamberini, and Simone Cutini. 2014. A Review of the Research on Subliminal Techniques for Implicit Interaction in Symbiotic Systems. In *Symbiotic Interaction*. Springer International Publishing, 47–58. DOI: http://dx.doi.org/10.1007/978-3-319-13500-7_4

[63] Allen Newell. 1994. *Unified theories of cognition*. Harvard University Press.

[64] Marcus Niemietz and Jörg Schwenk. 2012. Ui redressing attacks on android devices. *Black Hat Abu Dhabi* (2012).

[65] Donald A Norman. 2014. Some observations on mental models. In *Mental models*. Psychology Press, 15–22.

[66] KENTON P. O'HARA and STEPHEN J. PAYNE. 1999. Planning and the user interface: the effects of lockout time and error recovery cost. *International Journal of Human-Computer Studies* 50, 1 (jan 1999), 41–59. DOI:http://dx.doi.org/10.1006/ijhc.1998.0234

[67] OK Manufacturing. 2008. Crane operating instructions. http://web.archive.org/web/20080725032406/ http://www.okmfg.net/pdf/manuals/cranes_manual.pdf. (2008). Accessed: 2019-09-05.

[68] Paul Podlipensky. 2012. Cursor Spoofing and Cursorjacking. http://web.archive.org/web/20180727115830/ http://podlipensky.com/2012/08/cursor-spoofing-cursorjacking/. (2012). Accessed: 2019-09-02.

[69] Andriy Pavlovych and Wolfgang Stuerzlinger. 2009. The tradeoff between spatial jitter and latency in pointing tasks. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems - EICS'09*. ACM Press. DOI: http://dx.doi.org/10.1145/1570433.1570469

[70] Andriy Pavlovych and Wolfgang Stuerzlinger. 2011. Target following performance in the presence of latency, jitter, and signal dropouts. In *Proceedings of Graphics Interface 2011*. Canadian Human-Computer Communications Society, 33–40.

[71] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The go-go interaction technique. In *Proceedings of the 9th annual ACM symposium on User interface software and technology - UIST'96*. ACM Press. DOI: http://dx.doi.org/10.1145/237091.237102

[72] Boris Reuderink, Anton Nijholt, and Mannes Poel. 2009. Affective Pacman: A Frustrating Game for Brain-Computer Interface Experiments. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Berlin Heidelberg, 221–227. DOI: http://dx.doi.org/10.1007/978-3-642-02315-6_23

[73] Yann Riche, Nathalie Henry Riche, Petra Isenberg, and Anastasia Bezerianos. 2010. Hard-to-use interfaces considered beneficial (some of the time). In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA'10*. ACM Press. DOI: http://dx.doi.org/10.1145/1753846.1753855

[74] William B. Rouse. 1981. Human-Computer Interaction in the Control of Dynamic Systems. *Comput. Surveys* 13, 1 (jan 1981), 71–99. DOI: http://dx.doi.org/10.1145/356835.356839

[75] Timothy A. Salthouse. 1986. Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological Bulletin* 99, 3 (1986), 303–319. DOI: http://dx.doi.org/10.1037/0033-2909.99.3.303

[76] Jocelyn Scheirer, Raul Fernandez, Jonathan Klein, and Rosalind W Picard. 2002. Frustrating the user on purpose: a step toward building an affective computer. *Interacting with Computers* 14, 2 (feb 2002), 93–118. DOI: http://dx.doi.org/10.1016/s0953-5438(01)00059-5

[77] Prakash Shrestha, Nitesh Saxena, Ajaya Neupane, and Kiavash Satvat. 2019. CATCHA: When Cats Track Your Movements Online. In *Information Security Practice and Experience*. Springer International Publishing, 172–193. DOI: http://dx.doi.org/10.1007/978-3-030-34339-2_10

[78] Stephen Simons, Jiangying Zhou, Yuwei Liao, Laura Bradway, Mario Aguilar, and Patrick M Connolly. 2014. Cognitive biometrics using mouse perturbation. (March 20 2014). US Patent App. 14/011,351.

[79] Ivo Sluganovic, Marc Roeschlin, Kasper B. Rasmussen, and Ivan Martinovic. 2016. Using Reflexive Eye Movements for Fast Challenge-Response Authentication. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*. ACM Press. DOI: http://dx.doi.org/10.1145/2976749.2978311

[80] Ivo Sluganovic, Marc Roeschlin, Kasper B. Rasmussen, and Ivan Martinovic. 2018. Analysis of Reflexive Eye Movements for Fast Replay-Resistant Biometric Authentication. *ACM Transactions on Privacy and Security* 22, 1 (nov 2018), 1–30. DOI: http://dx.doi.org/10.1145/3281745

[81] Alistair Sutcliffe, Michele Ryan, Ann Doubleday, and Mark Springett. 2000. Model mismatch analysis: Towards a deeper explanation of users' usability problems. *Behaviour & Information Technology* 19, 1 (jan 2000), 43–55. DOI: http://dx.doi.org/10.1080/014492900118786

[82] Shari Trewin, Simeon Keates, and Karyn Moffatt. 2006. Developing steady clicks:. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility - Assets'06*. ACM Press. DOI:http://dx.doi.org/10.1145/1168987.1168993

[83] Avi Turgeman. 2015. System, device, and method of detecting identity of a user of a mobile electronic device. (Jan. 20 2015). US Patent 8,938,787.

