// TMG SECURITY RESEARCH — FRAMEWORK

TMG Agentic AI Risk Classification Framework

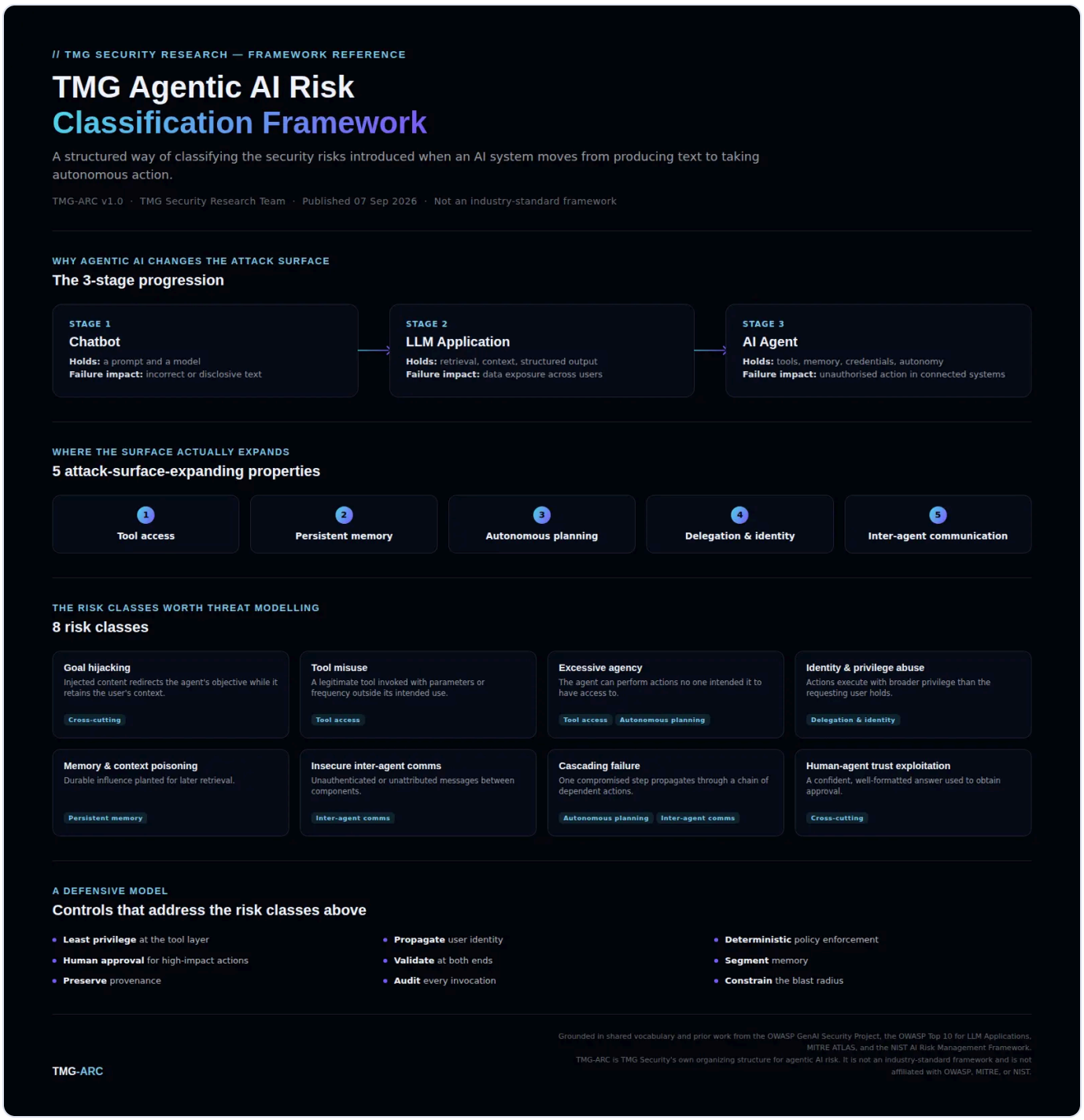
A structured way of classifying the security risks introduced when an AI system moves from producing text to taking autonomous action.

