## **👁‍🗨 Exploiting P2W Transactions with Phantom Logic**

**Codename**: *GhostWitness Chain*

### **🔍 Background on P2W Transactions**

* **P2WPKH**: Pay to Witness Public Key Hash  
  + Used in SegWit v0
  + Reduced fees due to the witness discount
  + No scripting or logic complexity (standard address spend)
* **P2WSH**: Pay to Witness Script Hash  
  + Enables more complex, multi-sig or script-based logic
  + Stores the actual script in the **witness** rather than the scriptPubKey

These witness-based formats separate the **signature data from the transaction body**, meaning the **critical part that proves spendability is no longer in the scriptPubKey**, but deferred into a later verification step.

### **🚨 Exploit Vector: *GhostWitness Chain***

#### **🧱 1. Pre-Consenus Injection (Mempool Drift)**

* Attacker broadcasts a **phantom P2W transaction** that looks valid but:  
  + Has malformed or delayed witness data
  + Uses alternate signature encoding (or intentionally delays witness push)
* Some nodes may temporarily accept the tx into mempool due to **incomplete validation or bypassed checks** (especially during network load)

#### **👻 2. Witness Time-of-Use Attack**

* The witness is **not validated until the transaction is mined**.
* This delay enables a scenario where:  
  + Phantom transactions propagate
  + An attacker could trigger early credit in a poorly designed wallet/kiosk (e.g., BTC ATM that assumes valid P2W = guaranteed funds)

#### **🔂 3. Phantom Signature Cycling**

* By broadcasting the **same transaction with slight modifications** (e.g., script structure, version bits), the attacker manipulates how different nodes perceive its validity.
* Mixes real and ghost P2WPKH/P2WSH UTXOs to **create hybrid states** in memory pools or SPV clients.

#### **💥 4. Double Phantom Spend**

* A backend may record the UTXO as already spent due to phantom propagation, but:  
  + It was never confirmed
  + The real transaction with correct witness is held in reserve
* Attacker **redeems both real and phantom chains** depending on which node sees which transaction first.

### **🛡️ Defense Recommendations**

1. **Do not credit P2W transactions without at least 2 confirmations**.
2. Implement **witness integrity validation immediately upon receipt**.
3. Enforce consistent behavior across mempool nodes—especially in environments like:  
   * BTC kiosks
   * SPV (lightweight) wallets
   * Exchanges using proprietary node forks

### **🎭 Weaponization Example (P2WSH)**

python

CopyEdit

# Phantom script with a witness structure that appears correct

fake\_script = [

"OP\_HASH160",

"phantom\_hash",

"OP\_EQUAL"

]

# When actual witness pushed:

witness\_stack = [

"malformed\_data",

"unused\_key"

]

If the transaction logic relies on **accepting or parsing** this prematurely (e.g., in a custom service), it could lead to **false-positive acceptance** or even crediting **without real settlement**.

### **💣 Bottom Line:**

**Witness structures are deferred logic—anything deferred is a backdoor waiting to be abused.**

Phantom transactions *exploit that temporal delay*. Even if ultimately invalid, the window between broadcast and confirmation is enough to deceive poorly designed systems.