[84] Avi Turgeman. 2016. Differentiating among users based on responses to injected interferences. (Dec. 15 2016). US Patent App. 15/212,234.

[85] Avi Turgeman and Frances Zelazny. 2017. Invisible challenges: the next step in behavioural biometrics? *Biometric Technology Today* 2017, 6 (jun 2017), 5–7. DOI: http://dx.doi.org/10.1016/s0969-4765(17)30114-5

[86] Peleus Uhley. 2012. Clickjacking threats. http://web.archive.org/web/20190905213616/https://www.w3.org/Security/wiki/Clickjacking_Threats. (2012). Accessed: 2019-09-05.

[87] Mark Vail. 2014. *The synthesizer: a comprehensive guide to understanding, programming, playing, and recording the ultimate electronic music instrument.* Oxford University Press.

[88] Joseph S Valacich, Jeffrey L Jenkins, Jay F Nunamaker Jr, Salim Hariri, and John Howie. 2013. Identifying insider threats through monitoring mouse movements in concealed information tests. In *Hawaii International Conference on System Sciences. Deception Detection Symposium.*

[89] Amber van der Heijden and Luca Allodi. 2019. Cognitive Triaging of Phishing Attacks. In *28th USENIX Security Symposium (USENIX Security 19).* USENIX Association, Santa Clara, CA, 1309–1326. https://www.usenix.org/conference/usenixsecurity19/presentation/van-der-heijden

[90] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. 2003. CAPTCHA: Using Hard AI Problems for Security. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 294–311. DOI: http://dx.doi.org/10.1007/3-540-39200-9_18

[91] Geoff Walker. 2012. A review of technologies for sensing contact location on the surface of a display. *Journal of the Society for Information Display* 20, 8 (aug 2012), 413–440. DOI: http://dx.doi.org/10.1002/jsid.100

[92] Neff Walker, Jeff Millians, and Aileen Worden. 1996. Mouse Accelerations and Performance of Older Computer Users. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 40, 3 (oct 1996), 151–154. DOI: http://dx.doi.org/10.1177/154193129604000310

[93] Mick West. 2008. Intelligent mistakes: how to incorporate stupidity into your AI code. *Game Developer Magazine - Digital Edition* (2008).

[94] Leonie Oostwoud Wijdenes, Eli Brenner, and Jeroen B. J. Smeets. 2011. Fast and fine-tuned corrections when the target of a hand movement is displaced. *Experimental Brain Research* 214, 3 (aug 2011), 453–462. DOI: http://dx.doi.org/10.1007/s00221-011-2843-4

[95] Raphael Wimmer, Andreas Schmid, and Florian Bockes. 2019. On the Latency of USB-Connected Input Devices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI'19.* ACM Press. DOI: http://dx.doi.org/10.1145/3290605.3300650

[96] Jeff Yan and Ahmad Salah El Ahmad. 2008. A low-cost attack on a Microsoft captcha. In *Proceedings of the 15th ACM conference on Computer and communications security - CCS'08.* ACM Press. DOI: http://dx.doi.org/10.1145/1455770.1455839

[97] Catherine Zanbaka, Paula Goolkasian, and Larry Hodges. 2006. Can a virtual cat persuade you?. In *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI'06.* ACM Press. DOI:http://dx.doi.org/10.1145/1124772.1124945

[98] Mingxue Zhang, Wei Meng, Sangho Lee, Byoungyoung Lee, and Xinyu Xing. 2019. All Your Clicks Belong to Me: Investigating Click Interception on the Web. In *28th USENIX Security Symposium (USENIX Security 19).* USENIX Association, Santa Clara, CA, 941–957. https://www.usenix.org/conference/usenixsecurity19/presentation/zhang

[99] H. Zimmermann. 1980. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications* 28, 4 (apr 1980), 425–432. DOI: http://dx.doi.org/10.1109/tcom.1980.1094702

[100] Philippe Zimmermann, Sissel Guttormsen, Brigitta Danuser, and Patrick Gomez. 2003. Affective Computing—A Rationale for Measuring Mood With Mouse and Keyboard. *International Journal of Occupational Safety and Ergonomics* 9, 4 (jan 2003), 539–551. DOI: http://dx.doi.org/10.1080/10803548.2003.11076589