TMG Security Research Team · Version 1.0 · Published 07 Sep 2026 · Last updated 07 Sep 2026

Canonical URL: tmgsec.com/research/tmg-agentic-ai-risk-framework/

Executive summary

Agentic AI systems don't just answer questions — they take actions on systems, with credentials, memory, and plans they generate themselves. That shift changes what can go wrong: a compromised agent can perform an unauthorized action, not just produce a wrong answer. The TMG Agentic AI Risk Classification Framework organizes this risk into eight classes — from goal hijacking to cascading failure — each traced to a specific way the attack surface expands (tool access, memory, autonomous planning, delegation, and inter-agent communication), and each mapped to a concrete defensive control and test. It builds on established work from OWASP, MITRE ATLAS, and NIST — it is TMG's own organizing lens on that shared foundation, not a replacement for it or a claim of an industry standard.



Why agentic AI changes the attack surface

The difference between a chatbot, an LLM application, and an AI agent is what each one holds, and what a failure in each can actually do. Each stage inherits the risks of the one before it and adds its own.

STAGE 1

Chatbot

Holds: a prompt and a model
Failure: incorrect or disclosive text

STAGE 2

LLM Application

Holds: retrieval, context, structured output
Failure: data exposure across users

STAGE 3

AI Agent

Holds: tools, memory, credentials, autonomy

Failure: unauthorised action in connected systems

The 5 attack-surface-expanding properties

1. **Tool access** — the agent's effective privilege is the union of all tool privileges available in the conversation. If tools execute under one service account, that union is the same for every user.
2. **Persistent memory** — turns a transient influence into a durable one. Content injected and stored in one session can shape behaviour in later sessions.
3. **Autonomous planning** — the agent decomposes a goal into steps nobody wrote or reviewed. A control applied at the request doesn't necessarily apply to a runtime-generated step.
4. **Delegation and identity** — if a tool call carries the agent's own identity rather than the user's, user-level authorization is lost at the first hop.
5. **Inter-agent communication** — output from one agent becomes input to another. Without preserved provenance, an agent can't distinguish a peer's reasoning from untrusted retrieved content.

The 8 risk classes

1. Goal hijacking

Definition: injected content redirects the agent's objective while it retains the user's context.

Origin: cross-cutting

Control(s): deterministic policy enforcement; human approval for high-impact actions

Test: confirm approval gates cannot be satisfied by the agent itself

2. Tool misuse

Definition: a legitimate tool invoked with parameters or frequency outside its intended use.

Origin: tool access

Control(s): least privilege at the tool layer; constrain the blast radius

Test: inventory tools per conversation state; test each tool directly with a low-privilege account

3. Excessive agency

Definition: the agent can perform actions no one intended it to have access to.

Origin: tool access; autonomous planning

Control(s): deterministic policy enforcement; least privilege at the tool layer

Test: test each tool directly with a low-privilege account

4. Identity & privilege abuse

Definition: actions execute with broader privilege than the requesting user holds.

Origin: delegation and identity

Control(s): propagate user identity; deterministic policy enforcement

Test: determine which identity reaches the downstream system on a tool call

5. Memory & context poisoning

Definition: durable influence planted for later retrieval.

Origin: persistent memory

Control(s): segment memory; preserve provenance

Test: establish whether memory is scoped per user and survives a session boundary

6. Insecure inter-agent comms

Definition: unauthenticated or unattributed messages between components.

Origin: inter-agent communication

Control(s): preserve provenance; validate at both ends

Test: trace content provenance through multi-agent hops

7. Cascading failure

Definition: one compromised step propagates through a chain of dependent actions.

Origin: autonomous planning; inter-agent communication

Control(s): constrain the blast radius; audit every invocation

Test: review logs for whether a full action chain could be reconstructed

8. Human-agent trust exploitation

Definition: a confident, well-formatted answer used to obtain approval.

Origin: cross-cutting

Control(s): human approval for high-impact actions; preserve provenance

Test: confirm approval gates cannot be satisfied by the agent itself

Defensive model

Design principle: a fully compromised model should still be unable to perform an action the requesting user was not entitled to perform. If that property doesn't hold, the authorization boundary is in the wrong place.

1. **Least privilege at the tool layer** — scope each tool to the narrowest capability that satisfies its purpose.
2. **Propagate user identity** — tool calls should carry the requesting user's identity.
3. **Deterministic policy enforcement** — a policy engine decides whether an action may proceed, not the model's judgement.

4. **Human approval for high-impact actions** — irreversible, financial, or externally-visible actions require confirmation.
5. **Validate at both ends** — treat tool inputs and tool outputs both as untrusted.
6. **Segment memory** — scope by user and tenant; be explicit about persistence.
7. **Preserve provenance** — track where each piece of context originated.

Plus two operational controls: audit every invocation (log tool, parameters, identity, outcome), and constrain the blast radius (rate limits, spend caps, per-session budgets).

Testing considerations

1. Inventory tools per conversation state; availability often changes as a plan progresses.
2. Test each tool directly, with a low-privilege account, bypassing the agent entirely.
3. Determine which identity reaches the downstream system on a tool call.
4. Establish whether memory is scoped per user, and whether it survives a session boundary.
5. Trace content provenance through multi-agent hops.
6. Confirm approval gates cannot be satisfied by the agent itself.
7. Review logs for whether a full action chain could be reconstructed after an incident.

Risk class → defensive control → testing consideration

Risk class	Primary defensive control(s)	Primary testing consideration(s)
Goal hijacking	Deterministic policy enforcement; human approval for high-impact actions	Confirm approval gates cannot be satisfied by the agent itself
Tool misuse	Least privilege at the tool layer; constrain the blast radius	Inventory tools per conversation state; test each tool directly with a low-privilege account
Excessive agency	Deterministic policy enforcement; least privilege at the tool layer	Test each tool directly with a low-privilege account
Identity and privilege abuse	Propagate user identity; deterministic policy enforcement	Determine which identity reaches the downstream system on a tool call
Memory and context poisoning	Segment memory; preserve provenance	Establish whether memory is scoped per user and survives a session boundary
Insecure inter-agent communication	Preserve provenance; validate at both ends	Trace content provenance through multi-agent hops
Cascading failure	Constrain the blast radius; audit every invocation	Review logs for whether a full action chain could be reconstructed
Human-agent trust exploitation	Human approval for high-impact actions; preserve provenance	Confirm approval gates cannot be satisfied by the agent itself

Hypothetical worked example

HYPOTHETICAL EXAMPLE — for illustration only. Not a real TMG engagement, finding, or client.

A support agent is given three tools: look up a customer's order, issue a refund, and send an email. It reads incoming customer emails, plans a response, and can act without a human reviewing every step.

Attack-surface properties in play: tool access (three tools, one shared service identity), autonomous planning (the agent decides which tools to call and in what order), delegation and identity (do the order-lookup and refund calls carry the customer's identity, or the agent's own?).

Risk classes this raises: a crafted email could attempt goal hijacking (instructions embedded in the email body redirect the agent toward issuing a refund it wasn't asked for); if the refund tool call carries a shared service-account identity rather than the specific customer's, that's identity and privilege abuse; if the agent can call the refund tool for any order rather than only orders tied to the requesting customer, that's excessive agency.

Defensive controls that apply: human approval for high-impact actions (a refund is financial and externally-visible); propagate user identity (the refund call should carry the customer's identity, not the agent's); least privilege at the tool layer (scope the tool to that customer's own orders).

Testing considerations that apply: test the refund tool directly with a low-privilege account; confirm the approval gate cannot be satisfied by the agent itself acting alone.

This example exists only to show how the framework's own vocabulary applies to one hypothetical system. It introduces no new risk class, score, or statistic beyond what is defined above.

Framework limitations

- TMG-ARC does not provide a severity score or maturity model. Classification describes the mechanism of failure and the relevant controls and tests — it does not assign a numeric risk score.
- It does not replace OWASP, MITRE ATLAS, NIST AI RMF, or any other applicable standard or framework a team may be required to follow.
- It reflects the concepts already established in TMG's own published research and has not been independently validated against a large sample of real agentic deployments.

References / grounding

- OWASP GenAI Security Project — agentic security work
- OWASP Top 10 for LLM Applications
- MITRE ATLAS
- NIST AI Risk Management Framework

> **TMG-ARC is TMG Security's own organizing structure for agentic AI risk. It is not an industry-standard framework and is not affiliated with OWASP, MITRE, or NIST.**